Semantische Modellierung und Reasoning für Kontextinformationen in Infrastrukturnetzen

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik

der Ludwig-Maximilians-Universität München

Florian Fuchs

16. September 2008

1. Berichterstatterin:	Prof. Dr. Claudia Linnhoff-Popien, Ludwig-Maximilians-Universität München
2. Berichterstatter:	Prof. Dr. Alexander Schill, Technische Universität Dresden
Tag des Rigorosums:	21. Oktober 2008

Danksagung

Diese Arbeit entstand am Lehrstuhl für Mobile und Verteilte Systeme des Instituts für Informatik an der Ludwig-Maximilians-Universität München und bei Siemens Corporate Technology im Fachzentrum für Intelligente Autonome Systeme.

Herzlich bedanken möchte ich mich bei Prof. Dr. Claudia Linnhoff-Popien, die sich bereit erklärte, meine Arbeit von universitärer Seite zu betreuen, und mir bei der Wahl und Bearbeitung meines Themas viele Freiheiten ließ. Gleichzeitig hatte sie für Fragen und Probleme stets ein offenes Ohr und stand mir mit Rat und Tat zur Seite. Prof. Dr. Alexander Schill danke ich für die bereitwillige Übernahme des Zweitgutachtens.

Mein ganz besonderer Dank gilt meinem Betreuer bei Siemens, Dr. Michael Berger, für die vielen Impulse, die er meiner Arbeit gab, sein Vertrauen und die vielfältigen Möglichkeiten, die er mir bei Siemens bot. Außerdem danke ich Siemens für die finanzielle Unterstützung während meiner gesamten Promotion.

Vielmals bedanken möchte ich mich bei meinen Wegbegleitern und Kollegen: Nicht nur der fachliche Austausch an der Universität – unter anderem bei unvergesslichen Lehrstuhl-Workshops – war für mich sehr wertvoll. Bei Siemens genoss ich die tolle Integration in das Team und die anregenden fachlichen Diskussionen. Besonders danke ich Michael Pirker und seinem unvergleichlichen Motivationstalent. Erwähnen möchte ich außerdem unsere Diplomanden und Werkstudenten sowie die Kollegen beim EU-Projekt InteGRail.

Nicht zuletzt bin ich meinen Eltern dankbar, die es meisterten, mir die Verfolgung all meiner Vorhaben zu ermöglichen. Christiane Dargatz gilt mein Dank für ihr Verständnis, ihre Unterstützung und ihre Fröhlichkeit.

München, im November 2008

Zusammenfassung

Infrastrukturen wie Verkehrs- und Energienetze bilden das Rückgrat unserer Gesellschaft und Wirtschaft. Präzises Wissen über den aktuellen technischen Zustand der Infrastrukturkomponenten gilt als Grundvoraussetzung zur Befriedigung des ständig wachsenden Kapazitätsbedarfs und zur Erhöhung der Kosteneffizienz, insbesondere bei der Instandhaltung. Zwar liefern Fernüberwachungssysteme verschiedener Organisationen bereits heute unterschiedlichste Statusinformationen. Es fehlt jedoch ein generischer Ansatz zur integrierten Auswertung dieser Daten, um komplexe Gesamtzustände der Infrastrukturkomponenten abzuleiten.

Diese Arbeit versteht die Zustandsüberwachung für Infrastrukturnetze als ein *kontextsensitives System* im Sinne der *Ambient Intelligence (Umgebungsintelligenz)*: Fernüberwachungssysteme liefern *Kontextinformationen*, und anstelle der Situation einer Entität soll damit der *Zustand eines Überwachungsobjekts* ermittelt werden. Da sich hierfür bei kontextsensitiven Systemen wissensbasierte Ansätze bewährt haben, überträgt diese Arbeit einen solchen Ansatz auf die Zustandsüberwachung in Infrastrukturnetzen. Damit sollen generische Verfahren sowohl zur Integration als auch zur Auswertung (*Reasoning*) von Kontextinformationen in Infrastrukturnetzen konzipiert und umgesetzt werden.

Eine Analyse von Schienen- und Stromnetzen identifiziert als Anforderungen unter anderem die Interoperabilität der Kontextinformationen zwischen Systemen und Betreibern sowie die Möglichkeit, auch komplexe Zustände ableiten zu können. Die Standards des *Semantic Web* auf Basis der Beschreibungslogik $\mathcal{SHIN}$ bieten hierfür eine attraktive Grundlage und gewährleisten sowohl die Umsetzbarkeit als auch die Zukunftstüchtigkeit.

Bei der Erstellung beschreibungslogischer Modelle (*Ontologien*) benötigen Domänenexperten Unterstützung: Dazu wird ein Entwurfsmuster entwickelt, mit dem sie hochwertige Domänenontologien für Kontextinformationen und Infrastrukturzustände modellieren. Ein Architekturmuster ermöglicht es, mehrere Domänenontologien zu einer konsistenten Systemontologie zu integrieren.

Für die automatisierte Auswertung (*Reasoning*) müssen die Besonderheiten von Infrastrukturnetzen berücksichtigt werden: Einerseits fallen Kontextinformationen von Überwachungssystemen räumlich verteilt und bei verschiedenen Organisationen an. Deshalb werden Verfahren entwickelt, die konjunktive Anfragen auch bei verteilten

Wissensbasen korrekt und vollständig beantworten. Dies wird theoretisch gezeigt und praktisch evaluiert. Andererseits müssen topologiebezogene Anfragen beantwortet werden, wie die Suche nach optimalen Pfaden und k-nächsten Nachbarn. Dazu wird eine hierarchische Modellierung des Infrastrukturnetzes entwickelt. Ein generisches Konzept ermöglicht es, damit verschiedene Verfahren für topologiebezogene Anfragen umzusetzen.

Zur praktischen Umsetzung dieser Konzepte in einem Zustandsüberwachungssystem wird eine geschichtete Systemarchitektur spezifiziert. Ein Fallbeispiel aus dem europäischen Schienenverkehr zeigt ihre Realisierung: Mehrere Organisationen stellen unter anderem Achslast-, Gleisgeometrie- und Schienenprofilmessungen als Kontextinformationen zur Verfügung. Unabhängig von deren Verteilung über ganz Europa werten die entwickelten Reasoningverfahren die Semantik der Systemontologie aus und demonstrieren so die zustandsorientierte Wartung des Schienennetzes.

Abstract

Infrastructures such as transportation and energy networks form the backbone of our society and economy. Precise knowledge about the current technical condition of infrastructure components is considered a key prerequisite for coping with the ever growing capacity demand and for increasing cost efficiency, particularly of maintenance. Already today, various types of monitoring data are produced by existing remote monitoring systems operated by different organizations. What is missing, however, is a generic approach for the integrated interpretation of this data in order to derive complex overall conditions of infrastructure components.

This thesis understands condition monitoring for infrastructure networks as a *context-aware system* in the sense of *Ambient Intelligence*: Remote monitoring systems produce *context information*, and instead of using it for determining the situation of an entity, the *condition of a monitoring subject* is desired. As knowledge-based approaches proved successful in the case of context-aware systems, this work applies such an approach to condition monitoring for infrastructure networks. The goal is to develop and implement generic techniques for interpreting and reasoning on context information in infrastructure networks.

An analysis of railway and power networks identifies requirements including the interoperability of context information between systems and operators as well as the automatic derivation of complex conditions. Adopting the standards of the *Semantic Web*, which are based on the Description Logic $\mathcal{SHIN}$, ensures both the implementability and future usability of the approach.

Domain experts require support for engineering Description Logic models (*ontologies*). A design pattern is presented, which enables them to develop high-quality domain ontologies for context information and infrastructure conditions. Using an architecture pattern, multiple domain ontologies can be integrated into a consistent system ontology.

Automated reasoning requires to take into account the characteristics of infrastructure networks: On the one hand, context information is produced by monitoring systems in a distributed manner and at different organizations. This work therefore develops techniques for answering conjunctive queries in a sound and complete way even if knowledge bases are distributed. This is shown theoretically and evaluated practically. On the other hand, topology-related queries have to be answered, such as the search for optimal paths or k-nearest neighbors. To this end, a hierarchical model of the

infrastructure network is developed. Based on a generic approach, this allows to implement different algorithms for answering topology-related queries.

Finally, a layered system architecture is specified, which enables the practical realization of the proposed concepts in a condition monitoring system. This is shown by a case study on the European railway system: Several organizations provide context information as measurements of wheel impact load, track geometry, and rail profile, among others. Despite being distributed across Europe, the developed reasoning techniques evaluate this context information with respect to the semantics formalized in the system ontology. Based on this, condition-based maintenance of the railway infrastructure is demonstrated.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einführung

„Infrastrukturen sind mehr als nur die materielle Voraussetzung für wirtschaftliches Leben. Sie sind gespeicherter gesellschaftlicher Reichtum und bestimmen über die Teilhabe am Gemeinwesen“ – so beschreiben Reinhard Loske und Roland Schaeffer die Bedeutung von Infrastrukturnetzen für unser Leben in dem Sammelband „Die Zukunft der Infrastrukturen“ [LS05]. Zu den *materiellen* oder *netzgebundenen* Infrastrukturen (manchmal auch *Netz-Infrastrukturen* oder *physische Infrastrukturnetze*) zählen Verkehrsinfrastrukturen wie Schiene, Straße, Luft und See, Energieinfrastrukturen wie Strom- und Gasnetze sowie Wasserversorgungs- und -entsorgungsinfrastrukturen. Diese sind im Laufe der Zeit mit den zunehmenden Bedürfnissen ihrer Nutzer gewachsen. Um ihrer Rolle auch in Zukunft gerecht werden zu können, sind bei der Bereitstellung dieser Infrastrukturen grundlegende Veränderungen nötig.

Dieses Kapitel gibt einen Überblick über aktuelle und zukünftige Herausforderungen für die Betreiber von Infrastrukturnetzen. Daraus wird die Aufgabenstellung dieser Arbeit motiviert und eine Vorschau auf die entwickelten Lösungsbeiträge gegeben. Dazu beziehen sich die folgenden Beispiele auf Schienen- und Stromnetze als typische Vertreter von Verkehrs- und Energieinfrastrukturen.

Herausforderungen für Infrastrukturnetze der Zukunft. Alle Prognosen stimmen überein: Betreiber von Infrastrukturnetzen müssen immer höhere Kapazitäten bereitstellen, sie müssen ihre Netze verstärkt mit denen anderer Betreiber und über Landesgrenzen hinweg integrieren, und gleichzeitig muss die Kosteneffizienz von Betrieb und Wartung stetig gesteigert werden. Diese Vorhersagen werden unter anderem durch die folgenden Fakten gestützt.

Weil Infrastrukturnetze bereits heute an ihrer Belastungsgrenze betrieben werden, müssen sie in Zukunft *höhere Kapazitäten* zur Verfügung stellen: So musste das britische Schienennetz in den letzten zehn Jahren einen Anstieg der Personenkilometer um 42 Prozent und der Tonnenkilometer um 58 Prozent verkraften [Oll06]. Für den gesamten europäischen Schienenverkehr wird bis zum Jahr 2020 eine Erhöhung der Personenkilometer um 40 Prozent und der Tonnenkilometer um sogar 70 Prozent im Vergleich zu 2000 prognostiziert. Das bedeutet, dass sich das Marktvolumen des schienenbasierten Passagier- und Güterverkehrs verdreifachen und der Anteil am Gesamtverkehrsmarkt damit verdoppeln würde [SRRA02].

Auch Elektrizitätsnetze werden immer stärker strapaziert: So hat sich die Last auf die europäischen Stromnetze seit den 1960er Jahren mehr als verdoppelt und ist damit stärker gestiegen als prognostiziert und als bei der Planung der Netze vorgesehen. Zudem sind die meisten Komponenten für eine Lebensdauer von höchstens 40 Jahren ausgelegt [SRAE07]. Dies äußert sich in einer zunehmenden Zahl von Stromausfällen, die sich vor allem in den USA häufen [Maz05, GW04]. Bis zum Jahr 2030 wird schließlich

weiterhin von einem Wachstum des Stromverbrauchs um jährlich 1,4 Prozent allein in Europa ausgegangen [SRAE06].

Als besonders kritisch stellt sich die Kapazitätserhöhung für städtische Infrastrukturen dar: In der wachsenden Zahl sogenannter *Megacities* – Städten mit mehr als 10 Millionen Einwohnern – können Kapazitätserhöhungen in der Regel nur durch Steigerung der Effizienz der vorhandenen Infrastrukturen erreicht werden, da kaum Möglichkeiten zum physischen Ausbau gegeben sind. Eine von Siemens in Auftrag gegebene Studie kommt zu dem Schluss, dass der wichtigste Wettbewerbsvorteil einer solchen Megacity in Zukunft das effiziente Funktionieren ihrer Verkehrs- und Energieinfrastrukturen sein wird [EIU07].

Gleichzeitig ist eine *stärkere Integration und Interoperabilität* der Infrastrukturen gefordert. Dies ist einerseits eine Folge aus der Liberalisierung der nationalen Märkte: Die zunehmende Verteilung von Verantwortlichkeiten auf unterschiedliche Organisationen (im Schienenverkehr z. B. zwischen Infrastrukturbetreibern und mehreren Bahnbetriebsgesellschaften) führt zu erhöhtem Koordinationsbedarf zwischen den Beteiligten [GW04, VDE03]. In Elektrizitätsnetzen wird diese Situation durch die zunehmende dezentrale Energieerzeugung zusätzlich verschärft [DG04]. Andererseits wird auch die grenzüberschreitende Interoperabilität als Kernvoraussetzung angesehen, um die Leistungsfähigkeit und Verfügbarkeit von Infrastrukturnetzen zu erhöhen und Kostensenkungspotentiale zu realisieren [SRAE07, SRRA02, TK07].

Schließlich stellt sich grundsätzlich das Problem, eine *höhere Kosteneffizienz* bei der Bereitstellung zu erzielen. Denn einerseits stehen Infrastrukturen untereinander in Konkurrenz, wie z. B. das Schienennetz zum Straßennetz. Andererseits erzeugt die zunehmende Marktliberalisierung auch spürbar mehr Wettbewerb zwischen Infrastrukturbetreibern. Dies zwingt sie dazu, die Bedürfnisse ihrer Kunden besser zu erfüllen als ihre Mitbewerber. Dazu gehören die Erhöhung der Ausfallsicherheit und Verfügbarkeit, aber vor allem auch die Senkung der Kosten durch effizientere Nutzung der vorhandenen Ressourcen [RAC+07, Maz05, SRAE07, SRRA07]. Beispielsweise machen im Schienenverkehr allein die Kosten für die Instandhaltung der Infrastruktur etwa die Hälfte der Gesamtkosten für den Betrieb der Infrastruktur aus [SRRA07]. Jegliche Reduzierung dieser Kosten würde also nicht nur zu besserer Kosteneffizienz führen, sondern stellt auch Mittel bereit, die in die Erhöhung der Attraktivität des Schienenverkehrs investiert werden können. Für die Betreiber von Stromnetzen spielt vor allem eine Rolle, dass Regulatoren die Netznutzungsentgelte immer weiter senken werden, um den Wettbewerb zu erhöhen. Dies wird den Wert der Stromnetze senken und die Betreiber zu immer effizienterer Bewirtschaftung ihrer Netze zwingen [RAC+07].

Insgesamt stehen aktuell also alle materiellen Infrastrukturen vor der Herausforderung, auf Basis der vorhandenen Anlagen höhere Kapazitäten zur Verfügung zu stellen, sich zunehmend zu integrieren und gleichzeitig ihre Kosteneffizienz zu erhöhen.

Strategien zur Bewältigung dieser Herausforderungen. Strategien zur Begegnung dieser Anforderungen werden auf europäischer Ebene in Form sogenannter *Strategic Research Agendas* entwickelt. Im Zentrum stehen dabei ein besserer Informationsaustausch zwischen den Beteiligten sowie optimierte Wartung und Instandhaltung [SRAE07, SRRA07, For06, Oll06, VDE03, LS05].

Durch einen *besseren Informationsaustausch* zwischen beteiligten Organisationen bezüglich ihrer individuellen Kapazitäten wird beispielsweise im Schienenverkehr erwartet, dass das gesamte Netz effizienter genutzt und dadurch die Gesamtkapazität erhöht werden kann [IGR05]. Mittels eines besseren Informationsaustauschs soll zudem die grenzüberschreitende Integration enger und damit Versorgungssicherheit und eine weitere Kapazitätserhöhung erreicht werden [SRAE07].

Mit Hilfe von *optimierter Instandhaltung* soll vor allem Kosteneffizienz erreicht werden. Zentraler Ansatz ist hier die *zustandsorientierte Instandhaltung* (engl. *condition-based maintenance*): Anstatt wie bisher nach festen Zeitplänen vorzugehen (engl. *preventive maintenance*), werden Inspektionen und Wartungsarbeiten in Zukunft bedarfsgesteuert und nach belastungs- oder zustandsabhängigen Zeitplänen durchgeführt. Dies geht einher mit einer zunehmenden Automatisierung der (Fern-)Überwachung des Infrastrukturzustandes und der Wartungsarbeiten, wodurch personalintensive Tätigkeiten reduziert und damit auch der aufgrund der demographischen Entwicklung zunehmende Mangel an qualifizierten Experten besser bewältigt werden kann [SRRA07]. Darauf aufbauend lässt sich mittels *vorausschauender Instandhaltung* (engl. *predictive maintenance*) zudem die Ausfallsicherheit einer Infrastruktur als auch ihre Kapazität weiter erhöhen: Wenn auf Basis aktueller Zustandsdaten und Hintergrundwissens zukünftige Fehler und Ausfälle abzusehen sind, können Wartungsarbeiten entsprechend geplant werden, um die Beeinträchtigung der Infrastrukturnutzung zu minimieren und Ausfälle zu verhindern [SRRA07].

Ansätze zur Umsetzung der Strategien. Grundvoraussetzung zur Umsetzung all dieser Strategien ist präzises Wissen über den aktuellen technischen Zustand der Infrastruktur. Dies stellt sich in der Realität als sehr schwieriges Problem dar: Zwar ist bereits heute eine Vielzahl von Fernüberwachungssystemen im Einsatz, die Unmengen an Daten produzieren und von unterschiedlichen Organisationen betrieben werden. Diese Daten beschreiben jedoch isolierte Phänomene und werden in der Regel lokal und unabhängig voneinander ausgewertet. Deshalb ist es heute nicht möglich, auf Basis der vorhandenen Daten einen aktuellen und komplexen Gesamtzustand abzuleiten und für die verschiedenen Komponenten der Infrastruktur anzugeben. Die Situation lässt sich mit einem Zitat von John Naisbitt aus seinem Bestseller *Megatrends* von 1984 pointiert auf den Punkt bringen: „Wir ertrinken in Informationen, aber hungern nach Wissen“ [Nai84].

Mehrere Forschungsprojekte und Initiativen versuchen zur Zeit, unter Schlagwörtern wie *Intelligent Infrastructures*, *Smart Infrastructures* und *Smart Grids* Lösungsansätze für diese Problematik zu entwickeln.

So hat das britische Foresight-Projekt[1] *Intelligent Infrastructure Systems* mit dem Schwerpunkt Verkehrsinfrastrukturen mögliche Beiträge von Wissenschaft und Technik untersucht. Als einer der drei Haupttreiber für die nächsten 50 Jahre wird dabei – neben Energie und Rohstoffen – die Umgebungsintelligenz (*Ambient Intelligence*) gesehen,

[1] Die britische Regierung betreibt mit dem sogenannten Foresight-Programm wissenschaftliche Zukunftsforschung als Grundlage für politische Entscheidungen. Informationen zum Foresight-Projekt Intelligent Infrastructure Systems (2004-2006) findet man unter http://www.foresight.gov.uk/Previous_Projects/Intelligent_Infrastructure_Systems/.

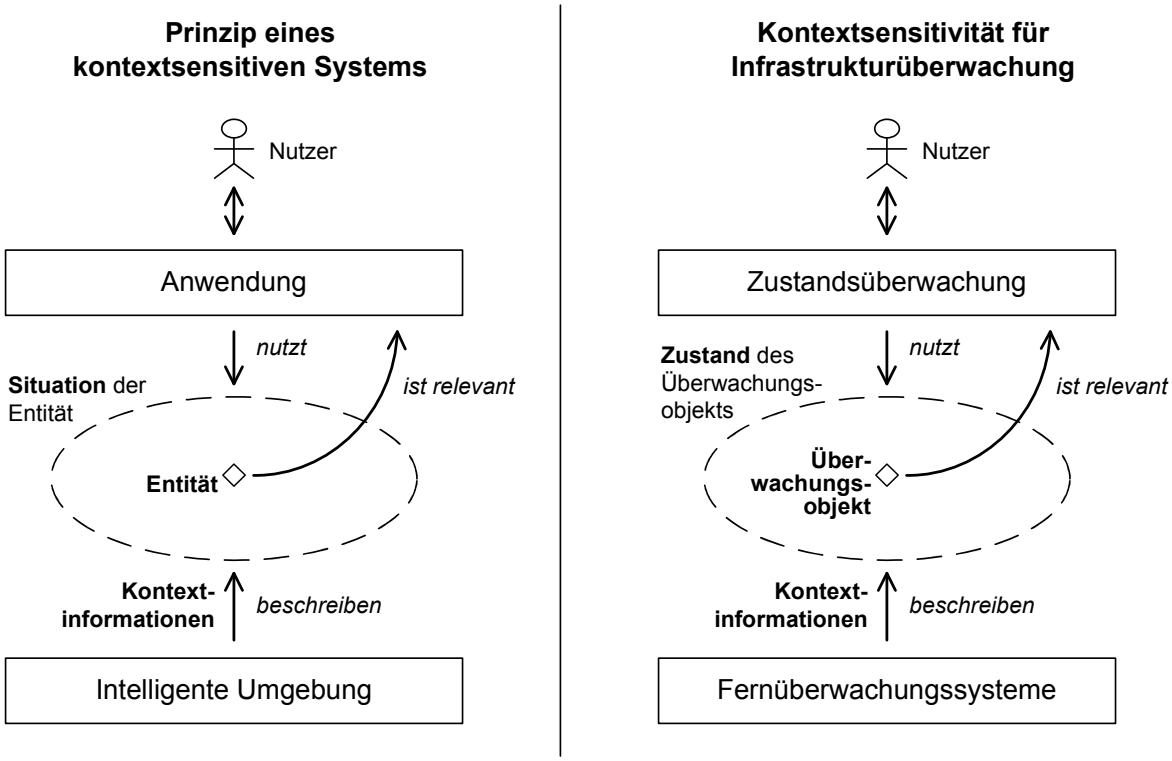

Abbildung 1.1.: Illustration des Prinzips kontextsensitiver Systeme (links) und seine Übertragung auf die Zustandsüberwachung für Infrastrukturnetze (rechts).

also die Durchdringung der realen Welt mit vernetzten Sensoren, Aktuatoren und Prozessoren. Auf dem Gebiet der Schienennetze beschäftigt sich das europäische Projekt *InteGRail* (*Intelligent Integration of Railway Systems*) mit europaweiter Interoperabilität, um dadurch präziseres Zustandswissen zu erreichen [IGR05]. Auf dem Gebiet der Elektrizitätsnetze werden in weiteren europäischen Projekten IT-Systeme vorgeschlagen und entwickelt, um Netze für dezentrale Energieerzeugung aufzurüsten [DG04] und Entscheidungsunterstützung für Netzbetreiber anzubieten [SLW+07]. In den USA wurde die *GridWise*-Initiative gegründet, um unter dem Stichwort *Smart Energy* Lösungen für höhere Zuverlässigkeit und Effizienz von Stromnetzen voranzutreiben [GW04, Maz05].

Ansatz dieser Arbeit. Eine Analyse der Anforderungen an die Zustandsüberwachung für Infrastrukturnetze offenbart die Verwandtschaft zu *kontextsensitiven Systemen*, wie sie bei Anwendungen der *Ambient Intelligence* und des *Ubiquitous Computing* eingesetzt werden. Das Prinzip kontextsensitiver Systeme nach Dey [Dey01] ist in Abbildung 1.1 auf der linken Seite illustriert: Eine kontextsensitive Anwendung unterstützt den Nutzer bei der Bewältigung einer Aufgabe, indem sie Kontextinformationen auswertet, um ihn mit hilfreichen Informationen und Diensten zu versorgen. Diese Kontextinformationen beziehen sich auf *Entitäten*, die für die Aufgabe relevant sind. Entitäten umfassen Personen, Orte sowie Dinge. *Kontextinformationen* (auch: *Kontext*) sind definiert als beliebige Informationen, die die Situation einer solchen Entität beschreiben. Sie werden

typischerweise von Sensoren erfasst, wie sie in intelligenten Umgebungen (engl. *smart environments*) eingesetzt werden.

Dieses Prinzip lässt sich wie folgt auf die Zustandsüberwachung bei Infrastrukturnetzen übertragen (siehe rechte Seite von Abbildung 1.1): Hier stellt die Infrastrukturüberwachung die Anwendung dar, die Kontextinformationen auswertet, um den Nutzer mit hilfreichen Informationen zu versorgen. Die relevanten *Entitäten* sind in diesem Fall die Überwachungsobjekte. Abhängig von der Anwendung können dies Infrastrukturkomponenten unterschiedlicher Granularität sein – im Schienenverkehr beispielweise von der einzelnen Achse bis zum gesamten Zug. Die kontextsensitive Zustandsüberwachung benötigt *Kontextinformationen* zu diesen Überwachungsobjekten, wozu alle Informationen zählen, die ihren Zustand beschreiben. Diese werden in Infrastrukturnetzen typischerweise ebenfalls von Sensoren geliefert, die in Fernüberwachungssystemen eingesetzt werden. Auch andere relevante Informationen wie Umwelt- und Wetterdaten zählen dazu.

Während die Problemstellung bei der Zustandsüberwachung also der Problemstellung bei kontextsensitiven Systemen entspricht, werden in beiden Domänen jedoch *unterschiedliche* Ansätze zu ihrer Lösung verfolgt: Traditionelle Systeme zur Zustandsüberwachung für Infrastrukturnetze betrachten die Daten der verschiedenen Fernüberwachungssysteme isoliert voneinander. Werden in Einzelfällen ausgewählte Überwachungssysteme integriert, so beruht diese Integration auf Ad-hoc-Verfahren, die jedes Mal neu entwickelt werden müssen. Kontextsensitive Systeme verfolgen hingegen zunehmend generische Ansätze, die auf formaler Wissensrepräsentation und -verarbeitung basieren. Hier wird differenziert zwischen dem anwendungsabhängigen Wissensmodell, das die Semantik der Kontextinformationen formalisiert, und der anwendungsunabhängigen Verarbeitungslogik. Dadurch wird die Integration von Kontextinformationen und die Erweiterung um zusätzliche Kontextinformationen stark vereinfacht.

Ausgangsidee dieser Arbeit ist deshalb, den wissensbasierten Ansatz kontextsensitiver Systeme auf die Zustandsüberwachung bei Infrastrukturnetzen zu übertragen. Weil Interoperabilität, die Berücksichtigung komplexer Zusammenhänge und Wartbarkeit hier sogar stärker im Vordergrund stehen, ist dieser Ansatz dafür besonders vielversprechend.

Darüber hinaus wurden in den letzten Jahren wichtige Fortschritte auf dem Gebiet der formalen Wissensrepräsentation und -verarbeitung (engl. *knowledge representation and reasoning*) erzielt, die diesem Ansatz zugute kommen. Erste Grundlagen wurden unter anderem von Minsky mit seinem Frame-Ansatz bereits im Jahr 1975 gelegt [Min75]. Schnell stellte sich jedoch heraus, dass eine höhere Ausdrucksstärke der Repräsentationssprache zwangsläufig mit einer höheren Berechnungskomplexität der automatisierten Verarbeitung einher geht. Das ist einer der Gründe, weshalb diese Formalismen lange Zeit trotz ihrer vielversprechenden Eigenschaften nur beschränkt praktisch einsetzbar waren. Angefangen mit der Entwicklung erster Beschreibungslogiken (engl. *Description Logics,* DL) [BCM+03] begann sich die Situation zu ändern: Nun wurde die Abhängigkeit zwischen Ausdrucksstärke und Berechnungskomplexität gezielt untersucht und Repräsentationssprachen mit den gewünschten Eigenschaften wurden konstruiert. Beschreibungslogische Wissensmodelle werden auch als (*formale*) *Ontologien* bezeichnet.

Einen enormen Schub erfuhr diese Art der Wissensrepräsentation durch die *Seman-*

tic Web-Initiative des *World Wide Web Consortiums* (W3C) [SW08]: Die Vision des Semantic Web besteht darin, die Semantik von Webseiten und -diensten formal zu repräsentieren, um sie für Maschinen zugänglich und damit automatisiert auswertbar zu machen. Dadurch soll das Finden relevanter Informationen vereinfacht und vielfältige Verknüpfungen ermöglicht werden [BLHL01]. Konkret wurde dafür im Jahr 2004 die *Web Ontology Language* (OWL) auf Basis einer bestimmten Beschreibungslogik spezifiziert und als W3C-Standard veröffentlicht. Die Verfügbarkeit einer Ontologiebeschreibungssprache als Standard einer einflussreichen Organisation – wie des W3C – hat vielfältige Synergien ermöglicht: Sie hat sich nicht nur im klassischen Web-Umfeld etabliert, sondern wurde auch für andere Anwendungen der Wissensrepräsentation und -verarbeitung übernommen. Dank dieses Standards entsteht zur Zeit eine Vielzahl von Entwicklungswerkzeugen (Editoren) und Software-Komponenten zur Verarbeitung dieser Daten (Reasoner, Triple Stores), die für die Realisierung wissensbasierter Systeme auf Basis von OWL genutzt werden können. Schließlich steigt auch die Zahl der verfügbaren, qualitativ hochwertigen Ontologien, die sich zur Wiederverwendung eignen.

Aus diesen Gründen verfolgt diese Arbeit für die Zustandsüberwachung bei Infrastrukturnetzen also einen wissensbasierten Ansatz, wie er sich bei den eng verwandten kontextsensitiven Systemen bereits bewährt hat. Begünstigt wird die praktische Umsetzbarkeit dieses Ansatzes zusätzlich von aktuellen Fortschritten bei der formalen Wissensrepräsentation und -verarbeitung im Umfeld des *Semantic Web*. Insgesamt sollen mit diesem Ansatz Grundlagen geschaffen werden, mit denen präzises Wissen über den aktuellen Zustand von Infrastrukturkomponenten gewonnen werden kann. Dies ist eine wichtige Voraussetzung, um die beschriebenen Herausforderungen an Infrastrukturnetze der Zukunft bewältigen zu können.

1.1. Aufgabenstellung und Ziele

Ausgangsidee für diese Arbeit ist das Verständnis der Zustandsüberwachung für Infrastrukturnetze als kontextsensitives System. Sowohl Daten von Überwachungssystemen als auch alle anderen Daten, die zur Zustandserkennung relevant sind, werden einheitlich als Kontextinformationen angesehen. Ziel der Arbeit ist es, auf Basis von *Semantic Web*-Technologien allgemeingültige Verfahren zur Integration und Auswertung von Kontextinformationen zu konzipieren und umzusetzen, mit denen komplexe Zustände von Infrastrukturkomponenten erkannt werden können. Dabei muss den besonderen Anforderungen der Zustandsüberwachung für Infrastrukturnetze Rechnung getragen werden.

Dazu werden folgende Fragestellungen bearbeitet und entsprechende Lösungsbeiträge entwickelt:

- Welche Eigenschaften besitzen Kontextinformationen und Zustände in Infrastrukturnetzen, und wie können diese geeignet modelliert werden? Welchen Anforderungen müssen diese Modelle zum Zweck der Infrastrukturüberwachung genügen, und wie kann dies sichergestellt werden?

- Welche Besonderheiten von Infrastrukturnetzen müssen bei der automatisierten

Auswertung der im Modell formalisierten Semantik berücksichtigt werden? Welche Anfragen müssen unterstützt werden, und mit welchen Verfahren kann dies effizient erreicht werden?

- Wie lassen sich die entwickelten Verfahren in der Praxis umsetzen? Wie sehen Systemarchitekturen aus, die die Anforderungen der Infrastrukturüberwachung erfüllen, und welche Komponenten und Schnittstellen sind dafür erforderlich?

Im Zentrum dieser Arbeit steht somit ein generisches Konzept zur Formalisierung der Semantik von Kontextinformationen und Zuständen in Infrastrukturnetzen sowie Verfahren zu deren Auswertung. Dies schließt die Entwicklung einer geeigneten generischen Systemarchitektur und ausgewählte Umsetzungen sowie die Anwendung auf bestimmte Infrastrukturdomänen ein. Dabei soll auf bestehenden Standards der *Semantic Web*-Initiative des W3C aufgebaut werden, um die Umsetzbarkeit und Zukunftsträchtigkeit der Konzepte zu gewährleisten.

Nicht Teil der Aufgabenstellung ist es, ein integriertes Überwachungssystem für eine bestimmte Infrastrukturdomäne vollständig zu realisieren. Dazu müssten konkrete Überwachungssysteme und ihre Schnittstellen, Kommunikationstechnologien und andere Aspekte, wie z. B. Informationssicherheit, im Detail untersucht werden. Zur Modellierung sollen aus den genannten Gründen die bestehenden *Semantic Web*-Standards des W3C verwendet werden. Die Evaluierung darüber hinaus gehender Wissensrepräsentationssprachen bezüglich ihrer Modellierungseigenschaften wäre Thema einer eigenständigen Arbeit.

1.2. Vorgehensweise und Gliederung

Zur Erreichung der oben erläuterten Ziele wurde folgende Vorgehensweise gewählt. Diese spiegelt sich auch in der Gliederung der Arbeit wieder.

Zunächst wird das Gebiet der Zustandsüberwachung für Infrastrukturnetze anhand der repräsentativen Beispiele Schienen- und Stromnetz analysiert (Kapitel 2). Es zeigt sich, dass die Kernanforderungen übereinstimmen und sich damit generische Anforderungen an die Zustandsüberwachung für Infrastrukturnetze ableiten lassen.

Vor dem Hintergrund dieser Anforderungen wird das Gebiet der formalen Wissensrepräsentation eingeführt (Kapitel 3). Dabei ergibt sich, dass die deklarative Wissensrepräsentation auf Basis von Beschreibungslogiken aufgrund ihrer hohen Ausdrucksstärke im Verhältnis zur Verarbeitungskomplexität sowie der attraktiven Wartbarkeits- und Erweiterbarkeitseigenschaften einen vielversprechenden Formalismus für die Zustandsüberwachung darstellt.

Bei der Erstellung von Wissensmodellen für die Infrastrukturüberwachung (Kapitel 4) führen klassische Methodiken jedoch nicht zum Ziel, da das zu formalisierende Wissen über mehrere Domänenexperten verteilt ist und diese in der Regel nicht über die nötige Modellierungkompetenz verfügen. Um Domänenexperten dennoch in die Lage zu versetzen, qualitativ hochwertige Teilmodelle eines konsistenten Gesamtmodells zu erstellen, konzipiert diese Arbeit Entwurfsmuster für die Modellierung von Kontextinformationen und Infrastrukturzuständen. Dazu wird auf etablierten und bewährten Standards für die Zustandsüberwachung technischer Systeme aufgebaut.

Um die im Modell formalisierte Semantik für automatisiertes Schlussfolgern (*Reasoning*) nutzen zu können, müssen jedoch die Besonderheiten von Infrastrukturnetzen berücksichtigt werden: Kontextinformationen liegen – aufgrund der geographischen Ausdehnung, der vielen beteiligten Organisationen und der heterogenen Datenquellen – in der Regel verteilt vor. Da eine Zentralisierung nicht praktikabel ist, können bisherige Reasoning-Ansätze nicht eingesetzt werden. Als neuen Ansatz schlägt die Arbeit ein Framework auf Basis verteilter Reasoning-Knoten vor, das eine Ausgangsanfrage mittels generierter Unteranfragen an andere Reasoning-Knoten vollständig und korrekt beantwortet. Dazu werden unterschiedliche Arten der *Verteilheit* von Kontextinformationen identifiziert und entsprechende Strategien zur Generierung der Unteranfragen entwickelt und formal verifiziert (Kapitel 5). Diese werden auf Basis einer prototypischen Implementierung zusätzlich experimentell evaluiert.

Da topologiebezogene Anfragen, wie die Suche nach kürzesten Pfaden, mit der beschreibungslogischen Grundlage weder formuliert noch ausgewertet werden können, muss ein eigener Ansatz entwickelt werden, um Anfragen mit Bezug zur *Topologie* des Infrastrukturnetzes beantworten zu können. Dazu werden Konzepte zur Modellierung des Infrastrukturnetzes und zur netzbezogenen Integration von Kontextinformationen entwickelt. Darauf aufbauend wird ein neuartiger Ansatz vorgestellt, der bestehende Graphenalgorithmen zur Lösung topologiebezogener Anfragen mit beschreibungslogischem Reasoning kombiniert (Kapitel 6). Er wird am Beispiel von Optimale-Pfade-, Nächste-Nachbarn- und Bereichsanfragen demonstriert, und seine Implementierung wird vorgestellt.

Zur praktischen Umsetzung dieser entwickelten Konzepte zur Zustandsüberwachung für Infrastrukturnetze wird eine skalierbare und erweiterbare Systemarchitektur benötigt. Dazu spezifiziert diese Arbeit funktionale Schichten mit ihren Schnittstellen und ordnet ihnen die entwickelten Verfahren in Form von Komponenten zu (Kapitel 7). Auf Basis dieser Architektur wird gezeigt, wie bei Bedarf weitere spezialisierte Verfahren analog zu den Verfahren für topologiebezogene Anfragen in das Gesamtsystem eingebunden werden können. Dadurch ist es möglich, die einmal formalisierte Semantik für unterschiedliche Zwecke mehrfach zu nutzen. Am Beispiel des europäischen Schienennetzes wird eine Umsetzung der vorgeschlagenen Systemarchitektur mit europaweit verteilten Komponenten vorgestellt.

Im Anschluss an eine Zusammenfassung der Ergebnisse dieser Arbeit (Kapitel 8) wird ein Ausblick auf weitere Forschungsfragestellungen in den behandelten Gebieten gegeben.

2. Zustandsüberwachung für Infrastrukturnetze

Infrastrukturen erfüllen grundlegende Bedürfnisse von Staat und Gesellschaft. Der Sammelbegriff *Infrastruktur* ist vom lateinischen *infra* (unten, unterhalb) abgeleitet. Er bezeichnet langlebige Grund- und Versorgungseinrichtungen personeller, technischer oder institutioneller Art, welche das Funktionieren einer arbeitsteiligen Volkswirtschaft garantieren. In den letzten Jahren gewann der Begriff *kritische Infrastrukturen* außerdem immer mehr an Bedeutung. Er bezeichnet „Organisationen und Einrichtungen mit wichtiger Bedeutung für das staatliche Gemeinwesen, bei deren Ausfall oder Beeinträchtigung nachhaltig wirkende Versorgungsengpässe, erhebliche Störungen der öffentlichen Sicherheit oder andere dramatische Folgen eintreten würden." [BSI04] Der Infrastrukturbegriff umfasst hier also z. B. auch Behörden, Verwaltung und Justiz sowie das Finanz-, Geld- und Versicherungswesen. Im Rahmen dieser Arbeit werden speziell *materielle* oder *Netz-Infrastrukturen* betrachtet. Dazu zählen Transport und Verkehr (Schiene, Straße, Luft und See), Energie (Strom, Gas) und Wasser (Versorgung, Entsorgung).

Im vorangegangenen Kapitel wurden bereits zukünftige Herausforderungen für diese Infrastrukturen und Strategien zu ihrer Erreichung dargestellt. Diesen Ansätzen liegt immer präzises Wissen über den aktuellen Zustand von Infrastrukturkomponenten zugrunde. Dieses Kapitel analysiert nun auf Basis konkreter Infrastrukturnetz-Domänen, welche Herausforderungen sich bei der Realisierung einer Zustandsüberwachung für Infrastrukturnetze stellen. Dazu werden exemplarisch die charakteristischen Eigenschaften von Schienen- und Stromnetzen in den Abschnitten 2.1 und 2.2 kurz vorgestellt. Abschnitt 2.3 leitet daraus generische Anforderungen an die Zustandsüberwachung für Infrastrukturnetze ab, die die Leitlinien für diese Arbeit bilden.

2.1. Schienennetze

Die erste öffentliche Bahnlinie wurde 1825 in England in Betrieb genommen. Die Eisenbahn entwickelte sich innerhalb weniger Jahrzehnte zu einem vernetzten Verkehrsmittel, das die Reisezeiten in Europa und Nordamerika drastisch verkürzte. Heutzutage sind Schienennetze weltweit verbreitet und übernehmen wichtige Funktionen sowohl beim Passagier- als auch beim Frachtverkehr.

2.1.1. Eigenschaften

Zur Charakterisierung der Zustandserkennung in Schienennetzen werden folgende Aspekte beschrieben: Struktur, beteiligte Organisationen, vorhandene Quellen für

Kontextinformationen, relevante Infrastrukturzustände und Lebensdauer. Da in den Beispielen in späteren Kapiteln die englischen Fachbegriffe verwendet werden, werden diese hier ebenfalls angegeben.

Struktur. Im Schienenverkehr unterscheidet man grundsätzlich zwischen dem *Rollmaterial* (engl. *rolling stock*) und der ortsfesten *Infrastruktur*. Das Rollmaterial umfasst alle Fahrzeuge, wie z. B. Lokomotiven, Triebwagen und Waggons. Im Gegensatz dazu bezeichnet Infrastruktur alle ortsfesten Eisenbahnanlagen, wie z. B. Eisenbahnstrecken und Bahnhöfe. Eine Eisenbahnstrecke ist die logische Verbindung von Orten mit einem Schienenweg, die sich physisch als *Trasse* (engl. *track*) manifestiert. Der Begriff (*Eisen-*)*Bahnlinie* (engl. *route*) hingegen bezeichnet den regelmäßig auf einer Strecke stattfindenden Verkehr. Eine Trasse kann mit einem oder mehreren *Gleisen* (engl. *road*), *Hochbauten*, *Weichen* und *Kreuzungen*, *Oberleitungen*, *Bahnhöfen* und *Haltepunkten* ausgestattet sein. Sie besitzt einen *Unterbau*, der lagestabil aufgebaut ist, und einen auf einer Gleisbettung aus Schotter liegenden *Oberbau* aus *Schienen* (engl. *rails*) und *Schwellen* (engl. *sleepers*).

Als Beispiel für die Größenordnungen einer Eisenbahninfrastruktur wird hier das Netz der Deutschen Bahn AG angeführt. Bei über 6600 Haltestellen umfasste es Ende 2006 eine Betriebslänge (Länge des fahrplanmäßig genutzten Gleisnetzes) von 34.019 km bei einer Gleislänge (Länge aller Gleisanlagen) von insgesamt 62.948 km. Dazu kommen 77.080 Kreuzungen und Weichen sowie 673 Tunnel und 24.915 Brücken [DB07].

Beteiligte Organisationen. Eine Vielzahl von Organisationen ist an Bau, Betrieb und Instandhaltung des Schienenverkehrs beteiligt und produziert damit Daten, die für die Zustandsüberwachung relevant sind. Trotz einer allgemeinen Tendenz zur Privatisierung des Schienenverkehrs in Europa gibt es dabei große nationale Unterschiede. Als Beispiel für eines der am weitesten privatisierten Systeme soll hier der britische Schienenverkehr genannt werden. Dort gibt es einerseits *(Bahn-)Betriebsgesellschaften* (oder *Eisenbahnverkehrsunternehmen*) für den Personenverkehr und den Güterverkehr. Diese sind jedoch nicht selbst Eigentümer des eingesetzten Rollmaterials, sondern leasen es von darauf spezialisierten *Leasinggesellschaften*. Wartung und Instandhaltung des Rollmaterials werden entweder ebenfalls von den Leasinggesellschaften oder direkt von den *Rollmaterial-Herstellern* durchgeführt. Die Infrastruktur (Schienen, Bahnhöfe, Betriebswerkstätten und Signale) wird hingegen von *Eisenbahninfrastrukturunternehmen* betrieben und instand gehalten (in Großbritannien z. B. von der halbstaatlichen *Network Rail*). Das führt dazu, dass Daten, die für die Zustandsüberwachung relevant sind, über unterschiedliche Organisationen verteilt sind.

Vorhandene Quellen für Kontextinformationen. Im Schienenverkehr wird unterschiedlichste Sensorik bereits heute und in zunehmendem Maße verwendet. Zum Beispiel sind im britischen Schienennetz zur Zeit mehr als 3000 Fernüberwachungssysteme im Einsatz [Oll06]. Prinzipiell kann zwischen Überwachungssystemen auf dem Zug und an der Strecke unterschieden werden.

Systeme am Zug erfassen z. B. den Zustand des Stromabnehmers, den Status von Türen, Klimaanlagen und Toiletten sowie Bremsen, die Achslagertemperatur und auch

den Radschlupf – bespielsweise aufgrund von Laub auf der Schiene. Sie werden aber auch eingesetzt, um Daten über die Infrastruktur zu erfassen. So misst der Messzug *New Measurement Train* (*NMT*) von Network Rail unter anderem die Gleisgeometrie (räumliche Anordnung der Schienen zueinander) mittels Laserscannern und das Schienenprofil (Abnutzung der Schienen) mittels Ultraschall [Oll06]. Andere Systeme erfassen zusätzlich die Welligkeit der Schienenoberfläche.

Überwachungssysteme an der Strecke messen einerseits Daten zum Zustand der Strecke wie z. B. Streckenüberflutungen, das Vorhandensein von Streustrom aufgrund von Isolations-Fehlstellen mittels Durchschlagspannungssensoren sowie den Zustand von Weichen und Weichenheizungen mittels induktiver Weichensensoren. Es gibt jedoch auch Systeme, die von der Strecke aus das Rollmaterial überwachen. Die wichtigsten sind *Achslastgeber* (engl. *Wheel Impact Load Detector,* WILD) und *Heißläuferortungsanlagen* (engl. *hot axle box detector,* HABD) [SPK06]. Achslastgeber erkennen, wenn ein Schienenfahrzeug aufgrund überlasteter Achsen oder unrunder Räder die Infrastruktur beschädigt (bis zum Bruch von Schienen) und deshalb aus dem Verkehr gezogen werden muss. Heißläuferortungsanlagen erkennen heißgelaufene Achsen und Achslager, bevor ein Zug auf offener Strecke liegen bleibt. Dazu werden z. B. Infrarotkameras eingesetzt. Die Messung der Achstemperatur ist ein Beispiel, bei dem dasselbe Phänomen sowohl von Sensorik an der Strecke als auch von Sensorik am Zug erfasst werden kann.

Relevante Infrastrukturzustände. Es gibt eine Vielzahl aussagekräftiger Zusammenhänge zwischen Zuständen des Rollmaterials und der Infrastruktur sowie zwischen Zuständen innerhalb der Infrastruktur.

Die Schnittstelle zwischen Rad und Schiene ist besonders wichtig, weil dort die meisten Kosten für die Wartung entstehen und das Risiko für Unfälle wie Entgleisungen am höchsten ist [Lag07]. Um den Zustand eines Radsatzes beurteilen zu können, sind mehrere Aspekte relevant: Neben der aktuellen Beladung des Wagens sowie der Rad- und Achslagertemperaturen, die von Heißläuferortungsanlagen ermittelt werden, müssen auch die Abnutzung der Bremsbeläge, eventueller Radschlupf sowie die Abflachung der Räder, die von Achslastgebern erfasst wird, berücksichtigt werden. Im Zusammenhang gesehen, kann dann beispielsweise differenziert werden zwischen einer schleifenden Bremse und abgenutzten Rädern. Dadurch kann die Instandsetzung zielgerichtet erfolgen, und Standzeiten können minimiert werden, weil Werkstätten auf die zu erledigenden Arbeiten vorbereitet sind. Zudem wird vorausschauende Instandsetzung möglich, wenn die einzelnen Überwachungssysteme für sich gesehen beispielsweise noch keinen Alarm auslösen (Achslast ist unter dem kritischen Wert von 350 kN), jedoch in der Kombinationen aufgrund mehrerer erhöhter Werte (Achslast von mehr als 200 kN *und* erhöhte Achslagertemperatur) darauf schließen lassen, dass sich demnächst ein Problem entwickeln wird.

Auch bei den Gleisen müssen mehrere Aspekte im Zusammenhang gesehen werden: Dazu gehören die Abweichung der räumlichen Anordnung der Schienen von der spezifizierten Gleisgeometrie und die Abriebprofile der einzelnen Schienen, aber auch die Abnutzung der Schwellen und der Zustand des Schotterbettes, welche teilweise durch menschliche Inspektion erfasst werden. Auch Umgebungsinformationen wie Niederschlagsmengen und Informationen über Baumbestand und dadurch Laubfall

auf die Schienen im Herbst sind entscheidend. Auf der Ebene der Trasse sind außerdem weitere Informationen wie die Abnutzung der Oberleitung und die Qualität der Stromversorgung relevant [IGR05].

Auch um Fehlinterpretationen von Daten zu verhindern, ist es wichtig, Meldungen von Überwachungssystemen zu integrieren. So lässt eine erhöhte Radtemperatur nicht eindeutig auf schleifende Bremsen schließen, sondern kann genauso gut durch entweichende Auspuffgase verursacht worden sein [Oll06].

Lebensdauer. Das Schienennetz ist eine sehr langlebige Transportinfrastruktur. Der Aufbau des Schienennetzes begann in Deutschland vor mehr als 150 Jahren. Vereinzelt sind Schienen schon viele Jahrzehnte lang im Einsatz. Bei Schienenfahrzeugen wird typischerweise von einer Lebensdauer von 40 Jahren ausgegangen. Das durchschnittliche Alter der Gleisanlagen in Deutschland lag 2006 bei 19,8 Jahren, das der Weichen bei 16,9 Jahren. Brücken sind durchschnittlich 53,1 Jahre, Bahnsteige durchschnittlich 47,6 Jahre alt [DB07].

2.1.2. Herausforderungen für die Zustandsüberwachung

Leistungskennzahlen (engl. *key performance indicators*) machen die Leistung des Schienenverkehrs quantifizierbar. Für den Bereich der Infrastruktur lassen sich diese klassifizieren in *Sicherheit*, *Verfügbarkeit*, *Effizienz*, *Kapazität* und *Qualität* [IGR05]. Die Leistungskennzahl für Kapazität wird dann beispielsweise weiter aufgeschlüsselt in *Geschwindigkeit*, *Achslast*, *Lichtraumprofil*, *Anzahl der Trassen*, *Energieversorgung* und *Umweltverschmutzung*. Die Zustandsüberwachung wird eingesetzt, um diese Leistungskennzahlen zu verbessern.

Für die Zustandsüberwachung bei Schienennetzen wurden dazu folgende Leitlinien identifiziert [Oll06]: Daten sollen häufiger und regelmäßiger erfasst werden. Die erfassten Daten sollen nicht isoliert voneinander in Datensilos gehalten, sondern müssen integriert und im Zusammenhang gesehen werden. Daten müssen so interpretiert werden, dass sie zu nützlichen Informationen für den jeweiligen Nutzer werden. Systeme zur Zustandsüberwachung benötigen zudem folgende Eigenschaften: Sie sollen eine offene und erweiterbare Architektur besitzen; sie sollen sowohl lokale als auch globale Informationen zur Verfügung stellen; der Zugriff auf alle Arten von Fernüberwachungssystemen soll einheitlich sein; sie sollen auf kommerziell verfügbaren Softwarekomponenten basieren; sie sollen zukunftsfähig und interoperabel sein; und sie sollen eine Standard-Softwarearchitektur verwenden. Zusammengefasst leiten sich daraus folgende Anforderungen für die Zustandsüberwachung im Schienenverkehr ab:

- *Integration von Kontextinformationen*: Um Infrastrukturzustände präzise beurteilen zu können, müssen Kontextinformationen im Zusammenhang gesehen werden, z. B. die Geometrie eines Gleises in Kombination mit den Profilen seiner Schienen. Auch Umweltbedingungen, die bisher nicht durch Überwachungssysteme erfasst werden, müssen berücksichtigt werden.

- *Semantische Interoperabilität*: Die zu integrierenden Kontextinformationen werden von unterschiedlichen Beteiligten erfasst, z. B. von Bahn- und Infrastrukturbe-

treibern. Um diese austauschen und integrieren zu können, ist Interoperabilität sowohl auf syntaktischer als auch auf semantischer Ebene erforderlich.

- *Menschenlesbare Dokumentation der Zusammenhänge*: Wie gezeigt gibt es sehr viele relevante Zusammenhänge. Jeder Domänenexperte kennt jedoch nur einen Ausschnitt des Gesamtsystems, z. B. die Domäne der Achslager. Zudem wird es immer wieder zu Erweiterungen und Korrekturen des Modells kommen. Deshalb müssen die modellierten Zusammenhänge leicht verständlich und nachvollziehbar dokumentiert werden.

- *Umgang mit unvollständigen Daten*: Da Kontextinformationen von einer Vielzahl von Systemen produziert und diese von unterschiedlichen Organisationen betrieben werden, kann nicht davon ausgegangen werden, dass alle Daten zu jeder Zeit zur Verfügung stehen. Beispielsweise können Daten, die auf dem Zug gesammelt werden, nicht zugreifbar sein. Stattdessen muss man bei der Integration damit umgehen können, dass die verfügbaren Daten unvollständig sind.

- *Zukunftsfähigkeit und Erweiterbarkeit*: Die lange Lebensdauer von Schieneninfrastrukturen und die hohen Kapitalinvestitionen zeigen, dass ein neuer Ansatz zur Zustandsüberwachung zukunftsfähig sein muss: Er muss an zukünftige Anforderungen anpassbar sein und auch für Anforderungen genutzt werden können, die bisher noch gar nicht bekannt sind. Zudem müssen dieselben Informationen von unterschiedlichen Anwendungen für unterschiedliche Zwecke genutzt werden können.

2.1.3. Stand der Technik bei der Zustandsüberwachung

Eine Vielzahl von (Fern-)Überwachungssystemen ist auch heute schon im Einsatz. Allein für die gleisseitige Überwachung sind mehrere Systeme kommerziell verfügbar [Lag07, ST98]: So werden Schleif- und Entgleisungsdetektoren eingesetzt, um herunterhängende Objekte und Entgleisungen zu erkennen. Zur Erkennung überhitzter Achslager gibt es Heißläuferortungsanlagen auf Basis von Infrarotsensoren und -kameras [SPK06]. Außerdem sind Radschlupfdetektoren im Einsatz, die erkennen, wenn Räder – aufgrund mechanischer oder menschlicher Fehler – durchdrehen und deshalb die Gefahr von Entgleisungen besteht. Diese Systeme sind entweder mechanisch oder auf Basis von Kameras realisiert. Akustische Sensoren erkennen bei Achslagern ungewöhnliche Vibrationen als Ergänzung zu traditionellen Heißläuferortungsanlagen. Beschleunigungssensoren messen die horizontalen und vertikalen Kräfte, die das Rollmaterial auf die Infrastruktur ausübt. Achslastgeber werden eingesetzt, um Schäden an der Radoberfläche zu erkennen. Dazu werden die dynamischen vertikalen Kräfte gemessen, die ein unrundes Rad auf die Schiene ausübt. Andere Realisierungen verwenden Laser und Kameras. In Großbritannien wurde von Network Rail dazu seit dem Jahr 2000 an etwa 30 Stellen das *WheelChex*-System eingeführt (siehe Abbildung 2.1), welches zu einer wesentlichen Schadensverringerung geführt hat.

All diese Systeme sind jedoch komplett eigenständig und in sich geschlossen. Sie erfüllen zwar ihren spezifischen Zweck, es wird jedoch in der Regel kein Nutzen aus der Vernetzung und integrierten Betrachtung der gelieferten Daten gezogen [Oll06].

Abbildung 2.1.: Standorte der WheelChex-Achslastgeber in Großbritannien gekennzeichnet durch schwarze Punkte (siehe [NR07] für das Streckennetz auf Seite 15 und die Liste der Standorte in Anhang D).

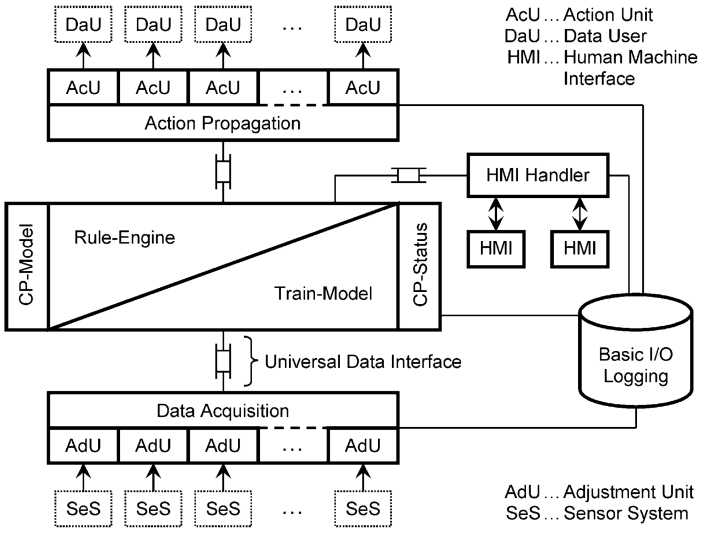

Abbildung 2.2.: Datenkonzentrator des Checkpoint-Systems [SM06].

Darüber hinaus bieten die großen Bahntechnik-Hersteller IT-Systeme für integrierte Schienenverkehrs-Leitstände an[1]. Diese realisieren jedoch lediglich einheitliche Plattformen zur Kommunikation mit den eigenständigen Überwachungssystemen (desselben Herstellers); eine semantische Integration und Interpretation der gewonnenen Daten findet nicht statt [MRSS05].

In der Forschung gibt es erste Ansätze zur inhaltlichen Integration der Überwachungsdaten: Steets und Tse [ST98] berichten von der integrierten Aufbereitung der Messergebnisse von Achslastgebern, Radschlupfdetektoren und Festbremsortungsanlagen. Dieses System ist jedoch sehr starr und auf eine bestimmte Menge an Sensoren zugeschnitten. Maly et al. haben sich im Rahmen des österreichischen Forschungsprojekts *Checkpoint* intensiv mit Zugüberwachung beschäftigt und als Sensorsysteme dynamische Gleiswaagen, Heißläufer- und Festbremsortungsanlagen sowie die Lichtraumprofilüberwachung integriert [SM06, MS05, MRSS05, MRS04]. Um die Hinzunahme weiterer Sensorquellen zu ermöglichen, wurde ein *Universal Data Interface* entworfen, zu dem jedoch keine Details veröffentlich sind (siehe Abbildung 2.2). Verarbeitet werden die gesammelten Daten mit einer Regelmaschine (*rule engine*), die für die vorgesehenen Sensorsysteme bestimmte, vordefinierte Auswertungen vornimmt. Daran offenbaren sich auch die Beschränkungen dieses Ansatzes: Es gibt keine formale Semantik, die maschinenverarbeitbar wäre und dadurch flexible Auswertungsmöglichkeiten erlauben würde. Stattdessen muss Domänenwissen in prozeduraler Form als Regeln erfasst werden, was die Nachvollziehbarkeit und Wartbarkeit erschwert.

Vor dem Hintergrund der vorher identifizierten Anforderungen zeigt sich, dass bei Schienennetzen bisher kein Ansatz zur (semantisch) integrierten Zustandsüberwachung existiert. Diese Problematik ist Gegenstand aktueller Forschungsprojekte [IGR05]. Zwar gibt es einzelne Beispiele, die Daten des einen Systems mit Daten des anderen

[1] Siemens RailCom Manager: http://www.transportation.siemens.com
Alstom ICONIS Integrated Control Center: http://www.transport.alstom.com
Bombardier Rail Control Solutions: http://www.bombardier.com

Systems in Beziehungen setzen; diese bieten jedoch nicht die nötige Erweiterbarkeit und Wartbarkeit.

2.2. Stromnetze

Die erste größere Stromleitung wurde 1882 von Oskar von Miller gebaut. Sie führte 57 Kilometer vom bayerischen Miesbach nach München und transportierte Gleichstrom. Bald ging man über zu Wechselstrom, der sich auch über weite Strecken mit relativ geringen Verlusten übertragen lässt. 1891 wurden in Köln das erste große Wechselstromkraftwerk und im selben Jahr eine 176 Kilometer lange Wechselstromleitung vom schwäbischen Lauffen nach Frankfurt am Main in Betrieb genommen. Heutzutage ist die zuverlässige Versorgung mit Strom für die moderne Gesellschaft unentbehrlich geworden.

2.2.1. Eigenschaften

Dieser Abschnitt führt ebenso wie bei Schienennetzen anhand von Struktur, beteiligten Organisationen, vorhandenen Quellen für Kontextinformationen, relevanten Infrastrukturzuständen und Lebensdauer kurz in die Zustandsüberwachung bei Stromnetzen ein.

Struktur. Eine Stromversorgungsinfrastruktur setzt sich aus vielen unterschiedlichen Stromnetzen zusammen. Dabei werden die Netztypen nach der Spannung unterschieden, mit der sie Strom übertragen [SM02, OSHA07]:

- *Höchstspannungsnetze* werden (in Deutschland) mit 230 kV oder 400 kV betrieben. Sie sind Übertragungsnetze, die die von Kraftwerken eingespeiste Energie landesweit an Transformatoren nahe der Verbrauchsschwerpunkte transportieren. Über Kuppelleitungen sind sie an das internationale Verbundnetz angeschlossen.
- *Hochspannungsnetze* werden mit 50 kV bis 150 kV betrieben. Sie sorgen für die Grobverteilung. Die Leitungen führen in Ballungszentren und zu großen Industriebetrieben.
- *Mittelspannungsnetze* werden mit 6 kV bis 30 kV betrieben. Sie verteilen den Strom an die Transformatorstationen des Niederspannungsnetzes. Stadtwerke speisen ihren Strom in dieses Netz.
- *Niederspannungsnetze* werden mit 230 V bis 400 V betrieben, in der Industrie auch mit 500 V oder 690 V. Sie sind für die Feinverteilung zuständig, um Haushalte, Industrie und Verwaltungen zu versorgen.

Höchstspannungsnetze werden auch *Übertragungsnetze* genannt, während Hoch-, Mittel- und Niederspannungsnetze als *Verteilnetze* bezeichnet werden.

Die Verbindung von Stromnetzen unterschiedlicher Spannungsebenen erfolgt über *Transformatoren*, die in *Umspannanlagen* installiert sind. Die Steuerung des Stromflusses durch ein Stromnetz und zu Stromnetzen der gleichen Spannungsebene erfolgt über

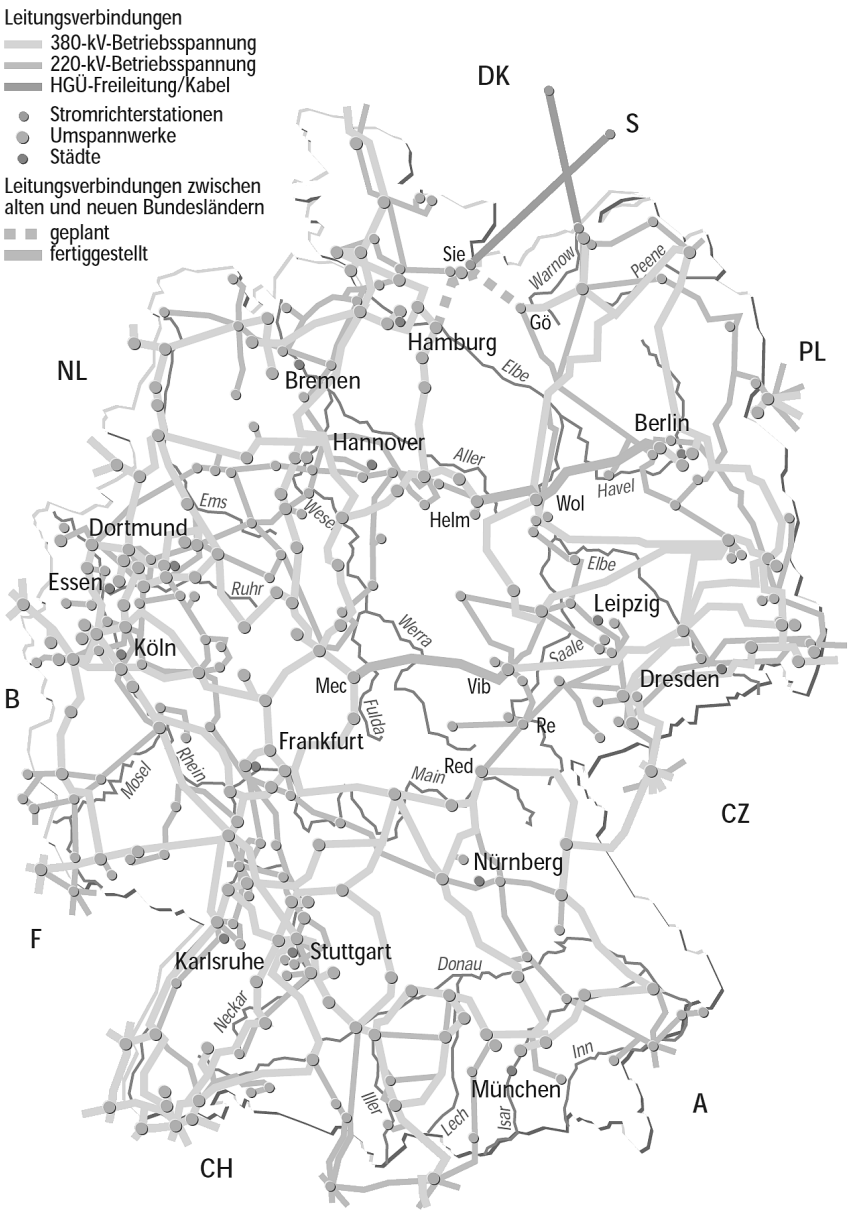

Abbildung 2.3.: Stromübertragungsnetz in Deutschland (Stand: 1. Januar 1996, nach [Sch05]).

Schaltanlagen. Die Gesamtlänge aller Leitungen beträgt in Deutschland zur Zeit etwa 1,67 Millionen Kilometer. Es sind etwa 550.000 Transformatoren installiert (siehe Abbildung 2.3 für eine Illustration des Übertragungsnetzes in Deutschland).

Beteiligte Organisationen. Prinzipiell unterscheidet man zwischen *Energieerzeugern*, *Übertragungsnetz-* sowie *Verteilnetzbetreibern*. In Europa gibt es bei den Höchstspannungsnetzen ein Verbundsystem. Es wird in Deutschland von den vier Übertragungsnetzbetreibern *EnBW*, *E.ON*, *RWE* und *Vattenfall* betrieben. Neben diesen vier gibt es noch rund 900 Verteilnetzbetreiber, die auf regionaler Ebene agieren und den Strom zu den Endverbrauchern liefern. Zusätzlich betreibt die Deutsche Bahn deutschlandweit

ein Hochspannungsnetz zur Energieversorgung elektrischer Bahnen.

Vorhandene Quellen für Kontextinformationen. Leitungen und vor allem Umspannwerke sind mit umfangreicher Sensorik ausgestattet: Bei Leitungen werden zum Beispiel die Leiterseiltemperatur gemessen und Frequenzgänge, Strom- und Spannungskurven erfasst. In Umspannwerken werden bei Transformatoren mittels Drucksensoren die Dichte des Dämmgases, mittels Temperatursensoren die Öl- und Transformatortemperatur sowie mittels Stromsensoren die Last gemessen. Es werden Daten über die Schaltgeschwindigkeit von Unterbrechern sowie ihr aktueller Schaltzustand erfasst. Bei Stromzuleitungen werden unter anderem die Blindleistung sowie Last- und Spannungsprofile gemessen.

Relevante Infrastrukturzustände. Zum optimalen Betrieb und zur Instandhaltung eines Stromnetzes werden Zustandsinformationen benötigt, die nur durch Integration mehrerer Kontextinformationen abgeleitet werden können.

Zum Beispiel ist es zur Beurteilung des Zustands eines Transformators wichtig, sowohl seine Last und aktuelle Temperatur (inklusive Zustand des Kühlsystems und der Umgebungstemperatur) als auch die Unversehrtheit seiner Abdichtung und die produzierte Menge an brennbaren Gasen zu kennen. Dadurch können typische Fehler wie Ölverlust, Ausfall der Zusatzkühlung sowie Fehler am Stufenschalter oder Buchholzschutz erkannt werden. Für Leistungs- oder Trennschalter sind z. B. die Dichte des Dämmgases, Auslösungs- und Schließzeiten, Temperatur und Zustand des Heizelements ausschlaggebend. Auch wenn einzelne Überwachungssysteme noch keinen kritischen Zustand erkannt haben und deshalb keinen Alarm schlagen, kann aus der Kombination mehrerer erhöhter Werte ein zukünftiger Ausfall rechtzeitig vorhergesehen werden [Kre03a].

Bei Stromzuleitungen müssen Blindleistung, Lastprofil und Spannungsprofil betrachtet werden, um Überlastungen und Abnutzung zu beurteilen. Dabei müssen auch Umgebungsbedingungen wie Blitzschlag und Wetter berücksichtigt werden. Denn fällt eine Leitung im Zusammenhang mit einem Blitzschlag aus, so kann sie nach einiger Zeit problemlos wieder zugeschaltet werden, ohne vor Ort eine Wartung durchführen zu müssen. Dies gilt jedoch nur, sofern es sich um eine Überland- und keine Erdleitung handelt – ein Beispiel, wo unvollständige Informationen bei unvorsichtiger Interpretation zu falschen Schlussfolgerungen führen können [Kre03b].

Schließlich sind auch die Spannungsniveaus in den verschiedenen Teilnetzen wichtig und müssen zwischen den Betreibern ausgetauscht werden, um vertraglich festgelegte Vorgaben einhalten zu können.

Lebensdauer. Auch Stromnetze sind sehr langlebige Infrastrukturen. Wie erwähnt, begann der Aufbau des Stromnetzes in Deutschland bereits Ende des 19. Jahrhunderts. Aktuell haben Transformatoren auf der Ebene der Höchst- und Hochspannungsnetze ein durchschnittliches Alter zwischen 25 und 30 Jahren. Leistungs- und Trennschalter kommen auf etwa 20 Jahre, wohingegen die Masten für 220 kV-Netze durchschnittlich bereits bis zu 50 Jahre im Einsatz sind [Bun08]. Auch für Leitungen, wie z. B. Mittelspannungskabel, ist eine Lebensdauer von mindestens 20 Jahren vorgesehen. Bei der

Energieerzeugung sind Windenergienanlagen für eine Lebensdauer von 20 bis 25 Jahren, Kohle- und Kernkraftwerke für 35 bis 40 Jahre und Wasserkraftanlagen sogar für 60 bis 90 Jahre ausgelegt.

2.2.2. Herausforderungen für die Zustandsüberwachung

Wie bei Schienennetzen setzen auch die Betreiber von Stromnetzen Zustandsüberwachung ein, um Betriebsmittel besser zu nutzen, Netzausfälle zu vermeiden und Reparatur, Aufrüstung und Ersatz besser zu planen. Um zukünftigen Herausforderungen gewachsen zu sein, muss diese Zustandsüberwachung präziser werden. Dazu lassen sich folgende Anforderungen im Vergleich zu heutigen Systemen ableiten [Jul07, Kre03a, Kre03b]:

- *Automatische Integration von Diagnosedaten*: Diagnosedaten werden von unterschiedlichsten Systemen produziert und nur isoliert betrachtet, wie z. B. der Ölstand eines Transformators und der Zustand seines Kühlsystems. Um Zustände für zustandsorientierte Wartungsentscheidungen erkennen zu können, müssen Diagnosedaten in Beziehung gesetzt werden. Zudem werden Diagnosedaten heutzutage häufig noch manuell erfasst. Für den konsequenten Einsatz der Zustandsüberwachung muss dies automatisiert geschehen, um auch eine Selbstüberwachung zu ermöglichen.

- *Einbeziehung zusätzlicher Informationen*: Neben den Diagnosedaten, die von speziell installierten Überwachungssystemen geliefert werden, sind auch externe Informationen entscheidend, um einen Zustand beurteilen zu können. Dazu gehören beispielsweise Umgebungsinformationen wie Baumbestand und der Typ einer Stromleitung sowie Umweltbedingungen wie Blitzschlag und Niederschläge.

- *Befriedigung unterschiedlicher Informationsbedürfnisse*: Für unterschiedliche Zwecke werden Zustandsinformationen unterschiedlicher Art benötigt. Für den Betrieb ist die Lastverteilung relevant, während für die Instandsetzung die Abnutzung im Vordergrund steht. Deshalb soll es möglich sein, dieselben Ausgangsinformationen je nach Zweck unterschiedlich aufzubereiten.

- *Interoperabilität*: Ausgangsdaten für die Zustandsüberwachung werden aus unterschiedlichen Quellen integriert. Aufbereitete Zustandsinformation werden für unterschiedliche Zwecke genutzt sowie zwischen Organisationen, wie z. B. unterschiedlichen Stromnetzbetreibern, ausgetauscht. Um dies kosteneffizient umsetzen zu können, werden genormte und konsistente Beschreibungen dieser Informationen benötigt.

- *Einheitliche Systemarchitektur*: Ein weiterer Grund, warum Diagnosedaten in Stomnetzen bisher nur unzureichend integriert werden, ist das Fehlen einer geeigneten Systemarchitektur. Diese sollte einheitliche Ansätze für den Zugriff auf Daten, ihre Integration, ihre Speicherung sowie ihre Analyse beinhalten.

2.2.3. Stand der Technik bei der Zustandsüberwachung

Auch im Stromnetz sind bereits spezialisierte Systeme zur Fernüberwachung im Einsatz. Diese konzentrieren sich jeweils auf bestimmte Eigenschaften von ausgewählten Infrastrukturkomponenten und überwachen beispielweise die Funktionstüchtigkeit von Unterbrechern [KRL05]. Diese Systeme sind eigenständig und ziehen keinen Nutzen aus der Berücksichtigung anderer Datenquellen [Kre03a, Jul07].

Auf der Ebene der Umspannwerk-Automatisierung verfolgen Spezialanbieter wie *Cannon Technologies* erste integrierte Ansätze: Hier werden Informationen von verschiedenen Überwachungssystemen gesammelt, gefiltert und anhand vordefinierter Regeln verarbeitet. Bei Transformatoren werden unter anderem die Last, die Temperatur, der Kühlungszustand und die Unversehrtheit der Abdichtung überwacht; bei Unterbrechern die Dichte des Dämmgases sowie Auslösungs- und Schließzeiten; bei Leitungen die Blindleistung sowie die Last- und Spannungsprofile. Diese Systeme nutzen jedoch proprietäre Protokolle und Datenformate und unterstützen deshalb in der Regel nur Überwachungssysteme desselben Herstellers.

IT-Systeme der großen Hersteller für Stromnetz-Leitstände[2] bereiten die Informationen mehrerer Überwachungssysteme für den Operator grafisch auf. Die Interpretation dieser Information ist weiterhin Aufgabe eines Menschen [TK07]. Zudem verfügen Netzbetreiber heutzutage nur über Statusinformationen zu ihrem eigenen Netz, weil kein Datenaustausch mit anderen Betreibern stattfindet [Usl05, TK07].

In der Forschung werden erste generische Ansätze verfolgt. Diese verwenden formale Ontologien als Wissensmodell und versuchen so, die für die Domäne relevanten Informationen unabhängig von konkreten Überwachungssystemen zu modellieren [SLW+07, TK07]. Existierende Modelle beruhen jedoch auf Ad-hoc-Modellierungsansätzen und bilden nur sehr kleine Ausschnitte der Domäne ab. Vor allem fehlen prinzipielle Richtlinien zum konsistenten Aufbau eines Gesamtmodells [FWF04]. Grundsätzlich steht bei diesen Modellen die einfache Dokumentation von Zusammenhängen im Vordergrund, weshalb die formalisierte Semantik in der Regel nicht für eine automatisierte Auswertung geeignet ist [FSWF05]. Es wird zudem nicht betrachtet, wie derartige Konzepte in bestehenden Stromnetzen realisiert werden können.

Insgesamt muss festgestellt werden, dass auch bei Stromnetzen bisher kein System existiert, das eine semantische Integration und Auswertung von Zustandsinformationen ermöglicht. Zwar ist die semantische Interoperabilität zwischen Betreibern Gegenstand aktueller Forschungsprojekte, wozu auch ontologiebasierte Ansätze verfolgt werden [SLW+07, FSWF05]. Hier werden jedoch die automatisierte Auswertbarkeit der formalisierten Semantik und die Realisierbarkeit in bestehenden Stromnetzen vernachlässigt.

2.3. Generische Anforderungen an die Zustandsüberwachung für Infrastrukturnetze

In den vorherigen Abschnitten wurden Schienen- und Stromnetze als repräsentative Beispiele für Infrastrukturnetze vorgestellt und jeweils die wichtigsten Herausforderungen für die Zustandsüberwachung identifiziert. Darauf aufbauend werden nachfolgend

[2] z. B. Siemens Energy: http://www.energy-portal.siemens.com

generische Anforderungen an die Zustandsüberwachung für Infrastrukturnetze formuliert.

Prinzipiell gilt, dass Infrastrukturzustände auf Basis unterschiedlichster Informationen charakterisiert werden können. Diese werden nicht ausschließlich von klassischen Überwachungssystemen geliefert, sondern umfassen auch Umweltbedingungen, die von externen Diensten bereitgestellt werden, genauso wie Inspektions- und Wartungsberichte, die in Datenbanken vorgehalten werden.

Eine analoge Situation stellt sich bei kontextsensitiven Systemen, die eine Kernfunktionalität von Systemen des *Ubiquitous Computing* und der *Ambient Intelligence* darstellen. Dort werden derartige Informationen als *Kontextinformationen* bezeichnet. Aufgrund dieser Gemeinsamkeiten betrachtet diese Arbeit Zustandsüberwachungssysteme für Infrastrukturnetze als kontextsensitive Systeme und bezeichnet zustandsrelevante Informationen einheitlich als *Kontextinformationen* (vergleiche Kapitel 1).

Kontext und *Kontextsensitivität* von Systemen werden in Anlehnung an Dey üblicherweise definiert als [Dey01]:

> „Kontext *ist jede Information, die zur Beschreibung der Situation einer Entität genutzt werden kann.*
>
> *Eine Entität ist eine Person, ein Ort oder ein Objekt, das als relevant für die Interaktion zwischen einem Nutzer und einer Anwendung erachtet wird, inklusive dem Nutzer und der Anwendung selbst.*
>
> *Ein System ist* kontextsensitiv*, wenn es Kontext nutzt, um dem Nutzer relevante Informationen und/oder Dienste anzubieten, wobei die Relevanz von der Aufgabe des Nutzers abhängt.*“

Ausgehend von den bereits erläuterten Eigenschaften und Anforderungen von Schienen- und Stromnetzen lassen sich folgende generischen Anforderungen für Zustandsüberwachungssysteme in Infrastrukturnetzen identifizieren:

A2.1 *Integration von Kontextinformationen*: Um Infrastrukturzustände präzise beurteilen zu können, müssen Kontextinformationen im Zusammenhang gesehen werden. Auch Umweltbedingungen, die bisher nicht durch Überwachungssysteme erfasst werden, müssen berücksichtigt werden.

A2.2 *Unterstützung komplexer Zusammenhänge*: Die dafür relevanten Zusammenhänge können sehr komplex sein: Sie können unterschiedlichste Arten und Kombinationen von Kontextinformationen betreffen. Je nach Nutzungszweck können außerdem unterschiedliche Sichten und Auswertungen nötig sein.

A2.3 *Modellierung durch unterschiedliche Domänenexperten*: Das Wissen über die relevanten Zusammenhänge ist über viele Domänenexperten verteilt. Es wird ein Ansatz benötigt, mit dem das jeweilige Teilwissen auf konsistente Art und Weise zu einem Gesamtsystem zusammengetragen werden kann.

A2.4 *Umgang mit unvollständigen Daten*: Da Kontextinformationen von zahlreichen Systemen produziert und diese von unterschiedlichen Organisationen betrieben werden, kann nicht davon ausgegangen werden, dass alle relevanten Informationen immer vollständig vorliegen. Stattdessen muss es möglich sein, auch mit

unvollständigen Daten umgehen zu können. Vor allem dürfen aus ihnen keine falschen Schlussfolgerungen gezogen werden.

A2.5 *Unterstützung unterschiedlicher Organisationen*: Die Zustandserkennung wird erschwert durch die große Zahl an Organisationen, die an Nutzung und Instandhaltung einer Infrastruktur und damit auch an der Produktion von Kontextinformationen beteiligt sind. Um diese integrieren zu können, ist neben syntaktischer auch semantische Interoperabilität erforderlich. Auch abgeleitete Kontextinformationen und Zustände müssen semantisch interoperabel repräsentiert werden, damit sie wiederum von unterschiedlichen Organisationen für ihre jeweiligen Zwecke genutzt und ausgetauscht werden können.

A2.6 *Eignung für räumlich stark verteilte Systeme*: In Infrastrukturnetzen wird die Zustandserkennung zusätzlich dadurch erschwert, dass Kontextinformationen aufgrund der geographischen Ausdehnung der Infrastrukturen verteilt produziert und in der Regel verteilt gehalten werden. Es müssen also Konzepte entwickelt werden, die mit verteilten Daten zurecht kommen.

A2.7 *Eignung für lange Lebensdauer und permanenten Ausbau*: Es hat sich gezeigt, dass Infrastrukturnetze für lange Lebensdauern ausgelegt sind und gleichzeitig permanent erweitert und ausgebaut werden. Deshalb muss ein System zur Zustandsüberwachung generisch und flexibel konzipiert sein, um seine Zukunftsfähigkeit sicherzustellen. Dazu muss es so anpassbar und erweiterbar sein, dass es zusätzliche Kontextquellen einbinden und neuartige Auswertungsmöglichkeiten umsetzen kann, ohne bestehende Funktionalitäten zu beeinträchtigen.

2.4. Zusammenfassung

Dieses Kapitel identifizierte Anforderungen an Zustandsüberwachungssysteme für Infrastrukturnetze. Dazu wurden Schienen- und Stromnetze als repräsentative Beispiele für Infrastrukturnetze vorgestellt und ihre Eigenschaften sowie Anforderungen an die Zustandsüberwachung systematisch analysiert. Außerdem wurde jeweils der Stand der Technik skizziert und gezeigt, weshalb bisherige Ansätze nicht ausreichen. Diese Untersuchung ergab, dass (i) unterschiedlichste Informationen für die Zustandsbeurteilung relevant sind und deshalb nicht nur Meldungen von Überwachungssystemen, sondern allgemein Kontextinformationen berücksichtigt werden müssen. Außerdem (ii) besitzen Zustandsüberwachungssysteme für Schienen- und Stromnetze im Wesentlichen dieselben Anforderungen, weshalb es möglich ist, allgemeingültige Anforderungen an die Zustandsüberwachung für Infrastrukturnetze aufzustellen. Vor dem Hintergrund des so erstellten Anforderungskatalogs wird im Folgenden in das Gebiet der formalen Wissensrepräsentation und -verarbeitung eingeführt.

3. Formale Wissensrepräsentation und Reasoning

Bisher wurden aktuelle Probleme und Herausforderungen bei der Zustandsüberwachung von Infrastrukturnetzen analysiert. Dieses Kapitel führt in den Abschnitten 3.1 und 3.2 wichtige Begriffe und Konzepte zur formalen Wissensrepräsentation und -verarbeitung ein. Es schafft damit die Grundlage für eine semantische Modellierung und Verarbeitung von Kontextinformationen in Infrastrukturnetzen. Aufgrund ihrer attraktiven Eigenschaften wird der Schwerpunkt in Abschnitt 3.3 auf Beschreibungslogiken gelegt. Als eine formale Basis des *Semantic Web*-Standards OWL und damit auch dieser Arbeit führt Abschnitt 3.4 schließlich die Beschreibungslogik $\mathcal{SHIN}$ ein.

3.1. Wissensbasierte Systeme

Wissensrepräsentation und -verarbeitung (engl. *knowledge representation and reasoning,* KRR) sind motiviert durch *wissensbasierte Systeme*: Während bei konventionellen Programmen zwischen *fallabhängigen Daten* und *domänenabhängigen Algorithmen* differenziert wird, unterscheiden wissensbasierte Systeme zwischen *fallabhängigen Daten*, einem *domänenabhängigen Wissensmodell* und einem *domänenunabhängigen Reasoning-Algorithmus* (siehe Abbildung 3.1) [Pup91].

Das grundsätzliche Ziel bei wissensbasierten Systemen besteht darin, das Wissen über einen bestimmten Problembereich von den Verfahren zur Wissensverarbeitung (*Reasoning*) zu trennen [BKI06]. Für die Entwicklung einer neuen Anwendung wird lediglich ein entsprechendes Wissensmodell für das neue Problemgebiet (*Domäne*) erstellt; das Verfahren zur Wissensverarbeitung kann wiederverwendet werden. Fallabhängige Daten werden vom (domänenunabhängigen) Reasoning-Algorithmus nun mit dem neuen (domänenabhängigen) Wissensmodell verarbeitet und entsprechend der dort formalisierten

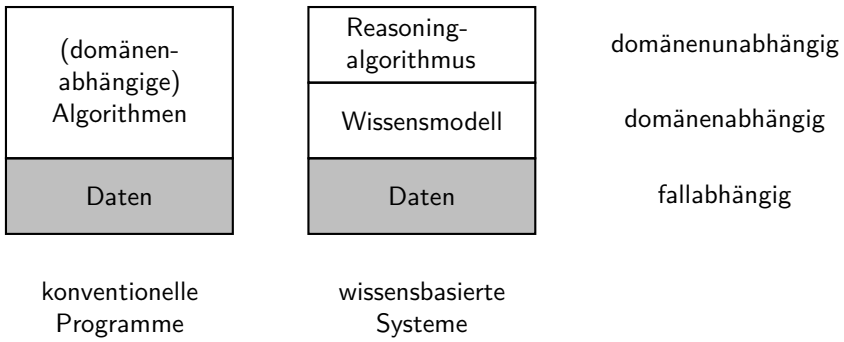

Abbildung 3.1.: Wissensbasierte Systeme und konventionelle Programme im Vergleich.

Zusammenhänge interpretiert. Dadurch wird allein durch die geeignete Erstellung des Wissensmodells das gewünschte Verhalten der Anwendung erreicht. Wissensmodelle werden mit einer *Wissensrepräsentationssprache* formuliert. Anwendungsentwickler können bei der Erstellung der Wissensmodelle durch Werkzeuge unterstützt werden. Der domänenunabhängige Reasoning-Algorithmus ist außerdem gut für den Umgang mit unvollständigen oder nur teilweise erfassbaren Daten geeignet [RN03].

Zur weiteren Charakterisierung wissensbasierter Systeme werden zunächst die zentralen Begriffe *Wissen*, *Repräsentation* und *Reasoning* definiert (vergleiche [BL04]):

Wissen. Philosophen beschäftigen sich seit der Antike kontrovers mit der Definition von Wissen. In der Informatik werden unter Wissen jedoch einfach *Aussagen* verstanden, die von einem Wissensträger als *wahr* angesehen werden. Wissen wird also immer als Relation zwischen einem Wissensträger und einer Aussage aufgefasst. Ein Beispiel dafür ist: „John weiß, dass ein Auto vier Räder hat“, wobei „John“ der Wissensträger und „Ein Auto hat vier Räder“ die Aussage ist.

Repräsentation. Der Begriff der Repräsentation ist philosophisch ähnlich kompliziert. Sie kann als Relation zwischen zwei Bereichen aufgefasst werden, wobei Elemente des ersten Bereichs für Elemente des zweiten Bereichs stehen (siehe Illustration in Abbildung 3.2). Der erste Bereich ist in der Regel konkreter und besser greifbar als der zweite.

Elemente des ersten Bereichs sind normalerweise *Symbole*, d. h. Zeichen oder Zeichenketten, die aus einem gegebenen Alphabet gebildet werden. Zum Beispiel steht das Symbol „7“ für die Zahl sieben, genauso wie das Symbol „VII“ oder „七“ (das chinesische Zeichen für die Zahl sieben). Im Gegensatz zu den Konzepten, die sie repräsentieren, sind diese Symbole für Computerprogramme zugänglich und manipulierbar.

Interessant wird es, wenn aus formalen Symbolen *Sätze* gebildet werden, die Aussagen repräsentieren. Zum Beispiel kann der Satz „Ein Auto hat vier Räder“ eine Repräsentation für die Aussage sein, dass ein Auto vier Räder hat.

Die Spezifikation, welche Symbole in einer Wissensrepräsentationssprache zur Verfügung stehen und wie daraus Sätze aufgebaut werden können, bezeichnet man als *Syntax* der Wissensrepräsentationssprache. Die Zuordnung von Symbolen und Sätzen der Repräsentation zu den Konzepten der repräsentierten Welt wird *Semantik* genannt. Erst durch eine derartige Relation erlangen die Sätze der Repräsentationssprache eine Bedeutung, die festlegt, ob ein Satz der Sprache in einer gegebenen Welt eine wahre Begebenheit bezeichnet oder nicht.

Wissensrepräsentation beschäftigt sich also mit der Nutzung von Symbolen, um eine Sammlung von Aussagen formal darzustellen, die für einen (mutmaßlichen) Wissensträger wahr sind. Dabei müssen nicht alle Aussagen dargestellt werden, die der Wissensträger als wahr erachtet, denn deren Zahl ist potentiell unendlich groß. Stattdessen wird davon ausgegangen, dass nur eine endliche Zahl von Aussagen (explizit) repräsentiert ist. Aus diesen weitere Aussagen abzuleiten, die implizit ebenfalls wahr sind, ist Aufgabe des Reasoning.

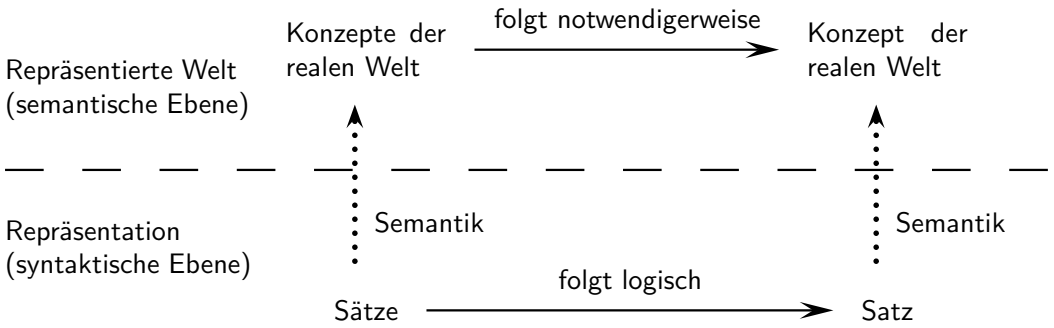

Abbildung 3.2.: Syntaktische und semantische Ebene [RN03].

Reasoning. Reasoning ist die formale Manipulation der Repräsentation von Aussagen (also Symbolen), um die *Repräsentationen weiterer Aussagen* zu erzeugen, welche notwendigerweise aus diesen folgen (vergleiche Abbildung 3.2). Ein Beispiel ist die Umformung der Zeichenkette „$2 + 5$“ zu der Zeichenkette „7“. Dabei wird ausgenutzt, dass Symbole – im Gegensatz zu den Aussagen, die sie repräsentieren – für Computerprogramme zugänglich und manipulierbar sind. Üblicherweise wird *Reasoning* als allgemeiner Begriff zur Beschreibung des Prozesses aufgefasst, mit dem Schlussfolgerungen erreicht werden.

So könnten zum Beispiel die Sätze „Ein Golf ist ein Auto“ und „Ein Auto hat vier Räder“ (als Repräsentationen der entsprechenden Aussagen) durch Manipulation der Symbole umgeformt werden zu dem Satz „Ein Golf hat vier Räder“ als eine Repräsentation der Aussage, dass ein Golf vier Räder hat. Diese Form des Reasonings würde man als *(logische) Inferenz* bezeichnen, da das Ergebnis eine logische Schlussfolgerung aus den Aussagen darstellt, die von den Ausgangssätzen repräsentiert werden. Eine andere Form des Reasonings ist aber z. B. auch topologisches Schlussfolgern auf Graphen: Der Graph repräsentiert die Nachbarschaftsbeziehung der Knoten. Bei der Suche nach allen Knoten, die von einem bestimmten Knoten aus erreichbar sind, wird dieser Graph so manipuliert, dass ein Graph entsteht, der die Erreichbarkeitsbeziehung repräsentiert.

Ein wissensbasiertes System hat demzufolge zwei zentrale Komponenten: eine *Wissensbasis* (engl. *knowledge base*) und einen *Reasoning-Algorithmus*. Die Wissensbasis enthält eine Menge von Sätzen, die Aussagen repräsentieren. Die Sätze sind mit einer Wissensrepräsentationssprache formuliert. Der Reasoning-Algorithmus leitet aus den Sätzen, die in der Wissensbasis repräsentiert sind, weitere Sätze ab, die wahr sind.

Daraus ergeben sich zwei Kernfragen für wissensbasierte Systeme: Wie wird das Wissen in der Wissensbasis repräsentiert? Und wie werden aus gegebenen Sätzen neue Sätze abgeleitet? Dabei ist die Frage des Reasonings untrennbar mit der Frage der Repräsentation des Wissens verbunden. Bei der Wissensrepräsentation wird prinzipiell zwischen *deklarativen* und *prozeduralen* Formen unterschieden [Kur92]:

Deklarative Wissensrepräsentation beschreibt *Sachverhalte* wie z. B.: „Die Summe aus 2 und 5 ist 7.“ Sie macht keine Aussage über Konstruktion und Gebrauch von Wissen zur Lösung eines konkreten Problems. Das Wissen wird als Sammlung von Fakten dargestellt. Die Vorteile der deklarativen Darstellung liegen darin, dass die

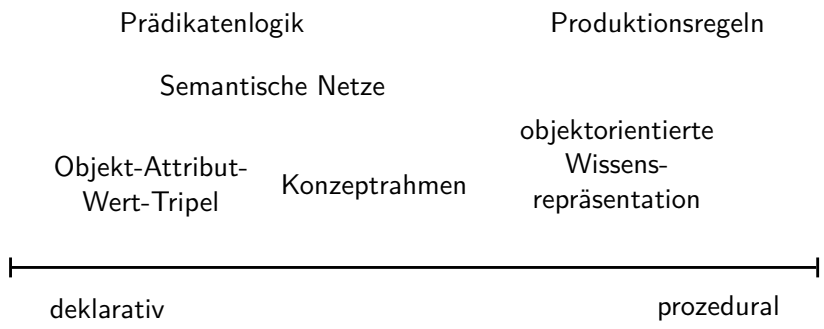

Abbildung 3.3.: Deklarative und prozedurale Wissensrepräsentation.

Darstellung unabhängig von der späteren Verarbeitung ist und somit diesbezüglich nicht auf ein bestimmtes Verfahren beschränkt ist. Bei der Modellierung muss also nicht berücksichtigt werden, wie das Wissen später ausgewertet wird. Jede Wissenseinheit muss nur einmal gespeichert werden, unabhängig davon, wie oft und in welcher Weise sie verwendet wird. Neue Wissenseinheiten können einfach hinzugefügt, modifiziert oder entfernt werden. Die Prädikatenlogik ist ein Beispiel für deklarative Wissensrepräsentation.

Prozedurale Wissensrepräsentation beschreibt *Verfahren* zur Konstruktion, Verknüpfung und Anwendung von Wissen, beispielsweise ein Verfahren zur Berechnung der Summe aus 2 und 5. Diese Repräsentationsform hat den Vorteil, dass so repräsentiertes Wissen sehr effizient ausgewertet werden kann. Bei bestimmten Sachverhalten kann es auch leichter fallen, das nötige Wissen prozedural anstatt deklarativ zu beschreiben. Nachteilig ist jedoch, dass Änderungen am repräsentierten Wissen sehr aufwändig sind, weil dazu Anpassungen an mehreren unterschiedlichen Stellen erforderlich sein können. Regeln, die häufig als Hornklauseln formalisiert werden, sind ein Beispiel für prozedurale Wissensrepräsentation.

Neben rein deklarativer und rein prozeduraler Wissensrepräsentation gibt es auch Mischformen. Abbildung 3.3 gibt einen Überblick. Bei der Zustandsüberwachung für Infrastrukturnetze sind besonders Verständlichkeit, Erweiterbarkeit und Wartbarkeit der zugrunde liegenden Wissensmodelle unverzichtbar. Deshalb werden im Folgenden *deklarative* Wissensrepräsentation und – als wichtige Vertreter davon – logikbasierte Formalismen betrachtet.

3.2. Logikbasierte Wissensrepräsentation

Logik als die „Lehre des vernünftigen Schlussfolgerns“ ist prädestiniert für Wissensrepräsentation und Reasoning. Prominente Beispiele sind Aussagenlogik, Prädikatenlogik erster Stufe und Beschreibungslogiken. In einer Logik sind sowohl die Syntax als auch die Semantik mathematisch präzise definiert. Auf dieser Basis lässt sich auch das Reasoning, das bei Logiken in der Regel als Inferenz oder logische Schlussfolgerung bezeichnet wird, formalisieren.

3.2.1. Syntax und Semantik

Zur Repräsentation von Wissen mit einer Logik müssen ihre Syntax und Semantik definiert werden. Die syntaktische Ebene wird festgelegt mittels *Signaturen* und *Formeln*. Die semantische Ebene umfasst *Interpretationen*, die die Zuordnung syntaktischer Elemente einer Wissensbasis zu den Objekten der repräsentierten Welt festlegen, und *Erfüllungsrelationen*, die die Gültigkeit einer Formel in einer Interpretation angeben. Diese Zusammenhänge sind in Abbildung 3.4 illustriert.

Die syntaktische Ebene betrifft die Darstellung von Sätzen in der Wissensbasis. Die Grundlage dafür ist ein Vokabular, d. h. eine Menge von Namen. Dieses wird als Signatur bezeichnet. Eine Signatur Σ ist in der Regel domänenabhängig: In der Arithmetik umfasst sie Symbole wie $0, 1, a, +, \leq$; im Schienenverkehr könnten hingegen Symbole wie Achslager, Drehgestell, Funktionstüchtig, Fehlerhaft verwendet werden. Mit Hilfe der Elemente der Signatur werden wohlgeformte Formeln auf Basis bestimmter Regeln aufgebaut. Diese Regeln sind domänenunabhängig. In der Logik stehen dafür binäre Verknüpfungoperatoren (Junktoren) wie z. B. $\wedge$ („und“), $\vee$ („oder“) und $\Rightarrow$ (Implikation) zur Verfügung, die rekursiv angewandt werden können, um komplexe Formeln zu konstruieren. Sei $Formel(\Sigma)$ die Menge der Formeln, die in einer bestimmten Logik über Σ gebildet werden kann. Dann ist eine Wissensbasis W eine Menge von Formeln über Σ, d. h. $W \subseteq Formel(\Sigma)$.

Die semantische Ebene stellt eine Beziehung zwischen den syntaktischen Elementen einer Wissensbasis und den Objekten der repräsentierten Welt her. Zunächst wird diese Beziehung für die Signatur Σ betrachtet. Dazu muss der Begriff der Interpretation eingeführt werden: Eine *Interpretation* $\mathcal{I}(\Sigma)$ besteht aus einer nicht-leeren *Grundmenge* $\Delta^{\mathcal{I}}$ (auch: *Domäne, Universum*) und einer Abbildung $\cdot^{\mathcal{I}} : \Sigma \to \Delta^{\mathcal{I}}$, die jedem Namen in Σ ein Element von $\Delta^{\mathcal{I}}$ zuweist. Wenn man sich als Grundmenge $\Delta^{\mathcal{I}}$ die Objekte der repräsentierten Welt vorstellt, weist eine Interpretation also den (willkürlich benannten) Elementen der Signatur eine Bedeutung in der repräsentierten Welt zu. Auf Basis von $Formel(\Sigma)$ und $\mathcal{I}(\Sigma)$ kann nun eine Erfüllungsrelation definiert werden, die angibt, wann eine Formel in einer Interpretation gilt, d. h. ob eine Formel F in einer Interpretation $\mathcal{I}$ wahr oder falsch ist. Dazu muss zusätzlich zu der Interpretation der Elemente in Σ die Interpretation der Junktoren definiert werden. So ist $A \wedge B$ zum Beispiel erfüllt, wenn sowohl A als auch B erfüllt sind. Damit kann auf Basis der Erfüllungsrelation die *logische Folgerung* (engl. *entailment*) definiert werden:

> Aus F folgt logisch G (geschrieben $F \models G$) genau dann, wenn *jede* Interpretation, die F erfüllt, auch G erfüllt.

Dabei ist wichtig zu verstehen, dass F selbst wahr sein muss. Denn ist F nicht wahr, lässt sich gemäß Definition von $\models$ jede Aussage ableiten!

Eine grundlegende Entscheidung ist, wie die Aussage $F \not\models G$ interpretiert wird. Bei der Annahme einer geschlossenen Welt (engl. *closed world assumption, CWA*) wird sie als $F \models \neg G$ interpretiert: Was auf Basis des gegebenen Wissens nicht explizit als wahr bewiesen werden kann, wird als falsch angesehen (engl. „negation as failure“). Diese Annahme wird z. B. bei PROLOG oder Datenbanken getroffen: Gibt es für eine Person keinen Eintrag im Telefonbuch, wird angenommen, dass diese Person kein Telefon besitzt.

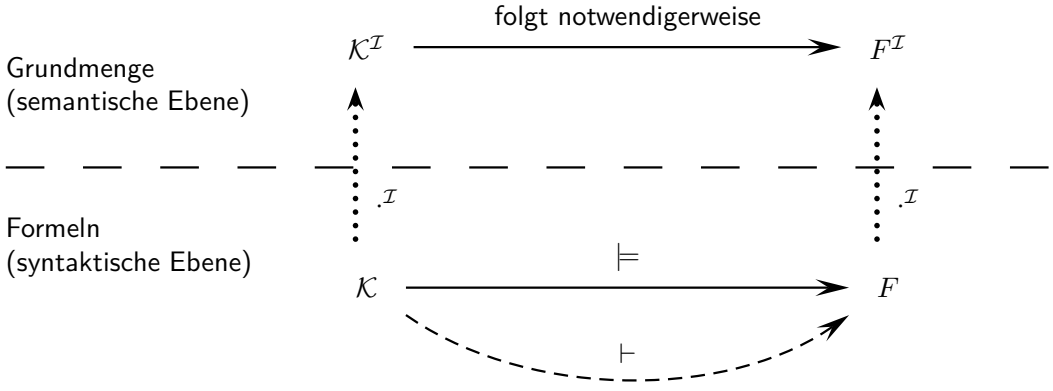

Abbildung 3.4.: Syntaktische und semantische Ebene bei logischen Formalismen.

Im Gegensatz dazu kann auch eine offene Welt angenommen werden (engl. *open world assumption, OWA*): Hier wird aus $F \not\models G$ nicht automatisch auf $F \models \neg G$ geschlossen, sondern $F \models \neg G$ wird nur dann als wahr angenommen, wenn diese Aussage auch explizit für alle Modelle abgeleitet werden kann (engl. „negation as unsatisfiability“). Bei der Annahme einer offenen Welt wird also prinzipiell davon ausgegangen, dass das bekannte Wissen unvollständig ist. Diese Annahme wird in den klassischen Logiken und damit auch in der Prädikatenlogik und bei Beschreibungslogiken getroffen: Fehlt für eine Person ein Eintrag im Telefonbuch, wird nicht gefolgert, dass diese Person kein Telefon besitzt. Stattdessen wird berücksichtigt, dass diese Information möglicherweise lediglich (noch) nicht bekannt ist.

3.2.2. Inferenz

Reasoning bei logikbasierten Formalismen wird in der Regel als Inferenz bezeichnet. Eine *Inferenzprozedur* (ein *Kalkül*) K besteht aus Axiomen und Inferenzregeln. Kann aus einer Menge von Formeln $F_1, \ldots, F_n$ durch eine Folge von Anwendungen dieser Inferenzregeln die Formel F abgeleitet werden, so schreiben wir dafür $F_1, \ldots, F_n \vdash_K F$. Eine Inferenzprozedur K definiert also eine Ableitungsrelation $\vdash_K$ zwischen Formeln bzw. Formelmengen. Es werden klassischerweise folgende Formen der Inferenz unterschieden [Pei31, BKI06]:

Deduktion. Bei deduktiver Inferenz (auch: *folgerichtigem Schließen*) sind eine Inferenzprozedur und eine Menge von Aussagen als Ausgangsbasis gegeben. Durch Anwendung der Inferenzprozedur werden weitere Aussagen abgeleitet. Diese Schlussfolgerungen sind alle korrekt, d. h. abgeleitete Aussagen sind stets wahr. Deduktion ist also eine Form des sicheren Schließens. Dabei schließt man in der Regel von einem allgemeinen auf einen speziellen Fall.

Das klassische Beispiel von Peirce lautet: „Alle Bohnen aus diesem Beutel sind weiß; diese Bohnen sind aus diesem Beutel, folglich sind sie weiß.“ [Pei31]

Deduktion ist außerdem eine Form des *monotonen Schließens*, d. h. alle Aussagen, die einmal abgeleitet werden konnten, bleiben auch dann wahr, wenn zusätzliche Aussagen hinzugenommen werden. Umgekehrt bedeutet dies, dass abgeleitete Aussagen durch Hinzunahme weiterer Aussagen nicht revidiert werden können.

Abduktion. Bei abduktiver Inferenz (auch: *ursächlichem Schließen*) sind eine Inferenzprozedur und eine Menge von Aussagen als Endergebnis gegeben. Unter Berücksichtigung der Inferenzprozedur wird abgeleitet, aus welcher Ausgangsbasis das gegebene Endergebnis gefolgert werden kann. Abduktion sucht also nach einer Erklärung für einen beobachteten Sachverhalt.

Das Beispiel hier: „Alle Bohnen aus diesem Beutel sind weiß; diese Bohnen sind weiß, also sind diese Bohnen aus diesem Beutel."

Diese Erklärungen sind jedoch nicht unbedingt eindeutig und können insbesondere auch falsch sein. Deshalb ist das abgeleitete Wissen nicht immer wahr, und Abduktion ist im Gegensatz zur Deduktion keine Form des sicheren Schließens.

Induktion. Bei Induktion (auch: *generalisierendem Schließen*) sind eine Menge von Aussagen als Ausgangsbasis und eine Menge von Aussagen als Endergebnis gegeben. Die induktive Inferenz versucht nun, eine Inferenzprozedur zu ermitteln, mit der das Endergebnis aus der Ausgangsbasis abgeleitet werden kann. Hier geht es also darum, regelhaftes Wissen aus einzelnen Sachverhalten zu erschließen.

Im Beispiel: „Diese Bohnen sind aus diesem Beutel; diese Bohnen sind weiß, folglich sind alle Bohnen aus diesem Beutel weiß."

Damit ist Induktion ebenfalls eine Form des unsicheren Schließens, weil eine induktiv abgeleitete Inferenzprozedur zwar mit allen beobachteten Sachverhalten vereinbar, im Allgemeinen aber trotzdem nicht gültig sein kann.

3.2.2.1. Korrektheit und Vollständigkeit von Inferenzprozeduren

Wichtige Eigenschaften von Inferenzprozeduren sind Korrektheit und Vollständigkeit:

Eine Inferenzprozedur ist (*logisch*) *korrekt*, wenn syntaktisch abgeleitetes Wissen auch semantisch logisch gefolgert werden kann, d. h. wenn für beliebige Formeln F und G gilt:

$$F \vdash G \quad \Longrightarrow \quad F \models G.$$

Eine korrekte Inferenzprozedur leitet nur Schlussfolgerungen ab, die auch auf der semantischen Ebene notwendig gültige Schlussfolgerungen sind.

Eine Inferenzprozedur ist *(logisch) vollständig*, wenn alle logischen Folgerungen auch mittels des Kalküls abgeleitet werden, d. h. wenn für beliebige Formeln F und G gilt:

$$F \models G \quad \Longrightarrow \quad F \vdash G.$$

Eine vollständige Inferenzprozedur findet für jede semantische Folgerung einen Beweis.

Die einzige Form des sicheren Schließens, d. h. eine im obigen Sinne *korrekte* Inferenzprozedur, ist die *Deduktion*. Ziel bei der Definition des Herleitungsoperators $\vdash$ ist es also, wie in Abbildung 3.4 illustriert, den semantisch definierten Folgerungsoperator $\models$ mittels $\vdash$ auf syntaktischer Ebene nachzubilden. Bei *Abduktion* und *Induktion* handelt es sich hingegen nicht um sichere Schlussweisen.

Für deduktive Inferenzprozeduren gibt es viele Beispiele: Für die Aussagenlogik gibt es die aussagenlogische Resolution. Für die Prädikatenlogik existiert eine prädikatenlo-

gische Variante. Beide beruhen auf dem Zusammenhang:

$$KB \models F \iff KB \cup \{\neg F\} \text{ ist unerfüllbar.}$$

Auch für logisches Programmieren wird in der klassischen Variante ein Resolutionskalkül eingesetzt, der auf Hornklauseln (d. h. Klauseln mit höchstens einem nicht-negierten Literal) arbeitet und deshalb besonders effizient ist.

3.2.2.2. Entscheidbarkeit und Komplexität von Problemen

Während sich die Begriffe Korrektheit und Vollständigkeit auf Inferenzprozeduren beziehen, ist eine ganz andere Frage, ob es überhaupt eine Inferenzprozedur mit diesen Eigenschaften zu einem bestimmten *Problem* geben kann. Das hängt davon ab, ob dieses Problem für eine bestimmte Logik entscheidbar ist. Die *Entscheidbarkeit* eines Problems M ist wie folgt definiert. Dabei repräsentiert M die Menge der Lösungen für das Problem.

M ist *entscheidbar*.	$\iff$	Es gibt einen Algorithmus, der für jedes x angibt, ob $x \in M$ oder $x \notin M$.
M ist *unentscheidbar*.	$\iff$	M ist nicht entscheidbar.
M ist *semi-entscheidbar/ rekursiv aufzählbar*.	$\iff$	Es gibt einen Algorithmus, der für jedes x aus der Menge M angibt, dass $x \in M$ gilt. (Insbesondere muss der Algorithmus für ein $x \notin M$ nicht unbedingt terminieren!)

Beispielsweise ist das Allgemeingültigkeitsproblem der Aussagenlogik entscheidbar, das Allgemeingültigkeitsproblem der Prädikatenlogik erster Stufe (PL1) jedoch unentscheidbar [Chu36].

Entscheidbare Probleme können schließlich weiter bezüglich ihrer *Komplexität* klassifiziert werden: Sei $\mathcal{C}$ eine Komplexitätsklasse, z. B. P, PSpace, NP oder ExpTime. Dann ist ein Problem Q in $\mathcal{C}$, wenn es einen Algorithmus gibt, der zur Klasse $\mathcal{C}$ gehört und das Problem Q löst. Dadurch wird also eine *obere* Schranke für die Komplexität angegeben. Insbesondere ist „entscheidbar“ eine obere Grenze, denn hier ist $\mathcal{C}$ die Klasse aller entscheidbaren Probleme. Ein Problem Q ist *$\mathcal{C}$-hart*, falls alle Probleme der Klasse $\mathcal{C}$ polynomiell auf das Problem Q reduziert werden können. Dies stellt also eine *untere* Grenze dar, denn Problem Q kann insbesondere auch wesentlich schwieriger oder sogar unentscheidbar sein. Schließlich ist ein Problem Q *$\mathcal{C}$-vollständig*, wenn es sowohl $\mathcal{C}$-hart als auch in $\mathcal{C}$ ist.

Man unterscheidet zwischen *effizient lösbaren* (engl. *tractable*) und *nicht effizient lösbaren* (engl. *intractable*) Problemen. Als effizient lösbare Probleme werden im Allgemeinen Probleme in P betrachtet, also Probleme, die auf *deterministische* Weise in polynomieller Zeit lösbar sind. NP bezeichnet Probleme, die auf *nichtdeterministische* Weise in polynomieller Zeit lösbar sind. Für diese Probleme sind deterministische Entscheidungsverfahren in der Regel nur mit exponentiellem Zeitaufwand bekannt.

Deshalb sind diese Probleme nur für kleine Probleminstanzen praktikabel lösbar und werden als nicht effizient lösbar bezeichnet.

3.3. Ontologien und Beschreibungslogiken

Ontologien sind wichtige Vertreter deklarativer Wissensmodelle. Der Begriff wurde übernommen aus der Philosophie, wo Ontologie die Wissenschaft vom Sein und vom Seienden im Allgemeinen bezeichnet.

3.3.1. Definition

Zur Abgrenzung vom philosophischen Verständnis des Begriffs kann im Sinne der Informatik von *formaler Ontologie* gesprochen werden. Die am häufigsten zitierte Definition formaler Ontologien stammt von Gruber: „Eine Ontologie ist eine explizite Spezifikation einer formalen Konzeptionalisierung." [Gru93] Sie wurde von Studer et al. unter Verschiebung des Schwerpunkts adaptiert als: „Eine Ontologie ist eine formale, explizite Spezifikation einer gemeinsamen Konzeptionalisierung." [SBF98] Hier wird der Aspekt der Interoperabilität stärker betont, indem eine Ontologie als „gemeinsame" Konzeptionalisierung charakterisiert wird. Beide Definitionen sind zwar sehr prägnant, ohne Vorkenntnisse jedoch schwer zu verstehen.

In einem aktuellen Lexikoneintrag schlägt Gruber deshalb eine eingängigere Definition vor, die Bezug auf die zuvor beschriebenen Begriffe *Wissen* und *Repräsentation* nimmt [Gru08]:

> *„Eine* Ontologie *definiert (spezifiziert) die Konzepte, Beziehungen und andere Unterscheidungsmerkmale, die für die Modellierung eines Wissens- oder Sachgebiets (Domäne) relevant sind.*
>
> *Dies geschieht in der Form von Definitionen eines Repräsentationsvokabulars (Klassen, Relationen usw.), durch die dem Vokabular Bedeutung zugeschrieben und formale Beschränkungen für seine kohärente Nutzung festgelegt werden."*

Mit Hilfe eines derartigen Repräsentationsvokabulars kann dann Wissen über Individuen der zu repräsentierenden Welt ausgedrückt werden.

Die vorgestellten Ontologie-Definitionen treffen keine Annahme über die Wissensrepräsentationssprache, die zur Formulierung der Ontologie verwendet wird. Tatsächlich können Ontologien mit unterschiedlichen Graden an Formalität spezifiziert werden. Uschold schlägt eine Einordnung anhand des *semantischen Kontinuums* vor [Usc03]. Er unterscheidet vier Klassen, die nach ihrem Grad der Formalität geordnet sind (siehe Abbildung 3.5):

Implizit. Ontologien können rein *implizit* existieren, wie zum Beispiel die stillschweigende Übereinkunft, dass mit der „ersten Achse" eines Zuges die in Fahrtrichtung erste Achse gemeint ist.

Informal. Sie können *explizit* dokumentiert, aber dennoch *informal* sein, wie zum Beispiel ein natürlichsprachlicher Text in einem Standardisierungsdokument.

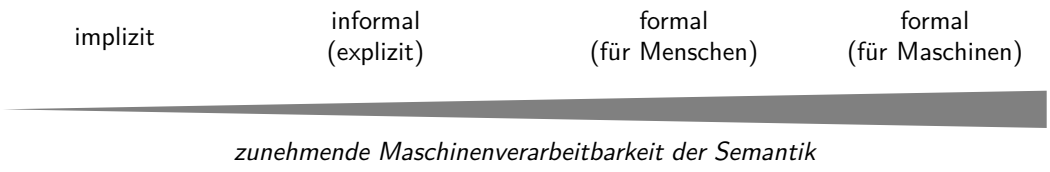

Abbildung 3.5.: Semantisches Kontinuum von Wissensrepräsentationssprachen (nach [Usc03]).

Formalisiert für Menschen. Ontologien können *explizit* dokumentiert und *für Menschen formalisiert* sein. Dazu reicht es aus, implizit die Semantik der relevanten Begriffe festzulegen und diese dann für die Repräsentation zu verwenden. Ein Beispiel hierfür ist die einheitliche Verwendung bestimmter Schlüsselbegriffe in einem Standardisierungsdokument.

Formalisiert für Maschinen. Schließlich können *explizite* Ontologien auch *für Maschinen formalisiert* sein. Dazu muss die Semantik der verwendeten Begriffe zusätzlich explizit und maschinenverarbeitbar formuliert werden. Dies gilt beispielweise für die logikbasierte Wissensrepräsentation. Diese Klasse der formalen und maschinenverarbeitbaren Ontologien wird in Abbildung 3.6 noch weiter differenziert [SW01].

Für den Zweck der automatisierten Zustandserkennung bei Infrastrukturnetzen wird eine Formalisierung für Maschinen benötigt, die gleichzeitig auch für den Menschen verständlich und handhabbar ist. Als Wissensrepräsentationssprache für derartige Ontologien werden deshalb im Folgenden *Beschreibungslogiken* vorgestellt.

3.3.2. Beschreibungslogiken als Ontologiesprachen

Terminologische Logiken oder *Beschreibungslogiken* (engl. *Description Logics,* DL) wurden als Sprachen für die Spezifikation von Ontologien entwickelt, die sowohl menschenverständlich als auch maschinenverarbeitbar sind [BCM+03]. Ihre Anfänge finden sich in ersten, noch informalen Ansätzen zur Wissensrepräsentation wie z. B. *semantischen Netzen* [Qui67] und *Konzeptrahmen* (engl. *frames*) [Min75]. Als Alternative zur

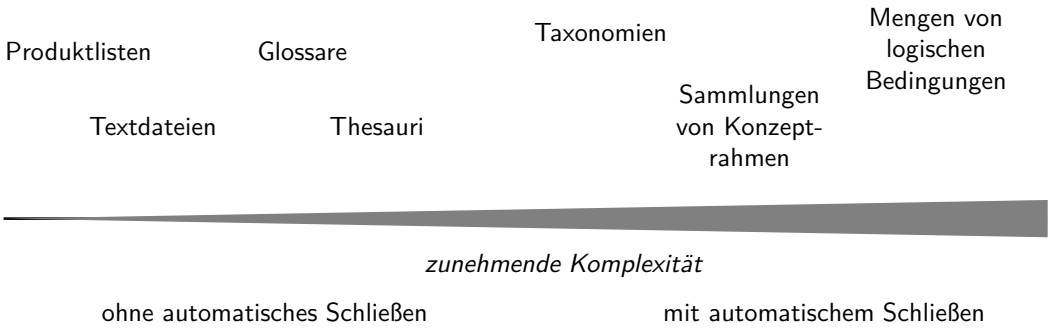

Abbildung 3.6.: Mögliche Ausprägungen formaler Ontologien geordnet nach ihrer Komplexität (nach [SW01]).

reinen Prädikatenlogik wurden diese Ansätze als intuitiver und praktisch besser anwendbar angesehen. Andererseits zeigten sich bald auch die Nachteile einer nicht präzise definierten Semantik: Unterschiedliche Systeme, die eigentlich denselben Formalismus umsetzen sollten, verhielten sich dennoch unterschiedlich [BCM+03]. Daraus entstand die Idee, die intuitiven Modellierungskonzepte von semantischen Netzen und Konzeptrahmen, wie z. B. Konzept-Hierarchien, durch die Definition einer wohldefinierten Semantik zu formalisieren.

Ein wichtiger Schritt in diese Richtung war die Erkenntnis, dass die Semantik von Konzeptrahmen auf Basis der Prädikatenlogik erster Stufe definiert werden kann. Genauer gesagt werden in Abhängigkeit der genauen Ausdrucksmöglichkeiten der Repräsentationssprache nur bestimmte *entscheidbare Fragmente* der Prädikatenlogik erster Stufe benötigt. Das führt dazu, dass logisches Schließen automatisiert werden kann, ohne dafür vollwertige Reasoner für die Prädikatenlogik erster Stufe zu benötigen, deren Vollständigkeit zudem nicht garantiert werden kann. Das war die Ausgangsbasis für die Entwicklung sogenannter *Konzeptsprachen*, aus denen später die Beschreibungslogiken entstanden. Im Zentrum stand die detaillierte Untersuchung des Zusammenhangs zwischen der Ausdrucksstärke einer Beschreibungslogik auf der einen und der Komplexität der Verfahren zur Lösung von Reasoning-Problemen auf der anderen Seite. Je nachdem, welche Modellierungskonstrukte in die Sprache aufgenommen werden, ergeben sich andere Komplexitäten. Diese werden immer als *Worst-Case*-Komplexitäten angegeben, also die Komplexität bei ungünstigster Eingabe für ein Problem. Konkrete Implementierungen der Reasoning-Verfahren zeigen jedoch, dass diese ungünstigsten Fälle in der Praxis selbst bei sehr ausdrucksstarken Beschreibungslogiken nur selten auftreten [BCM+03]. Zusammenfassend stellen Beschreibungslogiken also eine ganze *Familie* von unterschiedlichen Logiken dar, die durch die Konstruktoren, die sie zum Aufbau von Konzeptbeschreibungen anbieten, charakterisiert werden können [BCM+03, Tes01].

3.3.2.1. Ontologien als beschreibungslogische Wissensbasis

Beschreibungslogiken sind Wissensrepräsentationssprachen zur Spezifikation von Ontologien. Eine Ontologie stellt eine beschreibungslogische Wissensbasis dar. Bei einer beschreibungslogischen Wissensbasis $\mathcal{K}$ werden zwei Komponenten unterschieden: die *TBox* (*terminological box*) und die *ABox* (*assertional box*) (siehe Abbildung 3.7). Eine TBox $\mathcal{T}$ spezifiziert die Terminologie, d. h. das zur Wissensrepräsentation zur Verfügung stehende Vokabular der jeweiligen Anwendungsdomäne. Eine ABox $\mathcal{A}$ hingegen beinhaltet Zusicherungen (engl. *assertions*) über Individuen der zu repräsentierenden Welt bezüglich dieses Vokabulars.

Das Vokabular besteht aus *Konzepten*, die Mengen von *Individuen* bezeichnen, und *Rollen*, die binäre Relationen zwischen diesen Individuen angeben. Zusätzlich zu atomaren Konzeptnamen und atomaren Rollennamen ist es in allen Beschreibungslogiken möglich, komplexe Konzepte und Rollen auszudrücken. In der TBox können diesen komplexen Konzepten und Rollen Namen zugewiesen werden, um diese dann in der ABox für Zusicherungen über Individuen zu nutzen.

Zusätzlich zur Repräsentation von Wissen in der Wissensbasis bietet ein beschreibungslogisches wissensbasiertes System auch *Reasoning-Dienste* zum automatischen

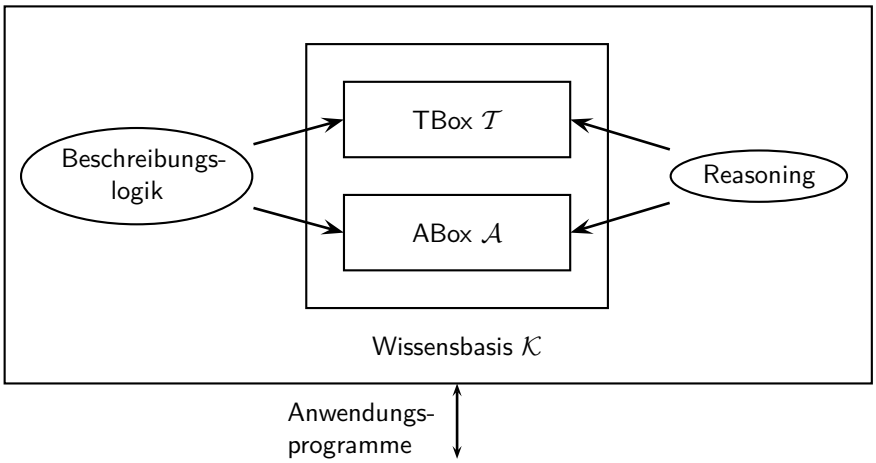

Abbildung 3.7.: Architektur eines beschreibungslogischen wissensbasierten Systems (nach [BCM+03]).

Schlussfolgern über dieses Wissen an. Diese können sich nur auf die TBox beziehen oder die ganze Wissensbasis, also TBox und ABox, betreffen. Wichtige Reasoning-Probleme für die TBox sind z. B. die *Konzept-Erfüllbarkeit*, d. h. die Entscheidung, ob eine Konzeptbeschreibung logisch widerspruchfrei ist, und der *Subsumptions-Test*, d. h. die Entscheidung, ob eine Konzeptbeschreibung allgemeiner ist als eine andere. Wichtige Reasoning-Probleme für die gesamte Wissensbasis sind die *Konsistenz-Prüfung*, d. h. die Entscheidung, ob ein Modell für die Wissensbasis existiert, und die *Instanz-Prüfung*, d. h. die Entscheidung, ob ein bestimmtes Individuum eine Instanz eines bestimmten Konzepts ist. Einige Anfragen an ein solches System können auch als komplexe Konzeptbeschreibungen ausgedrückt werden. Sie können deshalb mittels Instanz-Prüfungen beantwortet werden.

3.3.2.2. Die minimale Beschreibungslogik $\mathcal{AL}$

Beschreibungslogiken bauen auf atomaren Konzepten und atomaren Rollen auf. Komplexe Konzeptbeschreibungen werden mit Hilfe von Konzeptkonstruktoren aufgebaut. Als einfachste Beschreibungslogik und Basis aller anderen Beschreibungslogiken wird $\mathcal{AL}$ (*attributive language*) bezeichnet.

Komplexe Konzeptbeschreibungen C, D in $\mathcal{AL}$ sind syntaktisch entsprechend der folgenden Grammatik aufgebaut (sei A ein atomares Konzept):

$$
\begin{array}{lcll}
C, D & \rightarrow & A \mid & \text{(Konzeptname)} \\
 & & \top \mid & \text{(universelles Konzept)} \\
 & & \bot \mid & \text{(leeres Konzept)} \\
 & & \neg A \mid & \text{(atomare Negation)} \\
 & & C \sqcap D \mid & \text{(Schnittmenge)} \\
 & & \forall r.C \mid & \text{(Wertebeschränkung)} \\
 & & \exists r.\top & \text{(eingeschränkte Existenzquantifizierung)}
\end{array}
$$

Die Semantik dieser $\mathcal{AL}$-Konzepte wird mit Bezug auf Interpretationen definiert

(vergleiche Abschnitt 3.2.1): Eine Interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ besteht aus einer nichtleeren Grundmenge $\Delta^\mathcal{I}$ und der Interpretationsfunktion $\cdot^\mathcal{I}$. Diese weist jedem atomaren Konzeptnamen A eine Menge $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ und jedem atomaren Rollennamen r eine binäre Relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ zu. Darauf aufbauend wird $\cdot^\mathcal{I}$ für komplexe Konzepte mittels folgender rekursiver Definitionen erweitert:

$$\begin{aligned}
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
(\neg A)^\mathcal{I} &= \Delta^\mathcal{I} \setminus A^\mathcal{I} \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(\forall r.C)^\mathcal{I} &= \{a \in \Delta^\mathcal{I} \mid \forall b : (a,b) \in r^\mathcal{I} \Rightarrow b \in C^\mathcal{I}\} \\
(\exists r.\top)^\mathcal{I} &= \{a \in \Delta^\mathcal{I} \mid \exists b : (a,b) \in r^\mathcal{I}\}
\end{aligned}$$

Zwei Konzepte C, D werden als *äquivalent* bezeichnet (geschrieben $C \equiv D$), falls $C^\mathcal{I} = D^\mathcal{I}$ für alle Interpretationen $\mathcal{I}$.

3.3.2.3. Ausdrucksstarke Beschreibungslogiken

Auf Basis der minimalen Beschreibungslogik $\mathcal{AL}$ können ausdrucksstärkere Beschreibungslogiken definiert werden. Dies geschieht durch die Hinzunahme weiterer Konzept- und Rollenkonstruktoren.

Konzept- und Rollenkonstruktoren. Zur präzisen Benennung einer derartigen Beschreibungslogik schlugen Schmidt-Schauss und Smolka folgende Namenskonvention vor [SSS91]: Jeder Konstruktor wird durch einen bestimmten Großbuchstaben repräsentiert (siehe rechte Spalte in Tabelle 3.1). So wird die Unterstützung der Vereinigung in Konzeptbeschreibungen z. B. als $\mathcal{U}$ angegeben, die Unterstützung der allgemeinen Negation als $\mathcal{C}$. Diese Buchstaben werden mit dem Präfix $\mathcal{AL}$ zu dem Namen der Beschreibungslogik kombiniert, also z. B. $\mathcal{ALU}$ oder $\mathcal{ALC}$.

Eine weitere Differenzierung ist nötig für Rollenkonstruktoren: Die Erweiterung von $\mathcal{ALC}$ um transitive Rollen wird als $\mathcal{S}$ bezeichnet. Für die Unterstützung von Rolleninklusion (Rollenhierarchie) wird $\mathcal{H}$ verwendet. $\mathcal{I}$ bezeichnet inverse Rollen und $\mathcal{F}$ funktionale Rollen.

Eine derart exakte Angabe der Beschreibungslogik ist wichtig, da die Hinzunahme oder das Weglassen von Konstruktoren signifikante Auswirkungen auf die Entscheidbarkeit und Komplexität der Reasoning-Probleme haben kann.

Die Terminologie der TBox $\mathcal{T}$ wird als Menge von TBox-Axiomen definiert. Die ABox $\mathcal{A}$ ist eine Menge von ABox-Zusicherungen.

TBox-Axiome. Die TBox enthält Axiome zur Terminologie, d. h. Angaben, in welcher Beziehung komplexe Konzept- und Rollenbeschreibungen zueinander stehen (siehe Tabelle 3.2). Diese haben die Form $C \sqsubseteq D$ bzw. $r \sqsubseteq s$ oder $C \equiv D$ bzw. $r \equiv s$, wobei sich letztere durch erstere ausdrücken lassen (durch $C \sqsubseteq D, D \sqsubseteq C$). Die Semantik von $C \sqsubseteq D$ ist erwartungsgemäß definiert als $C^\mathcal{I} \subseteq D^\mathcal{I}$.

Sei TBox $\mathcal{T}$ also eine Menge von TBox-Axiomen. Dann wird $\mathcal{T}$ von einer Interpretation $\mathcal{I}$ genau dann erfüllt, wenn $\mathcal{I}$ alle Axiome in $\mathcal{T}$ erfüllt. Dann sagt man auch, $\mathcal{I}$ ist

Syntax	Semantik	Beschreibung	Symbol
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	Konzeptname	$\mathcal{AL}$
$\top$	$\Delta^{\mathcal{I}}$	universelles Konzept	$\mathcal{AL}$
$\bot$	$\emptyset$	leeres Konzept	$\mathcal{AL}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	Schnittmenge	$\mathcal{AL}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	Vereinigung	$\mathcal{U}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	allgemeine Negation	$\mathcal{C}$
$\forall r.C$	$\{\, a \in \Delta^{\mathcal{I}} \mid \forall b : (a,b) \in r^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}} \}$	Wertebeschränkung	$\mathcal{AL}$
$\exists r.C$	$\{\, a \in \Delta^{\mathcal{I}} \mid \exists b : (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \}$	Existenzquantifizierung	$\mathcal{E}$
$\geqslant_n r$ $\leqslant_n r$	$\{\, a \in \Delta^{\mathcal{I}} \mid \lvert\{b \mid (a,b) \in r^{\mathcal{I}}\}\rvert \geq n \}$ $\{\, a \in \Delta^{\mathcal{I}} \mid \lvert\{b \mid (a,b) \in r^{\mathcal{I}}\}\rvert \leq n \}$	unqualifizierte Anzahlrestriktion	$\mathcal{N}$
$\geqslant_n r.C$ $\leqslant_n r.C$	$\{\, a \in \Delta^{\mathcal{I}} \mid \lvert\{b \mid (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \geq n \}$ $\{\, a \in \Delta^{\mathcal{I}} \mid \lvert\{b \mid (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \leq n \}$	qualifizierte Anzahlrestriktion	$\mathcal{Q}$
$\{a\}$	$\{a^{\mathcal{I}}\} \subseteq \Delta^{\mathcal{I}}$	Nominal	$\mathcal{O}$
r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$	Rollenname	$\mathcal{AL}$
r^-	$\{\, (b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in r^{\mathcal{I}} \}$	inverse Rolle	$\mathcal{I}$
$\mathsf{Trans}(r)$	$(a,b) \in r^{\mathcal{I}} \wedge (b,c) \in r^{\mathcal{I}} \Rightarrow (a,c) \in r^{\mathcal{I}}$	transitive Rolle	$^+$

Tabelle 3.1.: Syntax und Semantik der Konstruktoren für komplexe Konzept- und Rollenbeschreibungen.

Syntax	Semantik	Beschreibung	Symbol
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	allgemeine Konzeptinklusion	$\mathcal{ALC}$
$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$	Rolleninklusion	$\mathcal{H}$

Tabelle 3.2.: Syntax und Semantik von TBox-Axiomen.

Syntax	Semantik	Beschreibung
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	Konzeptzugehörigkeit
$r(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	Rollenzugehörigkeit
$a \neq b$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$	Ungleichheit

Tabelle 3.3.: Syntax und Semantik von ABox-Zusicherungen.

ein Modell von $\mathcal{T}$.

ABox-Zusicherungen. Die ABox beschreibt einen bestimmten Zustand der zu repräsentierenden Welt in der Terminologie des Anwendungsgebiets. Dazu enthält sie sogenannte Zusicherungen über Individuen auf Basis der in der TBox spezifizierten Konzepte und Rollen.

Individuen werden eingeführt, indem ihnen ein Name a zugewiesen und in der ABox Eigenschaften zugeschrieben werden. Das geschieht in der Form von *Konzept-* und *Rollenzusicherungen*. Eine Konzeptzusicherung $C(a)$ gibt an, dass die Interpretation des Individuums a zu der Interpretation des Konzepts C gehört. Eine Rollenzusicherung $r(a,b)$ gibt an, dass Individuum a zu Individuum b über die Rolle r in Beziehung steht. Die Semantik dieser Axiome wird wieder über eine Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ definiert (siehe Tabelle 3.3). $\mathcal{I}$ bildet nun zusätzlich jedes Individuum a auf ein Element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ ab. Dabei wird nicht angenommen, dass Namen immer eindeutig sind (keine *unique name assumption*): Aus $a \neq b$ folgt nicht automatisch $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. Falls $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ modelliert werden soll, muss dies explizit angegeben werden. Dazu werden Zusicherungen der Form $a \neq b$ verwendet.

Sei ABox $\mathcal{A}$ also eine Menge von ABox-Zusicherungen. Dann wird $\mathcal{A}$ von einer Interpretation $\mathcal{I}$ genau dann erfüllt, wenn $\mathcal{I}$ alle Zusicherungen in $\mathcal{A}$ erfüllt. Dann wird $\mathcal{I}$ auch ein *Modell* von $\mathcal{A}$ genannt.

Wissensbasis. TBox $\mathcal{T}$ und ABox $\mathcal{A}$ bilden nun eine Wissensbasis $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. Mit $\mathsf{Ind}(\mathcal{K})$ wird die Menge aller Individuennamen bezeichnet, die in mindestens einer Zusicherung in $\mathcal{A}$ vorkommen. Eine Interpretation $\mathcal{I}$ ist ein Modell von $\mathcal{K}$ ($\mathcal{I}$ ist ein Modell von $\mathcal{A}$ bezüglich $\mathcal{T}$), wenn $\mathcal{I}$ sowohl ein Modell von $\mathcal{A}$ als auch ein Modell von $\mathcal{T}$ ist.

3.3.3. Reasoning-Probleme für Beschreibungslogiken

Neben der Repräsentation von Wissen ist das logische Schlussfolgern auf dem repräsentierten Wissen eine zentrale Zielsetzung für beschreibungslogische wissensbasierte Systeme. Analog zu einer prädikatenlogischen Wissensbasis enthält eine beschreibungslogische Wissensbasis zusätzlich zu dem explizit repräsentierten Wissen auch implizites Wissen, welches sich aus der Semantik ergibt. Mittels Reasoning-Verfahren wird dieses Wissen explizit gemacht.

3.3.3.1. Grundlegende Reasoning-Probleme

Üblicherweise werden vier grundlegende Reasoning-Probleme unterschieden:

- *Konsistenz-Prüfung*: Entscheidungsproblem, ob für eine Wissensbasis $\mathcal{K}$ (mindestens) ein nicht-leeres Modell existiert, d. h. ob $\mathcal{K} \models \top \not\equiv \bot$.
- *Konzept-Erfüllbarkeit*: Entscheidungsproblem, ob ein Modell von $\mathcal{K}$ existiert, in dem die Interpretation von Konzept C nicht leer ist, d. h. ob $\mathcal{K} \models C \not\equiv \bot$.
- *Subsumptions-Test*: Entscheidungsproblem, ob in *jedem* Modell von $\mathcal{K}$ gilt, dass die Interpretation von Konzept C in der Interpretation von Konzept D enthalten ist, d. h. ob $\mathcal{K} \models C \sqsubseteq D$.
- *Instanz-Prüfung*: Entscheidungsproblem, ob in *jedem* Modell von $\mathcal{K}$ gilt, dass die Interpretation des Individuums a in der Interpretation von Konzept C enthalten ist, d. h. ob $\mathcal{K} \models C(a)$.

Eine wichtige Erkenntnis ist, dass sich alle diese Reasoning-Probleme wie folgt auf die Konsistenz-Prüfung einer Wissensbasis zurückführen lassen. Damit können mit einem Verfahren zur Konsistenz-Prüfung auch die anderen Reasoning-Probleme gelöst werden. Mit a ein beliebiges Individuum, das nicht in $\mathcal{K}$ vorkommt, und $b \in \mathsf{Ind}(\mathcal{K})$ gilt:

Konzept-Erfüllbarkeit:	$\mathcal{K} \models C \not\equiv \bot$	$\iff$	$\mathcal{K} \cup \{C(a)\}$ ist konsistent
Subsumptions-Test:	$\mathcal{K} \models C \sqsubseteq D$	$\iff$	$\mathcal{K} \cup \{(C \sqcap \neg D)(a)\}$ nicht konsistent
Instanz-Prüfung:	$\mathcal{K} \models C(b)$	$\iff$	$\mathcal{K} \cup \{\neg C(b)\}$ ist nicht konsistent

Ziel bei der Definition einer Beschreibungslogik ist es, die Konstruktoren und damit die Ausdrucksstärke so zu wählen, dass diese grundlegenden Reasoning-Probleme entscheidbar sind. Dies ist für einfache Beschreibungslogiken wie $\mathcal{ALC}$ genauso der Fall wie für $\mathcal{SHIQ}$ und $\mathcal{SHOIN}$.

Die Komplexität der Reasoning-Probleme hängt dabei von den gewählten Konstruktoren ab. Bei der Konsistenz-Prüfung von Wissensbasen reicht sie z. B. von PSPACE-vollständig für $\mathcal{ALC}$-Wissensbasen bis zu NEXPTIME-hart für $\mathcal{SROIQ}$-Wissensbasen. Dieses Gebiet wird intensiv erforscht und neue Ergebnisse werden regelmäßig veröffentlicht. Der *Description Logic Complexity Navigator* [Zol08] bemüht sich, die aktuellsten Ergebnisse auf einer Webseite zusammenzufassen. Dabei ist zu beachten, dass die angegebenen Komplexitäten *Worst-Case*-Komplexitäten darstellen. Das heißt, dass sie im schlechtesten Fall zwar auftreten, die tatsächlichen Komplexitäten in der Praxis jedoch auch deutlich darunter liegen können.

3.3.3.2. Komplexere Reasoning-Probleme

Aufbauend auf den Verfahren zur Lösung dieser grundlegenden Reasoning-Probleme lassen sich komplexere Reasoning-Probleme, die für wissensbasierte Systeme unverzichtbar sind, definieren und lösen:

Eine *Instanz-Abfrage* liefert für ein gegebenes Konzept C und eine Wissensbasis $\mathcal{K}$ alle Individuen $a \in \mathsf{Ind}(\mathcal{K})$, für die gilt $\mathcal{K} \models C(a)$. Sie lässt sich auf die Instanz-Prüfung

zurückführen. Das duale Problem dazu ist die *Konzept-Realisierung*. Hier geht es darum, zu einem gegebenen Individuum a und einer Wissensbasis $\mathcal{K}$ das *spezifischste Konzept* $\mathsf{msc}_a^{\mathcal{K}}$ zu finden, für das gilt $\mathcal{K} \models \mathsf{msc}_a^{\mathcal{K}}(a)$. Das spezifischste Konzept ist minimal bezüglich der Konzeptinklusion $\sqsubseteq$. Dies lässt sich durch eine Kombination von Instanz-Prüfungen und Subsumptions-Tests umsetzen. Da die Konzeptinklusion nach oben durch $\top$ und nach unten durch $\bot$ beschränkt ist, muss immer ein spezifischstes Konzept existieren.

Definitionen, auf die in späteren Kapiteln Bezug genommen wird, werden im Folgenden nummeriert, um sie dort referenzieren zu können.

Definition 3.3.1 (Spezifischstes Konzept). *Sei $\mathcal{K}$ eine Wissensbasis und $a \in \mathsf{Ind}(\mathcal{K})$ ein Individuum. Eine Konzeptbeschreibung $\mathsf{msc}_a^{\mathcal{K}}$ ist das* spezifischste Konzept *(engl.* most specific concept*) von a in $\mathcal{K}$, falls für alle Konzeptbeschreibungen D gilt:*

$$\mathcal{K} \models D(a) \implies \mathcal{K} \models \mathsf{msc}_a^{\mathcal{K}} \sqsubseteq D.$$

Andere Definitionen sprechen bei der Konzept-Realisierung von a auch von mehreren spezifischsten Konzepten [BCM+03]. Beide Definitionen sind jedoch äquivalent, da alle Beschreibungslogiken den Schnittmengen-Konstruktor ($\sqcap$) unterstützen und sich diese spezifischsten Konzepte somit immer zu einem einzigen spezifischsten Konzept konjugieren lassen.

Für Rollen werden analog folgende Reasoning-Probleme definiert: Eine *Rollen-Prüfung* $r(a, b)$ entscheidet, ob in *jedem* Modell von $\mathcal{K}$ gilt, dass das Paar der Interpretationen der Individuen a und b in der Interpretation von Rolle r enthalten ist, d. h. ob $\mathcal{K} \models r(a, b)$. Eine *Rollen-Abfrage* liefert für eine gegebene Rolle r und eine Wissensbasis $\mathcal{K}$ alle Paare von Individuen $(a, b) \in \mathsf{Ind}(\mathcal{K}) \times \mathsf{Ind}(\mathcal{K})$, für die gilt $\mathcal{K} \models r(a, b)$. Dabei können a, b oder beide auch gegeben sein. Wenn a und b gegeben sind, lässt sich auch das duale Problem der *Rollen-Realisierung* definieren: Gegeben (a, b) und eine Wissensbasis $\mathcal{K}$ wird die *spezifischste Rolle* $\mathsf{msr}_{(a,b)}^{\mathcal{K}}$ gesucht, für die gilt $\mathcal{K} \models \mathsf{msr}_{(a,b)}^{\mathcal{K}}(a, b)$. Die spezifischste Rolle ist minimal bezüglich der Rolleninklusion $\sqsubseteq$.

Definition 3.3.2 (Spezifischste Rolle). *Seien $\mathcal{K}$ eine Wissensbasis und $a, b \in \mathsf{Ind}(\mathcal{K})$ zwei Individuen. Eine Rolle $\mathsf{msr}_{(a,b)}^{\mathcal{K}}$ ist die* spezifischste Rolle *(engl.* most specific role*) zu (a, b) in $\mathcal{K}$, falls für alle Rollen p gilt:*

$$\mathcal{K} \models p(a, b) \implies \mathcal{K} \models \mathsf{msr}_{(a,b)}^{\mathcal{K}} \sqsubseteq p.$$

Je nach Beschreibungslogik werden andere Reasoning-Verfahren benötigt, um diese Reasoning-Probleme korrekt und vollständig zu lösen.

3.3.3.3. Konjunktive Anfragen

Auch Verfahren zur Lösung der oben genannten komplexeren Reasoning-Probleme sind für reale Anwendungen jedoch in der Regel nicht ausreichend: Mittels Instanz-Abfragen können zwar für beliebig komplexe Konzeptbeschreibungen alle zugehörigen Individuen ermittelt werden. Diese Anfragemöglichkeit ist jedoch eingeschränkt, da Konzeptbeschreibungen keine beliebigen zyklischen Strukturen erlauben und deshalb nur *baumähnliche* relationale Strukturen abgefragt werden können.

Konjunktive Anfragen erweitern diese Möglichkeiten, indem mehrere dieser Reasoning-Probleme als Bedingungen an die gesuchten Lösungen konjugiert werden können. Sie sind auf dem Gebiet der relationalen Datenbanken wohl bekannt. Im Folgenden werden ihre Syntax und Semantik mit Bezug zu beschreibungslogischen Wissensbasen definiert.

Konjunktive Anfragen sind Konjunktionen von *Anfrageatomen*. Zur Vereinfachung der Notation werden sie als *Menge* von Anfrageatomen geschrieben, die implizit konjugiert sind. Eine konjunktive Anfrage mit den Variablen x und y könnte also beispielsweise folgendermaßen lauten:

$$\{\mathsf{Achslager}(x), \mathsf{Drehgestell}(y), \mathsf{teilVon}(x, y), \mathsf{Fehlerhaft}(x)\}.$$

Bei Variablen wird zwischen *benannten* (engl. *distinguished*) und *unbenannten* (engl. *non-distinguished*) Variablen unterschieden. In einer Lösung für eine konjunktive Anfrage müssen alle benannten Variablen an benannte Individuen $a \in N_I$ gebunden sein, während unbenannte Variablen auch an anonyme Individuen gebunden werden können, von denen lediglich gezeigt werden muss, dass sie existieren. Sind alle Variablen unbenannt, handelt es sich um eine *boole'sche konjunktive Anfrage*, deren Lösung entweder *wahr* oder *falsch* ist. Gibt es benannte Variablen, geht es um eine (nicht-boole'sche) *konjunktive Anfrage*, deren Lösungen die Tupel der Individuennamen sind, die an die benannten Variablen gebunden werden. Dies kann formal wie folgt formuliert werden (vergleiche auch [GHLS08]).

Definition 3.3.3 (Konjunktive Anfrage Q). *Seien N_C, N_R und N_I abzählbar unendliche Mengen von Konzept-, Rollen- und Individuennamen. Sei N_V eine abzählbar unendliche Menge von* Variablennamen *disjunkt zu N_C, N_R und N_I. Ein* Term *t ist ein Element aus $N_V \cup N_I$. Seien C eine Konzeptbeschreibung, r eine Rollenbeschreibung und t, t' Terme. Ein* Anfrageatom *ist ein Ausdruck $C(t)$ (*Konzept-Anfrageatom*) oder $r(t, t')$ (*Rollen-Anfrageatom*). Eine* konjunktive Anfrage *Q ist eine nicht-leere Menge von Anfrageatomen. $\mathsf{Var}(Q) \subseteq N_V$ ist die Menge aller Variablennamen, die in Q vorkommen; $\mathsf{Ind}(Q) \subseteq N_I$ die Menge aller Individuennamen, die in Q vorkommen; und $\mathsf{Terms}(Q) = \mathsf{Var}(Q) \cup \mathsf{Ind}(Q)$ bezeichnet die Menge aller Terme in Q. $|Q|$ ist die Zahl der Anfrageatome in Q. Eine* Unteranfrage von *Q ist eine Teilmenge von Q.*

Bezogen auf eine Interpretation ist die Erfüllung einer konjunktiven Anfrage wie folgt definiert.

Definition 3.3.4 (Erfülltheit einer konjunktiven Anfrage Q). *Sei $\mathcal{I}$ eine Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. Eine totale Funktion $\mu : \mathsf{Terms}(Q) \to \Delta^{\mathcal{I}}$ ist eine* Zuordnung*, falls gilt $\mu(a) = a^{\mathcal{I}}$. Dann schreibt man:*

- *$\mathcal{I} \models_\mu C(t)$ falls $\mu(t) \in C^{\mathcal{I}}$;*
- *$\mathcal{I} \models_\mu r(t, t')$ falls $(\mu(t), \mu(t')) \in r^{\mathcal{I}}$.*

Gilt für alle Anfrageatome $q \in Q$: $\mathcal{I} \models_\mu q$, dann folgt $\mathcal{I} \models_\mu Q$. Existiert eine Zuordnung μ, so dass $\mathcal{I} \models_\mu Q$, sagt man, $\mathcal{I}$ erfüllt *Q (geschrieben $\mathcal{I} \models Q$).*

Damit lassen sich die logische Folgerung und die Lösungen einer konjunktiven Anfrage mit Bezug zu einer Wissensbasis definieren.

Definition 3.3.5 (Logische Folge einer konjunktiven Anfrage Q). *Sei $\mathcal{K}$ eine Wissensbasis und Q eine konjunktive Anfrage. Dann folgt Q logisch aus $\mathcal{K}$ (geschrieben $\mathcal{K} \models Q$), falls $\mathcal{I} \models \mathcal{K}$ impliziert $\mathcal{I} \models Q$.*

Definition 3.3.6 (Lösungen einer konjunktiven Anfrage Q). *Sei $\mathcal{K}$ eine Wissensbasis und Q eine konjunktive Anfrage mit n Variablen $v_1, \ldots, v_n$, d. h. $|\mathsf{Var}(Q)| = n$. Seien die Variablen v_i für $1 \leq i \leq m \leq n$ benannte Variablen und sonst unbenannte Variablen.*

Die Lösungen *$M_{Q,\mathcal{K}}$ für Q bezüglich $\mathcal{K}$ sind alle Tupel $m_{Q,\mathcal{K}} = (a_1, \ldots, a_m) \in N_I^m$, für die es für alle Modelle $\mathcal{I} \models \mathcal{K}$ eine Zuordnung $\mu(v_i) = {a_i}^{\mathcal{I}}, 1 \leq i \leq m$, gibt, so dass $\mathcal{I} \models_\mu Q$.*

3.4. Beschreibungslogik $\mathcal{SHIN}$ als eine Grundlage der Web Ontology Language

Die im Folgenden beschriebenen Beschreibungslogiken bilden die Grundlage für die *Semantic Web*-Standards des *World Wide Web Consortiums* (W3C) und damit auch dieser Arbeit.

3.4.1. Web Ontology Language OWL

Die *Web Ontology Language* (OWL) ist die standardisierte Ontologiesprache des *World Wide Web Consortium* (W3C). Sie wurde 2004 als W3C *Recommendation* veröffentlicht [BHH+04].

OWL spielt eine wichtige Rolle in der geschichteten Architektur (auch „layer cake“ oder „Schichttorte“ genannt) des *Semantic Web*, die in Abbildung 3.8 dargestellt ist. Der Begriff des Semantic Web wurde bekannt durch den vielzitierten Artikel „The Semantic Web“ von Tim Berners-Lee et al. [BLHL01]. Um die zunehmende Informationsflut des *World Wide Web* (WWW) bewältigen zu können, wird dort die Vision eines semantisch erweiterten WWW entwickelt. Kern des Ansatzes ist es, die bisher nur für Menschen verarbeitbaren Informationen in und über Web-Dokumente und -Dienste zusätzlich mit formalisierter Semantik zu versehen und dadurch maschinenverarbeitbar zu machen.

Das ist natürlich ein klassisches Ziel der Wissensrepräsentation, und formale Ontologien spielen dafür eine zentrale Rolle. Dementsprechend wurde OWL auf Basis etablierter Beschreibungslogiken spezifiziert, die attraktive Kompromisse zwischen Ausdrucksstärke und Komplexität bieten. Innerhalb der OWL wird dabei noch einmal bezüglich der Ausdrucksstärke (und folglich der Reasoning-Komplexität) differenziert, um unterschiedliche Anwendungsszenarien unterstützen zu können. Es werden drei Varianten der OWL unterschieden:

- OWL *lite*: Basiert im Wesentlichen auf der Beschreibungslogik $\mathcal{SHIF}$ und ist die ausdrucksschwächste Variante. Hier sind sowohl Konsistenz-Prüfung als auch Konzept-Erfüllbarkeit ExpTime-vollständig.

- OWL DL: Basiert im Wesentlichen auf der Beschreibungslogik $\mathcal{SHOIN}$ und repräsentiert damit eine sehr ausdrucksstarke Beschreibungslogik, die entscheidbar ist. Hier sind sowohl Konsistenz-Prüfung als auch Konzept-Erfüllbarkeit NExpTime-vollständig.

- OWL *full*: Erlaubt sehr große Freiheiten, wie z. B. die Instanziierung einer Instanz, und ist deshalb nicht durch eine der bekannten Beschreibungslogiken abbildbar. OWL full ist deshalb nicht entscheidbar.

Die genannten Komplexitäten sind *kombinierte Komplexitäten*, d. h. sie beziehen sich auf die Größe von ABox und TBox zusammen. Bei vielen (wie auch den in dieser Arbeit betrachteten) Anwendungen, ist die TBox jedoch unveränderlich und steht einer veränderlichen und tendenziell sehr viel größeren ABox gegenüber. In diesen Fällen liefert die *Datenkomplexität*, d. h. die Komplexität bezüglich der Größe der ABox, eine wesentlich präzisere Abschätzung für das praktische Verhalten des Reasonings bei diesen Anwendungen. Für die Beschreibungslogik $\mathcal{SHIQ}$ wurde zum Beispiel gezeigt, dass die Konsistenz-Prüfung bezüglich der ABox NP-vollständig und die Instanz-Prüfung coNP-vollständig ist [HMS05].

Diese Arbeit soll auf dem OWL-Standard aufsetzen, um die Zukunftssicherheit zu gewährleisten und bei der praktischen Umsetzung von vielen verfügbaren Werkzeugen und Software-Komponenten profitieren zu können. Dabei ist es essenziell, dass die verwendete Wissensrepräsentationssprache entscheidbar ist. Außerdem werden mächtige Ausdrucksmöglichkeiten benötigt, um die komplexen Zusammenhänge bei Infrastrukturnetzen modellieren zu können. Darum wird für diese Arbeit die OWL DL-Variante gewählt. Auf die Verwendung von Nominalen wird jedoch bewusst verzichtet, weil dadurch die (kombinierten) Komplexitäten für Konsistenz-Prüfung und Konzept-Erfüllbarkeit beide auf ExpTime beschränkt werden können [Zol08]. Grundlage für diese Arbeit ist also die Beschreibungslogik $\mathcal{SHIN}$. Diese wird im Folgenden kurz eingeführt und ein Entscheidungsverfahren für die Konsistenz-Prüfung von $\mathcal{SHIN}$-Wissensbasen vorgestellt.

3.4.2. Syntax und Semantik von $\mathcal{SHIN}$

Syntax und Semantik von $\mathcal{SHIN}$ sind wie folgt definiert (vergleiche z. B. [HST00a]). Seien N_C, N_R und N_I abzählbar unendliche Mengen von *Konzept-*, *Rollen-* und *Individuennamen*. Als Untermenge enthält die Menge der Rollennamen zudem *transitive Rollennamen* $N_{\mathsf{Trans}(R)} \subseteq N_R$.

Eine *Rolle* ist ein Element von $N_R \cup \{r^- \mid r \in N_R\}$, wobei Rollen der Form r^- als *inverse Rollen* bezeichnet werden. Um keine Rollen der Form r^{--} berücksichtigen zu müssen, wird eine Funktion $\mathsf{Inv}()$ definiert als $\mathsf{Inv}(r) = r^-$ und $\mathsf{Inv}(r^-) = r$ mit $r \in N_R$. Außerdem wird eine Funktion $\mathsf{Trans}()$ eingeführt mit $\mathsf{Trans}(r) = \mathit{true}$ genau dann, wenn $r \in N_{\mathsf{Trans}(R)}$ oder $\mathsf{Inv}(r) \in N_{\mathsf{Trans}(R)}$. Eine Rolle s wird *einfache Rolle* bezüglich einer Rollen-Hierarchie genannt, wenn sie weder transitiv ist noch transitive Unter-Rollen hat.

Komplexe $\mathcal{SHIN}$-Konzeptbeschreibungen C, D sind mit $A \in N_C$, r eine Rolle und s eine einfache Rolle rekursiv wie folgt definiert:

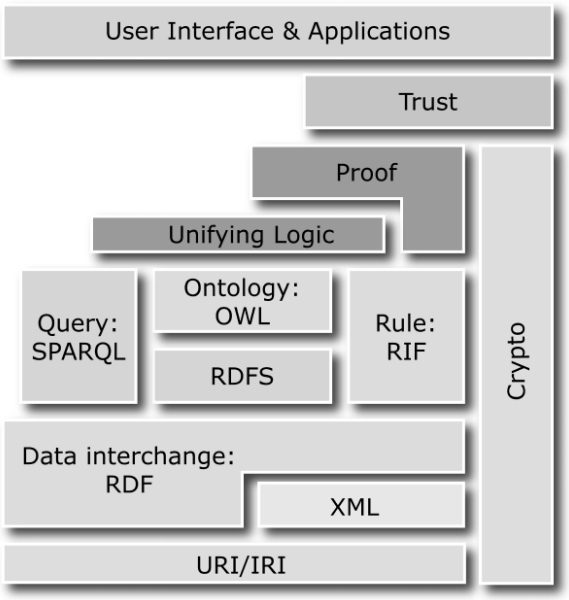

Abbildung 3.8.: „Schichttorte“ des Semantic Web [SW08].

$$
\begin{array}{lll}
C, D & \rightarrow & A \mid \quad \text{(Konzeptname)} \\
 & & \top \mid \quad \text{(universelles Konzept)} \\
 & & \bot \mid \quad \text{(leeres Konzept)} \\
 & & C \sqcap D \mid \quad \text{(Schnittmenge)} \\
 & & C \sqcup D \mid \quad \text{(Vereinigung)} \\
 & & \neg C \mid \quad \text{(allgemeine Negation)} \\
 & & \forall r.C \mid \quad \text{(Wertebeschränkung)} \\
 & & \exists r.C \mid \quad \text{(Existenzquantifizierung)} \\
 & & \leqslant_n s.\top \mid \quad \text{(obere Anzahlrestriktion)} \\
 & & \geqslant_n s.\top \quad \text{(untere Anzahlrestriktion).}
\end{array}
$$

Eine *Rolleninklusion* ist von der Form $r \sqsubseteq p$ mit r, p Rollen. Eine *Rollen-Hierarchie* ist eine endliche Menge von Rolleninklusionen. Eine *allgemeine Konzeptinklusion* hat die Form $C \sqsubseteq D$ mit Konzeptbeschreibungen C und D. Eine *TBox* $\mathcal{T}$ ist eine endliche Menge von Rollen- und Konzeptinklusionen. Eine *ABox* $\mathcal{A}$ ist eine endliche Menge von Zusicherungen der Form $C(a), r(a, b)$ und $a \neq b$ mit einer Konzeptbeschreibung C, einer Rolle r und Individuen a, b.

Die Semantik der $\mathcal{SHIN}$-Konzeptbeschreibungen, TBox-Axiome und ABox-Zusicherungen ist entsprechend der Tabellen 3.1, 3.2 und 3.3 definiert. Eine Interpretation $\mathcal{I}$ ist ein *Modell* von $\mathcal{T}$, falls sie jedes Axiom in $\mathcal{T}$ erfüllt. Eine Interpretation $\mathcal{I}$ ist ein *Modell* von $\mathcal{A}$, falls sie jede Zusicherung in $\mathcal{A}$ erfüllt. Schließlich ist eine Interpretation $\mathcal{I}$ ein *Modell* der $\mathcal{SHIN}$-Wissensbasis $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, falls sie ein Modell von $\mathcal{T}$ und $\mathcal{A}$ ist. $\mathcal{K}$ wird als *konsistent* bezeichnet, falls $\mathcal{K}$ ein Modell hat.

Zur Vereinfachung von Beweisen wird in späteren Kapitel die *Negation Normal Form* (*NNF*) als Normalform für komplexe Konzeptbeschreibungen angenommen.

Definition 3.4.1 (Negation Normal Form (NNF)). *Sei C eine Konzeptbeschreibung. Eine Konzeptbeschreibung ist in* Negation Normal Form (NNF)*, wenn der $\neg$-Konstruktor nur vor Konzeptnamen erscheint. Jedes C kann mit den folgenden Regeln in eine äquivalente Konzeptbeschreibung $\mathsf{NNF}(C)$ in Negation Normal Form überführt werden:*

$$\neg\neg C \equiv C \qquad \neg(C \sqcap D) \equiv \neg C \sqcap \neg D \qquad \neg\exists\, r.C \equiv \forall\, r.\neg C$$
$$\neg(C \sqcup D) \equiv \neg C \sqcap \neg D \qquad \neg\forall\, r.C \equiv \exists\, r.\neg C$$

3.4.3. Tableaux-Algorithmus zur Konsistenz-Prüfung

Um für eine gegebene $\mathcal{SHIN}$-Wissensbasis entscheiden zu können, ob sie konsistent ist, wird eine Inferenzprozedur benötigt. Mit einem derartigen Verfahren ist es dann außerdem möglich, die anderen grundlegenden beschreibungslogischen Reasoning-Probleme wie z. B. Konzept-Erfüllbarkeit und Instanz-Prüfung zu lösen. Wie in Abschnitt 3.3.3 gezeigt wurde, lassen sich diese auf die Konsistenz-Prüfung einer Wissensbasis zurückführen.

Dafür gibt es unterschiedliche Ansätze. Sehr verbreitet sind tableaux-basierte Verfahren. Aber auch resolutions-basierte Verfahren oder deduktives Datalog wurden vorgeschlagen. Tableaux-Verfahren haben sich in der Praxis besonders bewährt und sind auch die Grundlage erfolgreicher Implementierungen von Reasoner-Komponenten wie Pellet [SPG+07], RacerPro [HM03] und Fact++ [TH06]. Da die Beweise in Kapitel 5 ebenfalls auf dem Tableaux-Verfahren aufbauen, wird im Folgenden ein Tableaux-Algorithmus zur Konsistenz-Prüfung von $\mathcal{SHIN}$-Wissensbasen vorgestellt.

Die Grundidee eines Tableaux-Algorithmus ist folgende: Sei eine Wissensbasis $\mathcal{K}$, bestehend aus TBox $\mathcal{T}$ und ABox $\mathcal{A}$, gegeben. Dann wird versucht, ein konkretes Beispiel zu konstruieren für eine Interpretation, die alle TBox-Axiome in $\mathcal{T}$ und alle ABox-Zusicherungen in $\mathcal{A}$ erfüllt. Diese Interpretation repräsentiert also ein *Modell* für $\mathcal{K}$, womit nachgewiesen wird, dass ein solches existiert. Gelingt dies, folgt daraus, dass $\mathcal{K}$ konsistent ist.

Zur Konstruktion einer Interpretation wird von den expliziten Zusicherungen in $\mathcal{A}$ ausgegangen. Diese bilden einen initialen *Komplettierungswald*. Dann werden unter Berücksichtigung der ABox-Zusicherungen und TBox-Axiome in $\mathcal{T}$ bestimmte *Expansions-Regeln* angewandt, um erweiterte Komplettierungswälder abzuleiten, in denen zusätzliches implizites Wissen explizit gemacht wird. Abbildung 3.9 illustriert die prinzipielle Struktur eines Komplettierungswalds: Er besteht aus einem – nicht notwendigerweise zusammenhängenden – gerichteten Graph mit den in $\mathcal{A}$ erwähnten Individuen $\mathsf{Ind}(\mathcal{K})$ als Knoten und den zugesicherten Rollen als Kanten. Die benannten Knoten können als Wurzelknoten für Bäume aus *anonymen Individuen* dienen, die implizit existieren müssen – beispielsweise aufgrund einer Konzeptzugehörigkeit $(\exists r.C)(a)$ —, denen in $\mathcal{A}$ jedoch kein Name zugeordnet ist. Untereinander sind diese Wurzelknoten entsprechend der Rollen-Zusicherungen in $\mathcal{A}$ verbunden. Expansions-Regeln sind abhängig von der Beschreibungslogik und beziehen sich in der Regel auf die unterstützten Konzeptkonstruktoren. Einige von ihnen sind *nicht-deterministisch*, weshalb Fallunterscheidungen getroffen werden müssen. Dies geschieht solange, bis entweder ein logischer Widerspruch im Komplettierungswald erkannt wird (z. B. dass $A(a)$ und $\neg A(a)$ gleichzeitig gelten sollen) oder keine weiteren Regeln mehr auf ihn

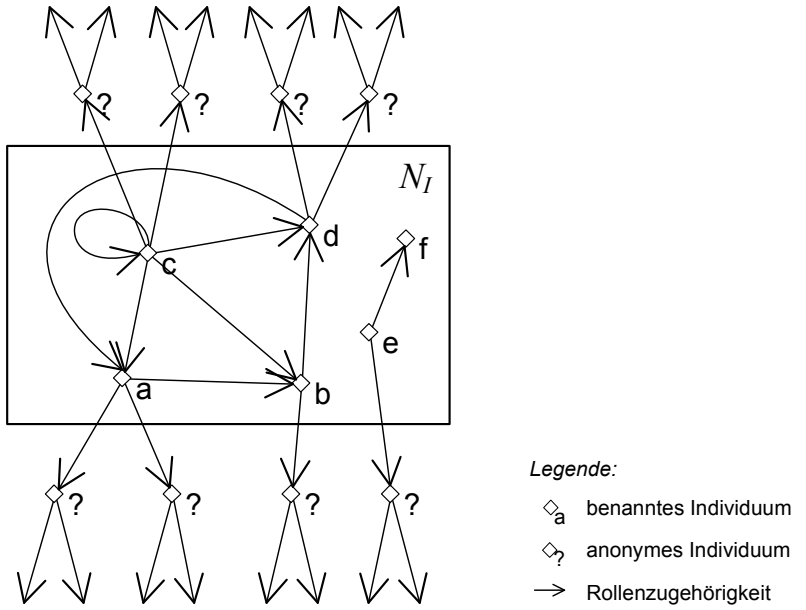

Abbildung 3.9.: Prinzipielle Struktur des Komplettierungswalds zu einer Wissensbasis.

anwendbar sind. Dazu muss die Terminierung sichergestellt sein. Für die Unterstützung transitiver Rollen wird deshalb zwischen blockierten Knoten, die nicht weiter expandiert werden, und nicht blockierten Knoten unterschieden. In Kombination mit inversen Rollen werden diese Blockierungsbedingungen weiter verfeinert (siehe Definition 3.4.3).

Sind keine Regeln mehr anwendbar und ist kein Widerspruch aufgetreten, dann ist der Komplettierungswald *widerspruchsfrei* und *vollständig*. In diesem Fall repräsentiert er ein konkretes Beispiel für ein Modell von $\mathcal{K}$, womit nachgewiesen wird, dass $\mathcal{K}$ konsistent ist. Da gezeigt werden kann, dass der Tableaux-Algorithmus genau für konsistente Wissensbasen einen widerspruchsfreien und vollständigen Komplettierungswald konstruiert, gilt anderenfalls, dass $\mathcal{K}$ inkonsistent ist [HST00b].

Als formale Grundlage für die Beweise in Kapitel 5 wird im Folgenden der Tableaux-Algorithmus T für $\mathcal{SHIN}$-Wissensbasen in Anlehnung an [HST00b] definiert. T hat als Eingabe eine TBox $\mathcal{T}$, die lediglich *Rollen*inklusionen (also keine Konzeptinklusionen) enthält, und eine ABox $\mathcal{A}$. Als Ausgabe hat er die Entscheidung, ob $\mathcal{A}$ bezüglich $\mathcal{T}$ konsistent ist. Eine beliebige Wissensbasis $\mathcal{K} = (\mathcal{T}_0, \mathcal{A}_0)$ kann in ein entsprechendes Paar $(\mathcal{T}, \mathcal{A})$ umgeformt werden, indem die Konzeptinklusionen aus $\mathcal{T}_0$ nach $\mathcal{A}_0$ *internalisiert* werden und daraus eine ABox $\mathcal{A}$ entsteht (für das genaue Verfahren siehe [HST00b]). $\mathcal{T}$ enthält dann nur noch die Rolleninklusionen aus $\mathcal{T}_0$. Diese Umformung kann in polynomieller Zeit durchgeführt werden. O. b. d. A. wird außerdem angenommen, dass alle Konzeptbeschreibungen in $\mathcal{A}$ in NNF (siehe Definition 3.4.1) sind.

Als Grundlage für den Tableaux-Algorithmus muss der *Komplettierungswald* zunächst formal definiert werden.

Definition 3.4.2 (Komplettierungswald $\mathcal{F}$). *Ein Komplettierungswald $\mathcal{F}$ ist eine Sammlung von Bäumen, deren Wurzelknoten benannt sind und die mittels gerichteter Kanten beliebig miteinander verbunden sein können. Jeder Knoten x ist mit einer Menge von Konzeptbeschreibungen $\mathcal{L}(x)$, jede Kante (x, y) mit einer Menge von Rollenbeschreibungen $\mathcal{L}(x, y)$ beschriftet. Außerdem gibt es eine symmetrische Gleichheits-Relation $\doteq$*

und eine symmetrische Ungleichheits-Relation $\neq$ über Knotenpaare.

*Für zwei Knoten x, y heißt y s-*Nachfolger *von x (und y $\mathsf{Inv}(s)$-*Vorgänger *von x), wenn es eine Kante (x, y) gibt mit $r \in \mathcal{L}(x, y)$ und $r \sqsubseteq^* s$, wobei $\sqsubseteq^*$ die transitive Hülle der Rolleninklusion $\sqsubseteq$ ist. y ist s-*Nachbar *von x, wenn y s-*Nachfolger *von x oder y $\mathsf{Inv}(s)$-*Vorgänger *von x ist. y ist ein* Nachfolger*,* Nachbar *oder* Vorfahre *von x, wenn y ein s-*Nachfolger*, s-*Nachbar *oder s-*Vorfahre *von x für eine Rolle s ist.* Vorfahre *ist die transitive Hülle von* Vorgänger*.*

Ein Knoten ist genau dann blockiert*, wenn er kein Wurzelknoten ist und entweder direkt oder indirekt blockiert ist. Ein Knoten x ist genau dann* direkt blockiert*, wenn keiner seiner Vorfahren blockiert ist und er Vorfahren x', y und y' besitzt, so dass*

1. *y kein Wurzelknoten eines aus anonymen Knoten bestehenden Baumes ist und*
2. *x ein Nachfolger von x' und y ein Nachfolger von y' ist und*
3. *$\mathcal{L}(x) = \mathcal{L}(y)$ und $\mathcal{L}(x') = \mathcal{L}(y')$ und*
4. *$\mathcal{L}((x', x)) = \mathcal{L}((y', y))$.*

In diesem Fall sagt man, y blockiert x.

Ein Knoten y ist genau dann indirekt blockiert*, wenn einer seiner Vorfahren blockiert ist oder er ein Nachfolger eines Knotens x mit $\mathcal{L}((x, y)) = \emptyset$ ist (dadurch werden unnötige Expansionen nach Anwendung der $\leqslant$-Regel vermieden).*

Für eine gegebene ABox $\mathcal{A}$ und eine gegebene TBox $\mathcal{T}$, die nur Rolleninklusions-Axiome enthält, versucht der Tableaux-Algorithmus T einen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}$ zu konstruieren.

Definition 3.4.3 (Tableaux-Algorithmus T für $\mathcal{SHIN}$-Wissensbasen). *Der Tableaux-Algorithmus T initialisiert einen Komplettierungswald $\mathcal{F}_0 = (N, E, \mathcal{L}_0)$, der nur aus Wurzelknoten besteht. Genauer gesagt enthält N für jedes Individuum $a \in \mathsf{Ind}(\mathcal{K})$ einen Knoten $n(a)$ und E für jede Rollenzusicherung $r(a, b) \in \mathcal{K}$ ein Paar $(n(a), n(b))$. Die Beschriftungen dieser Knoten und Kanten sowie die Relationen $\doteq$ und $\not\doteq$ werden wie folgt initialisiert:*

$$\begin{aligned} \mathcal{L}_0(n(a)) &:= \{\, C \mid C(a) \in \mathcal{A} \,\} \\ \mathcal{L}_0(n(a), n(b)) &:= \{\, r \mid r(a, b) \in \mathcal{A} \,\} \\ n(a) \not\doteq n(b) &\Leftrightarrow a \neq b \in \mathcal{K} \\ \doteq &:= \emptyset \end{aligned}$$

$\mathcal{F}_0$ wird nun expandiert, indem iterativ die Expansions-Regeln in Tabelle 3.4 angewandt werden.

Sei x ein Knoten. Dann enthält $\mathcal{L}(x)$ einen Widerspruch*, falls*

1. *$\{A, \neg A\} \subseteq \mathcal{L}(x)$ für einen Konzeptnamen $A \in N_C$,*
2. *$\leqslant_n s.\top \in \mathcal{L}(x)$ und x $n+1$ s-Nachbarn $y_0, \ldots, y_n$ mit $y_i \not\doteq y_j$ für alle $0 \leq i < j \leq n$ hat.*

$\sqcap$-Regel:	falls	1. $(C_1 \sqcap C_2) \in \mathcal{L}(x)$, x ist nicht indirekt blockiert, und
		2. $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	dann	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{C_1, C_2\}$
$\sqcup$-Regel:	falls	1. $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x ist nicht indirekt blockiert, und
		$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	dann	$\mathcal{L}(x) \longrightarrow \mathcal{L}(x) \cup \{E\}$ für ein $E \in \{C_1, C_2\}$
$\exists$-Regel:	falls	1. $(\exists\, s.C) \in \mathcal{L}(x)$, x ist nicht blockiert, und
		2. x hat keinen s-Nachbarn y mit $C \in \mathcal{L}(y)$
	dann	erzeuge einen neuen Knoten y mit $\mathcal{L}((x,y)) := \{s\}$, $\mathcal{L}(y) := \{C\}$
$\forall$-Regel:	falls	1. $(\forall\, s.C) \in \mathcal{L}(x)$, x ist nicht indirekt blockiert, und
		2. es gibt einen s-Nachbarn y von x mit $C \notin \mathcal{L}(y)$
	dann	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{C\}$
$\forall_+$-Regel:	falls	1. $(\forall\, s.C) \in \mathcal{L}(x)$, x ist nicht indirekt blockiert, und
		2. es gibt ein r mit $\mathsf{Trans}(r)$ und $r \sqsubseteq^* s$,
		3. es gibt einen r-Nachbarn y von x mit $\forall r.C \notin \mathcal{L}(y)$
	dann	$\mathcal{L}(y) \longrightarrow \mathcal{L}(y) \cup \{\forall r.C\}$
$\geqslant$-Regel:	falls	1. $(\geqslant_n\ s) \in \mathcal{L}(x)$, x ist nicht blockiert, und
		2. es gibt keine n s-Nachbarn $y_1, \ldots, y_n$ mit $y_i \neq y_j$, $1 \leq i < j \leq n$
	dann	erzeuge n neue Knoten $y_1, \ldots, y_n$ mit $\mathcal{L}((x,y_i)) := \{s\}$
		und $y_i \neq y_j$ für $1 \leq i < j \leq n$
$\leqslant$-Regel:	falls	1. $(\leqslant_n\ s) \in \mathcal{L}(x)$, x ist nicht indirekt blockiert, und
		2. x hat $n+1$ s-Nachbarn $y_1, \ldots, y_{n+1}$
		3. davon sind y, z zwei s-Nachbarn von x, ohne dass $y \neq z$
		4. y ist weder ein Wurzelknoten noch ein Vorfahre von z
	dann	1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ und
		2. falls z ein Vorfahre von x ist
		dann $\mathcal{L}((z,x)) \longrightarrow \mathcal{L}((z,x)) \cup \mathsf{Inv}(\mathcal{L}((x,y)))$
		sonst $\mathcal{L}((x,z)) \longrightarrow \mathcal{L}((x,z)) \cup \mathcal{L}((x,y))$
		3. $\mathcal{L}((x,y)) \longrightarrow \emptyset$
		4. setze $u \neq z$ für alle u mit $u \neq y$
$\leqslant_s$-Regel:	falls	1. $(\leqslant_n\ s) \in \mathcal{L}(x)$ und
		2. x hat $n+1$ s-Nachbarn $y_1, \ldots, y_{n+1}$
		3. davon sind y, z zwei s-Nachbarn von x, ohne dass $y \neq z$
		4. y und z sind beide Wurzelknoten
	dann	1. $\mathcal{L}(z) \longrightarrow \mathcal{L}(z) \cup \mathcal{L}(y)$ und
		2. für alle Kanten (y,w) :
		i. falls (z,w) nicht existiert, erzeuge sie mit $\mathcal{L}((z,w)) = \emptyset$
		ii. $\mathcal{L}((z,w)) \longrightarrow \mathcal{L}((z,w)) \cup \mathcal{L}((y,w))$
		3. für alle Kanten (w,y) :
		i. falls (w,z) nicht existiert, erzeuge sie mit $\mathcal{L}((w,z)) = \emptyset$
		ii. $\mathcal{L}((w,z)) \longrightarrow \mathcal{L}((w,z)) \cup \mathcal{L}((w,y))$
		4. setze $\mathcal{L}(y) = \emptyset$ und entferne alle Kanten nach/von y
		5. setze $u \neq z$ für alle u mit $u \neq y$ and setze $y \doteq z$

Tabelle 3.4.: Expansionsregeln für den Tableaux-Algorithmus für $\mathcal{SHIN}$-Wissensbasen [HST00b].

Ein Komplettierungswald $\mathcal{F}$ ist widerspruchsfrei*, wenn keine Beschriftung eines Knotens einen Widerspruch enthält. Ein Komplettierungswald ist* vollständig*, wenn keine Expansions-Regel aus Tabelle 3.4 mehr anwendbar ist.*

T startet mit dem Komplettierungswald $\mathcal{F}_0$ und expandiert ihn durch iterative Anwendung der Expansionsregeln in Tabelle 3.4, bis ein Widerspruch entsteht. Falls ein widerspruchsfreier und vollständiger Komplettierungswald konstruiert werden kann, antwortet T, dass $\mathcal{A}$ bezüglich $\mathcal{T}$ konsistent ist, und damit, dass $\mathcal{K}$ konsistent ist. Ansonsten antwortet T, dass $\mathcal{K}$ nicht konsistent ist.

Theorem 3.4.1 (Konsistenz einer $\mathcal{SHIN}$-Wissensbasis [HST00b]). *Sei $\mathcal{K}$ eine $\mathcal{SHIN}$-Wissensbasis und T der Tableaux-Algorithmus für $\mathcal{SHIN}$. T konstruiert genau dann einen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}$ für $\mathcal{K}$, wenn $\mathcal{K}$ konsistent ist.*

3.5. Zusammenfassung

Dieses Kapitel führte kurz in das Gebiet der formalen Wissensrepräsentation und Reasoning mit den für diese Arbeit relevant Formalismen ein. Dazu wurden grundlegende Begriffe der logikbasierten Wissensrepräsentation definiert und die Familie der Beschreibungslogiken vorgestellt. Diese sind im Gegensatz zur Prädikatenlogik erster Stufe entscheidbar und bieten jeweils einen attraktiven Kompromiss zwischen der Ausdrucksstärke der Sprache und der Komplexität des Reasoning-Verfahrens. Ausgehend von grundlegenden Reasoning-Problemen wurden dann komplexere Reasoning-Probleme und konjunktive Anfragen eingeführt.

Speziell wurde die Beschreibungslogik $\mathcal{SHIN}$ definiert, die die wesentliche Grundlage der OWL DL-Variante der Ontologiesprache des Semantic Web und damit auch dieser Arbeit bildet. Als Grundlage für spätere Beweise wurde außerdem ein Tableaux-Algorithmus beschrieben, der die Konsistenz einer $\mathcal{SHIN}$-Wissensbasis prüft und damit auch die komplexeren Reasoning-Probleme lösen kann.

4. Semantische Modellierung von Kontextinformationen und Infrastrukturzuständen

Im vorangegangenen Kapitel wurde OWL DL ohne Nominale – und damit die Beschreibungslogik $\mathcal{SHIN}$ – als vielversprechende Wissensrepräsentationssprache für die Zustandsüberwachung bei Infrastrukturnetzen eingeführt. Kapitel 1 und 2 haben motiviert, wie der technische Zustand von Überwachungsobjekten in Infrastrukturnetzen anhand von Kontextinformationen zu diesen Überwachungsobjekten charakterisiert werden kann. Dieses Kapitel beschäftigt sich mit der Entwicklung von $\mathcal{SHIN}$-Ontologien als Wissensmodelle für Kontextinformationen und Infrastrukturzustände. Diese müssen von Domänenexperten erstellt werden, weil nur diese über das nötige Domänenwissen verfügen, das in den Ontologien formalisiert werden soll. Andererseits besitzen Domänenexperten in der Regel nicht die dafür nötige Modellierungskompetenz. Zwar wissen sie, *was* modelliert werden soll, nicht jedoch, *wie* diese Zusammenhänge mit der Wissensrepräsentationssprache am Besten ausgedrückt werden. Dies liegt daran, dass Formalismen zur Wissensrepräsentation mit zunehmender Ausdrucksstärke auch zunehmend komplexer in der Anwendung werden. Um Domänenexperten dennoch in die Lage zu versetzen, effektiv modellierte Domänenontologien zu erstellen, schlägt dieses Kapitel einen musterbasierten Modellierungsansatz vor. Abschnitt 4.1 identifiziert zunächst Eigenschaften, die die zu erstellenden Domänenontologien haben sollen. Vor diesem Hintergrund werden in Abschnitt 4.2 verwandte Arbeiten untersucht. Da diesen eine verallgemeinerbare Vorgehensweise fehlt, wird in den Abschnitten 4.3, 4.4 und 4.5 ein eigener Ansatz auf Basis von Ontologiemustern vorgestellt. Dieser basiert auf den Ergebnissen in [FBP08, FLP+06, LFP+06]. Abschnitt 4.6 bewertet, wie auf diese Weise gewährleistet wird, dass Domänenexperten Ontologien mit den gewünschten Eigenschaften erstellen.

4.1. Anforderungen an die Domänenontologien

Die zu entwickelnde Systemontologie soll einerseits Wissen über primitive Kontextinformationen repräsentieren, die beispielsweise von (Fern-)Überwachungssystemen geliefert werden. Andererseits soll sie auch komplexe Infrastrukturzustände darstellen, wie sie von Anwendungen benötigt werden, die darauf aufsetzen. Wichtig ist vor allem, dass die Zusammenhänge zwischen primitiven Kontextinformationen und komplexen Infrastrukturzuständen formalisiert werden, damit sie automatisiert ausgewertet werden können.

Während *Domänenexperten* über dieses Wissen verfügen, kann nicht davon ausgegan-

gen werden, dass sie die nötige Modellierungskompetenz und -erfahrung besitzen, um qualitativ hochwertige Ontologien zu erstellen. Da *Modellierungsexperten* andererseits nicht über das entsprechende Domänenwissen verfügen, werden hier Möglichkeiten untersucht, Domänenexperten bei der Modellierung zu unterstützen. Dabei muss auch berücksichtigt werden, dass bei so komplexen Systemen wie Infrastrukturnetzen jeder Domänenexperte nur mit einem Teilaspekt des Gesamtsystems vertraut ist. Deshalb muss zusätzlich ermöglicht werden, dass mehrere Domänenexperten gemeinsam an der Modellierung der Systemontologie beteiligt sind.

Um Domänenexperten bei der Ontologieerstellung geeignet unterstützen zu können, werden zunächst Eigenschaften identifiziert, die eine qualitativ hochwertige Domänenontologie besitzen muss. Dazu wird auf entsprechenden Arbeiten von Gruber, Reich und Svatek aufgebaut [Gru95, Rei99, Sva04]. Vor dem Hintergrund der Anforderungen an die Zustandsüberwachung für Infrastrukturnetze aus Abschnitt 2.3 auf Seite 20 werden diese wie folgt erweitert und adaptiert:

A4.1 *Genauigkeit*: Die entstehende Ontologie soll die Realität korrekt abbilden und keine stillschweigenden Annahmen treffen (A2.3, A2.5). Das ist eine notwendige Voraussetzung, um die Ontologie später problemlos erweitern zu können, ohne dass unbeabsichtigte Widersprüche in der Modellierung entstehen.

A4.2 *Verständlichkeit*: Die Terminologie und der Aufbau der Ontologie sollen nachvollziehbar und auch für Menschen schnell erfassbar sein (A2.3). Damit soll erreicht werden, dass es bei der Anwendung der Ontologie zu keinen Missverständnissen bezüglich der in der Ontologie modellierten Begriffe und Konzepte kommt.

A4.3 *Schlussfolgerbarkeit*: Die Semantik von Kontextinformationen und Infrastrukturzuständen soll maschinenverarbeitbar formalisiert sein, um sie zur Laufzeit automatisiert auswerten zu können (A2.1, A2.2, A2.4). Dazu muss sichergestellt werden, dass die Ausdrucksmittel der Repräsentationssprache zweckentsprechend zur Formalisierung der relevanten Zusammenhänge eingesetzt werden.

A4.4 *Erweiterbarkeit*: Die Ontologie soll bereits so entworfen sein, dass sie Ansätze zur Erweiterung bietet. Vor allem soll sie auch zur Laufzeit um zusätzliche Kontextinformationen und Infrastrukturzustände erweitert werden können (A2.5, A2.7).

A4.5 *Wartbarkeit*: In Analogie zur Erweiterbarkeit um zusätzliche Konzepte, sollen zur Laufzeit auch flexible Änderungen an der Modellierung bestehender Konzepte möglich sein. Vor allem soll verhindert werden, dass dadurch unbeabsichtigte Nebeneffekte entstehen können (A2.5, A2.7).

4.2. Verwandte Arbeiten

Im Folgenden werden einige aktuelle und repräsentative Arbeiten vorgestellt, bei denen sich Modellierungsaufgaben stellen, die vergleichbar zur formalen Modellierung von Kontextinformationen und Infrastrukturzuständen sind. Es wird analysiert, inwiefern dort die zuvor identifizierten Anforderungen an die erstellten Ontologien erfüllt

werden. Konkret werden die Themengebiete *Sensor Web*, *Ambient Intelligence* und *Infrastrukturüberwachung* betrachtet.

4.2.1. Sensor Web

Die Vision des *Sensor Web* wurde 1997 vom *Jet Propulsion Laboratory* der NASA entwickelt [Del04]. Es hat als wichtigen Bestandteil die Modellierung von Sensordaten und damit Kontextinformationen. Das Sensor Web wird definiert als „System von kommunizierenden, räumlich verteilten Sensorkomponenten, die eingesetzt werden, um neue Umgebungen zu überwachen und zu erforschen“. Das Ziel des Sensor Web besteht darin, aus den gesammelten Daten Wissen zu extrahieren und darauf aufbauend Anpassungen vorzunehmen und zu reagieren [Del02, Del06]. Dabei werden der Austausch von Informationen zwischen Sensorkomponenten und die kollektive Interpretation dieses Wissens als zentrale Aspekte angesehen, um bessere Ergebnisse zu erzielen als dies mit einzelnen Sensorkomponenten möglich wäre.

Die *Sensor Web Enablement* (SWE) Initiative des *Open Geospatial Consortium* (OGC) greift diese Vision auf und möchte eine dienstorientierte Architektur für das Sensor Web spezifizieren [BPW+07]. Dazu werden Standards für Schnittstellen und Metadaten-Formate auf Basis von Web-Standards wie SOAP und XML entwickelt.

Im Rahmen der SWE Initiative entstehen mit dem *Observations & Measurements*-Standard (O&M) deshalb auch Spezifikationen für die Modellierung von Beobachtungen und Messungen, die von Sensoren geliefert werden. O&M bietet ein XML-Modell, das in Kombination mit der *Geography Markup Language* (GML) flexibel und erweiterbar Daten und Observationen beschreiben kann und auch die Möglichkeit bietet, große Datenmengen in ASCII oder binär zu kodieren. Gemäß Spezifikation hat eine Beobachtung (*observation*) ein Ergebnis (*result*), dessen Wert das Phänomen beschreibt. Die Beobachtung wird als ein Merkmal (*feature*) im Sinne des ISO/OGC *Feature Model* definiert, einem von der OGC in Zusammenarbeit mit der ISO entwickelten Katalogdienst (OGC *Catalog Service Specification* CAT *2.0*), der sich an zahlreichen ISO-Standards zur Katalogisierung von „feature types“ orientiert. Dieses Beobachtungsmerkmal verknüpft das Ergebnis mit dem Merkmal, das gemessen wurde.

Die SWE-Standards bieten zwar wichtige Spezifikationen für die Realisierung einer verteilten Sensorplattform, mit der Sensor-Ressourcen veröffentlicht, automatisch aufgefunden und abgefragt werden können. Die Möglichkeiten zur Beschreibung der Semantik von Beobachtungen und Messungen sind aufgrund der fehlenden Formalisierung jedoch sehr eingeschränkt. Das erschwert die dynamische Fusion von Sensorinformationen und die Veredelung zu höherwertigen Kontextinformationen. Das SWAP *Framework* erweitert deshalb den SWE-Ansatz und führt Ontologien für die Modellierung der Beobachtungen und Messungen ein [MS06a]. Dadurch soll die Semantik der Kontextinformationen formalisiert und damit für automatisches Schlussfolgern nutzbar gemacht werden. Als Grundlage für die Entwicklung derartiger Ontologien geben die Autoren jedoch lediglich eine Sammlung bestehender Ontologien an, auf denen aufgebaut werden soll. Dazu gehören die Ontologien aus dem SWEET-Projekt der NASA[1] sowie die *OntoSensor*-Ontologie [RKT05]. Wie eine Domänenontologie für ein

[1] Semantic Web for Earth and Environmental Terminology (SWEET) Ontologien: http://sweet.jpl.nasa.gov/ontology

bestimmtes Fachgebiet entwickelt und ihre Qualität sichergestellt werden kann, wird nicht weiter beleuchtet. Deshalb ist nicht zu erwarten, dass es Domänenexperten ohne Modellierungskompetenz gelingen wird, damit Fachontologien zu erstellen, die eine hohe Schlussfolgerbarkeit besitzen.

4.2.2. Ambient Intelligence und Ubiquitous Computing

Kontextmodellierung ist eine zentrale Herausforderung für Anwendungen des *Ubiquitous Computing* und der *Ambient Intelligence*. Zahlreiche Projekte in diesem Umfeld haben unterschiedlichste Ansätze entwickelt. Die Interoperabilität heterogener Kontextquellen sowie die Ableitung höherwertiger Kontextinformationen aus primitiven Kontextinformationen standen von Anfang an im Mittelpunkt. Dazu wurden früh Techniken der formalen Wissensrepräsentation eingesetzt [BFP06, FHKB05, GWPZ04, CFJ04]. Einige aktuelle und repräsentative Beispiele werden nachfolgend vorgestellt.

MobiLife-Projekt. Als Beispiel für ein europäisches Forschungsprojekt wird *MobiLife* vorgestellt. In seinem Rahmen wurde ein *Context Representation Framework* (*CRF*) entwickelt, das unter anderem auf Ontologien aufbaut [FPN+05]. Situationen werden als beschreibungslogische Konzepte definiert und können deshalb unter anderem mittels beschreibungslogischem Schlussfolgern abgeleitet werden [LMW+05]. Für die Modellierung dieser Situationskonzepte werden jedoch keine Vorgaben gemacht oder Hilfestellungen geliefert. Es wird zwar auf einer „Upper Context Ontology“ aufgebaut, dabei ist jedoch unklar, wie diese aussieht und wie dadurch die Qualität der entstehenden Ontologie erhöht werden kann.

Kontextsensitives Türschloss. Als weitere interessante Arbeit soll [TSB06] herausgegriffen werden, in der exemplarisch ein kontextsensitives Türschloss modelliert wird. Im Gegensatz zu anderen Arbeiten legen Turhan et al. den Schwerpunkt darauf, die wohldefinierte Semantik der beschreibungslogischen Wissensrepräsentation voll auszunutzen: Sie definieren relevante Situationsklassen als beschreibungslogische Konzepte und repräsentieren eine konkrete Situation als Individuum mit entsprechenden Rollen-Zusicherungen. Damit kann beschreibungslogisches Reasoning genutzt werden, das bewiesenermaßen vollständig und korrekt ist, um dieses Situations-Individuum zu klassifizieren. Im Gegensatz zu anderen Ad-hoc-Reasoning-Ansätzen, die nach Bedarf entwickelt werden, stellt die Vollständigkeit des hier verwendeten Verfahrens sicher, dass wirklich *alle* impliziten Zusammenhänge erkannt werden.

Zur Modellierung der Situationsontologie wird aufgabenbezogen vorgegangen: Für eine bestimmte kontextsensitive Anwendung werden Aufgaben identifiziert, die diese Anwendung erfüllen soll. Dann werden Situationen abgeleitet, in denen diese Aufgaben ausgeführt werden. Für diese werden entsprechende ontologische Konzepte erstellt. Auch hier fehlen jedoch allgemeingültige Leitlinien, wie solche Situationskonzepte modelliert werden müssen.

Schlussfolgerungen. Aktuelle Analysen teilen die Einschätzung, dass die effektive Modellierung von Kontextinformationen mit Hilfe von Beschreibungslogiken zwar

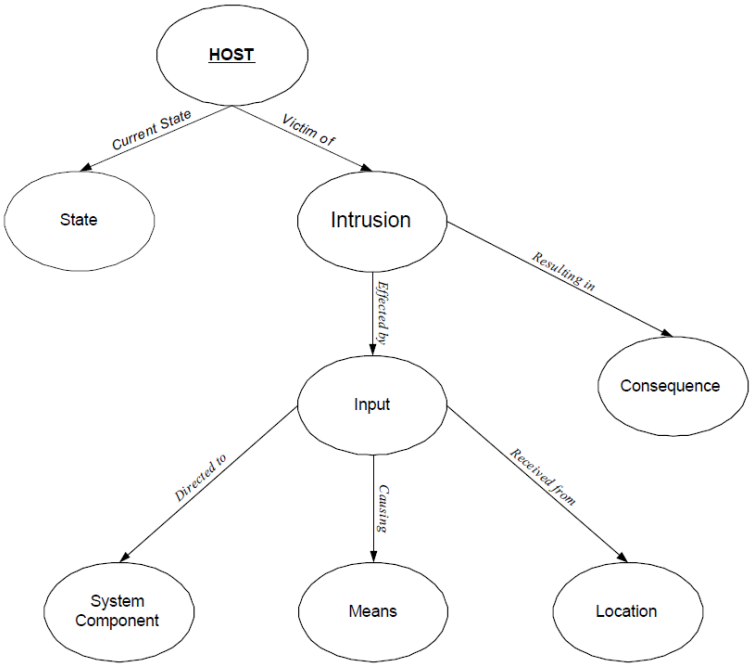

Abbildung 4.1.: Ontologie zur Eindringlingserkennung in Rechnernetzen [UJFP04].

vielversprechend ist [SLP04], jedoch noch in den Kinderschuhen steckt [KS07b]. Als wichtigste Anforderungen für zukünftige Arbeiten wird angeführt, dass qualitativ hochwertige Kontextontologien entwickelt werden müssen, die die beschreibungslogischen Ausdrucksmöglichkeiten effektiv nutzen, um die relevante Semantik maschinenverarbeitbar zu formalisieren. Gleichzeitig müssen diese Ontologien frei zugänglich gemacht werden, damit die so häufig genannten Ziele Interoperabilität und Wiederverwendbarkeit überhaupt erreicht werden können.

4.2.3. Infrastrukturüberwachung

Auch auf dem Gebiet der Überwachung von Infrastrukturnetzen gibt es erste formale Ansätze zur Modellierung von Kontextinformationen und Infrastrukturzuständen. Einige werden nachfolgend vorgestellt.

Rechnernetze. Undercoffer untersucht die formale Modellierung von Systemzuständen für Eindringlingserkennung (engl. *intrusion detection*) in Rechnernetzen [Und04, UJFP04]: Mittels systemnaher Verfahren werden verschiedene Charakteristika von Prozessen, dem Betriebssystem und dem Netzwerk erfasst, die als Indizien für Systemangriffe dienen können. Beispiele dafür sind *Denial-of-Service (*DoS*)* Angriffe, TCP *probes* und IP *spoofing*. In einem zweiten Schritt werden diese Charakteristika dann als Individuen repräsentiert und mittels beschreibungslogischem Reasoning zu Systemzustandskonzepten klassifiziert. Ein interessantes Beispiel ist der sogenannte Mitnick-Angriff, der aus mehreren Phasen besteht und einen DoS-Angriff, TCP Sequenzvoraussage und IP spoofing umfasst. Er wird z. B. folgendermaßen modelliert:

$$\mathsf{SystemUnderMitnickAttack} \equiv (\mathsf{SystemUnderProbeAttack} \sqcap$$

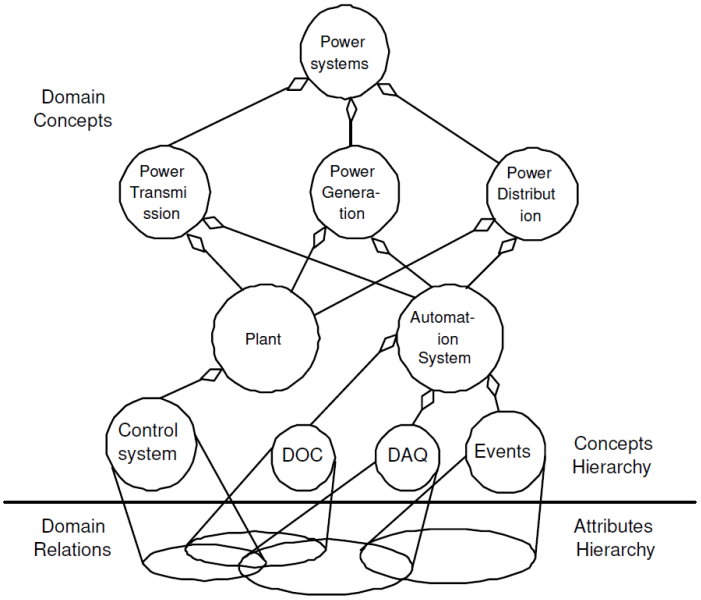

Abbildung 4.2.: Struktur einer Ontologie für Konzepte und Attribute in Stromnetzen [FWF04].

$$\exists\,\mathsf{connectedTo.SystemUnderProbeAttack})$$
$$\sqcup\ (\mathsf{SystemUnderDoSAttack} \sqcap$$
$$\exists\,\mathsf{connectedTo.SystemUnderDoSAttack})$$

$$\mathsf{SystemUnderProbeAttack} \equiv \mathsf{System} \sqcap \exists\,\mathsf{experiencing.Probe}$$

$$\mathsf{SystemUnderDoSAttack} \equiv \mathsf{System} \sqcap \exists\,\mathsf{experiencing.DoS}$$

$$\mathsf{SynFlood} \equiv \mathsf{DoS} \sqcap \exists\,\mathsf{tcpEstb.Amount\_WA\_Normal} \sqcap \exists\,\mathsf{tcpSynRec.Rate\_WA\_Normal}$$

$$\mathsf{RstProbe} \equiv \mathsf{Probe} \sqcap \exists\,\mathsf{icmpOutMsg.Rate\_WA\_Normal} \sqcap$$
$$\exists\,\mathsf{tcpEstabRst.Rate\_WA\_Normal} \sqcap \exists\,\mathsf{tcpOutRst.Rate\_WA\_Normal}$$

Zur Modellierung von Systemzuständen gibt Undercoffer ein Muster vor, das die Kernkonzepte *Input*, *Host* und *Intrusion* umfasst und in Abbildung 4.1 dargestellt ist. Die verschiedenen Konzepte werden in [UJFP04] genauer erläutert. Dem Ersteller der Ontologie wird damit eine Grundstruktur an die Hand gegeben, die sich für die Modellierung von Systemzuständen bewährt hat und die er deshalb bei der Modellierung eigener Systemzustände wiederverwenden kann. Zur Infrastrukturüberwachung im Rahmen dieser Arbeit lässt sich dieses Muster jedoch nicht übernehmen, weil es durch die Konzepte *Host*, *Intrusion* usw. auf die Domäne der Eindringlingserkennung zugeschnitten ist.

Stromnetze. Auf dem Gebiet der in dieser Arbeit betrachteten materiellen Infrastrukturen gibt es vor allem bei Stromnetzen erste Überlegungen für die formale Modellierung von Infrastrukturzuständen [SLW+07, FWF04, FSWF05].

Feng et al. schlagen die Nutzung beschreibungslogischer Ontologien vor [FWF04]. Sie verwenden jedoch eine sehr eingeschränkte Beschreibungslogik, die nur die Konjunktion

Anforderung	Sensor Web	Ambient Intelligence	Infrastruktur-überwachung
4.A1. Genauigkeit	+	+/−	+
4.A2. Verständlichkeit	+/−	−	+/−
4.A3. Schlussfolgerbarkeit	−	+	+/−
4.A4. Erweiterbarkeit	+/−	−	+/−
4.A5. Wartbarkeit	−	+/−	+/−

Tabelle 4.1.: Bewertung verwandter Arbeiten zur Modellierung von Kontextinformationen und Infrastrukturzuständen.

(Schnittmenge) von Konzepten unterstützt. So kann der Zustand eines Transformatorkühlers anhand der Temperatur des Kühlöls unter Verwendung atomarer Konzepte folgendermaßen formalisiert werden:

$$\mathsf{cooler.on} \equiv \mathsf{power.on} \sqcap \neg\mathsf{cooler.off} \sqcap \mathsf{cooling\_oil\_temperature.drop} \sqcap \mathsf{ambient\_temperature.unchanged}$$

Wie in Abbildung 4.2 dargestellt ist, wurde eine sogenannte *upper ontology* entwickelt, die grundlegende Zusammenhänge vorgibt und zwischen Konzepten und Attributen differenziert. Kontextinformationen werden als Attribute bezeichnet. Feng et al. listen zwar mehrere Beispiele für die Attribute einer Transformator-Ontologie auf. Es ist jedoch unklar, nach welchen Kriterien diese identifiziert und wie sie modelliert wurden. Unabhängig davon ist die vorgeschlagene Ontologie auf die Domäne der Stromnetze zugeschnitten und kann deshalb nicht für die Modellierung anderer Infrastrukturdomänen verwendet werden.

4.2.4. Zusammenfassende Bewertung

Die Analyse verwandter Arbeiten zeigt, dass es bisher nur wenige Ansätze zur formalen Modellierung von Kontextinformationen und Zuständen in Infrastrukturnetzen gibt. Tabelle 4.1 fasst die Bewertung der vorgestellten Ansätze noch einmal zusammen.

Die Arbeiten aus dem Gebiet des Sensor Web zeichnen sich durch ihre Allgemeingültigkeit und – als Standard eines großen Konsortiums – ihre Konsensfähigkeit aus. Da ihnen jedoch jegliche Formalisierung der Semantik fehlt, eignen sie sich nicht für automatisches Schließen. Ihre Komplexität führt dazu, dass die Wartung aufwändig ist. Proprietäre Erweiterungen um formale Modellierung sind Ad-hoc-Erweiterungen, die nur schwer für eigene Anwendungen übernommen werden können.

Kontextsensitive Systeme im Umfeld von Ambient Intelligence versuchen, Kontextinformationen oder sogar komplexe Situationen zu modellieren und greifen dazu häufig auf Ontologien zurück. Zur Ontologieerstellung wird in der Regel als einzige Hilfestellung eine *upper ontology* angegeben, die grundlegende Zusammenhänge vorgibt. Weitere Anhaltspunkte fehlen, was auch damit zu begründen ist, dass der Kontext- und Situationsbegriff sehr breit definiert ist [Dey01]. Im Gegensatz zu Sensor Web-Ansätzen

Ansatz	Genauigkeit	Verständlichkeit	Schlussfolgerbarkeit
Methodiken	*	*	
Upper ontologies	*	**	
Meta-Eigenschaften	**		
Benutzerhandbücher	*		
Gesammelte Hinweise	*	*	*
Modellierungsmuster	*	**	**

Tabelle 4.2.: Effektivität von Modellierungs-Ansätzen für die Qualität der Ontologie (nach [Sva04]).

fehlt Ambient Intelligence-Ansätzen die Allgemeingültigkeit und Erweiterbarkeit, um für die Modellierung von Infrastrukturzuständen genutzt werden zu können.

Auf dem Gebiet der Infrastrukturüberwachung gibt es einige wenige Beispiel für die Verwendung formaler Modellierung. Diese konzentrieren sich jeweils auf eine Domäne (Rechnernetze, Stromnetze) und legen dort eine sinnvolle Grundlage, weil sie die korrekte Terminologie verwenden. Da die Ontologien jedoch nicht generisch aufgebaut sind, können sie nicht verallgemeinert und auf andere Infrastrukturen übertragen werden. Zum automatischen Schließen werden zwar jeweils Beispiele angegeben. Diese sind jedoch ebenfalls auf die erwarteten Anwendungen zugeschnitten und lassen sich nicht verallgemeinern. Vor allem wird keine Anleitung gegeben, wie erreicht werden kann, dass eine Ontologie mit hoher Schlussfolgerbarkeit erstellt wird.

Insgesamt zeigt sich also, dass für einen generischen Ansatz zur Modellierung von Kontextinformationen und Infrastrukturzuständen hauptsächlich die Sensor Web-Arbeiten inhaltliche Anregungen bezüglich der relevanten Aspekte geben können. Für Modellierungstechniken und spezifische Beispiele können eventuell einige Arbeiten auf dem Gebiet der Ambient Intelligence und der Infrastrukturüberwachung verallgemeinert werden. Dafür muss ein Mittel gefunden werden, dass es auch Domänenexperten ohne Modellierungskompetenz ermöglicht, qualitativ hochwertige Ontologien zu erstellen.

4.3. Erstellung qualitativ hochwertiger Ontologien

Dieser Abschnitt untersucht, wie Domänenexperten bei der Erstellung qualitativ hochwertiger Domänenontologien unterstützt werden können.

4.3.1. Vergleich unterschiedlicher Ansätze

Um die Qualität von Ontologien sicherzustellen, wurden verschiedenste Ansätze vorgeschlagen. Svatek vergleicht die wichtigsten von ihnen und bewertet sie bezüglich der *Genauigkeit* (engl. *accuracy*), *Verständlichkeit* (engl. *transparency*) und *Schlussfolgerbarkeit* (engl. *reason-ability*) der entstehenden Ontologien [Sva04]. Das Ergebnis ist in Tabelle 4.2 dargestellt. Er berücksichtigt folgende Ansätze:

- *Methodiken* geben grundlegende Vorgehensweisen zur Erstellung von Ontologien vor. Sie sind unabhängig von der verwendeten Ontologiesprache und steuern den Prozess der Erstellung. Zur inhaltlichen Gestaltung werden in der Regel keine Anhaltspunkte gegeben.
- *Upper ontologies* sind sorgfältig erstellte Ontologien, die abstrakte und allgemeingültige Zusammenhänge der Welt oder einer bestimmten Domäne modellieren. Sie sollen als gemeinsamer Bezugspunkt für spezialisierte Ontologien dienen.
- *Meta-Eigenschaften* werden zur Annotation von Ontologiekonzepten und -rollen verwendet. Mittels vorgegebener Konsistenzregeln sollen Ontologien damit automatisiert (neben der logischen) auch auf inhaltliche Konsistenz geprüft werden können [GW02]. So soll z. B. verhindert werden, dass eine unveränderliche Eigenschaft wie Person als Unterkonzept einer veränderlichen Eigenschaft wie Student modelliert wird.
- *Benutzerhandbücher* sind für eine bestimmte Ontologiesprache geschrieben und beschreiben ihre Anwendung. Sie definieren die Semantik der Sprachkonstrukte sehr genau, geben umgekehrt in der Regel jedoch keine Anhaltspunkte, wie Sprachkonstrukte eingesetzt werden, um eine bestimmte Semantik auszudrücken.
- *Gesammelte Hinweise* entstehen üblicherweise aus der Modellierungspraxis heraus. Ausgehend von einer Modellierungsentscheidung beschreiben sie, wie sich diese mit einer bestimmten Sprache umsetzen lässt. Diese Sammlungen sind in der Regel ungeordnet und informal.
- *Modellierungsmuster* beschreiben ebenfalls bewährte Lösungen für häufig vorkommende Modellierungsprobleme. Im Gegensatz zu gesammelten Hinweisen, werden Modellierungsmuster jedoch systematisch dokumentiert und sorgfältig verallgemeinert, um breit einsetzbar zu sein.

Genauigkeit, *Verständlichkeit* und *Schlussfolgerbarkeit* wurden in Abschnitt 4.1 ebenfalls als wichtige Eigenschaften der erforderlichen Domänenontologien identifiziert. Um Domänenexperten in die Lage zu versetzen, derartige Ontologien zu erstellen, eignen sich gemäß Svatek also vor allem *gesammelte Hinweise* und – als ihre systematisierte Form – *Modellierungsmuster*. Diese helfen, die *Schlussfolgerbarkeit* und *Verständlichkeit* von Ontologien sicherzustellen. Die *Genauigkeit* kann mittels *Meta-Eigenschaften* erhöht werden. Dazu müssen Ontologien jedoch mit speziellen Meta-Eigenschaften annotiert werden. Obwohl dies sehr aufwändig ist, sind die dadurch automatisierbaren Überprüfungen in ihren Möglichkeiten stark eingeschränkt [Sva04]. Deshalb sollen Domänenexperten nicht damit belastet werden. Stattdessen wird hier der Ansatz gewählt, die Modellierungsmuster an bewährte und etablierte Standards auf dem Gebiet der Zustandsüberwachung anzulehnen, um dadurch eine hohe Genauigkeit der erstellten Domänenontologien sicherzustellen.

4.3.2. Modellierungsmuster für Ontologien

Um im Folgenden Modellierungsmuster für Domänenexperten entwickeln zu können, wird der musterbasierte Modellierungsansatz zunächst weiter untersucht:

Im Allgemeinen bieten *Muster* (engl. *patterns*) bewährte Lösungen für häufig auftretende Modellierungsprobleme. In der Informatik hat sich dieser Ansatz vor allem in der Softwaretechnik bewährt, wo Software-Entwurfsmuster (engl. *software design patterns*) Hilfestellung unter anderem bei der Strukturierung von Softwaresystemen geben [GHJV95]. Eine allgemeinere Definition von Mustern in der Informatik – wenn auch mit Bezug auf Systemarchitekturen – geben Buschmann et al. [BMR+96]:

Muster ...

... adressieren ein wiederkehrendes Entwurfsproblem, das sich in bestimmten Entwurfsituationen ergibt, und liefern dafür eine Lösung.

... dokumentieren vorhandene und bewährte Entwurfserfahrungen.

... identifizieren und spezifizieren Abstraktionen, die sich oberhalb einzelner Klassen und Instanzen von Komponenten bewegen.

... bieten ein einheitliches Vokabular und Verständnis für Entwurfsprinzipien.

... sind ein Mittel zur Dokumentation [einer Architektur].

... helfen bei der Konstruktion komplexer und heterogener Architekturen.

... helfen bei der Bewältigung von Komplexität.

Dieser Ansatz lässt sich auch auf die Erstellung beschreibungslogischer Ontologien übertragen. Erste Arbeiten beschäftigen sich mit der Entwicklung von *Ontologiemustern* (engl. *ontology design patterns,* ODP) [Rei99, Sva04, BS05, Blo04, SEM01, SWBP08]. Reich definiert Ontologiemuster wie folgt [Rei99]:

> *Ontologiemuster benennen, abstrahieren und identifizieren ontologische Entwurfsstrukturen, einzelne Begriffe, zusammengesetzte größere Ausdrücke und verwandte semantische Kontexte. [..]*
>
> *Ontologiemuster beschreiben einfache, elegante, flexible und wiederverwendbare Lösungen für spezifische Entwurfsprobleme [..]*

Als Kriterium zur Klassifikation von Ontologiemustern eignen sich (im Gegensatz zu Software-Entwurfsmustern) nicht die Phasen des Entwurfsprozesses, da sich Ontologiemuster im Laufe des Entwurfsprozesses in der Regel nicht signifikant ändern. Blomqvist und Sandkuhl schlagen stattdessen eine Klassifikation bezüglich ihres Abstraktionsgrades vor und unterscheiden folgende Klassen von Ontologiemustern [BS05]:

Syntaktische Muster beziehen sich auf eine bestimmte Repräsentationssprache und geben Leitlinien für die Verwendung dieser Sprache zur Modellierung bestimmter Zusammenhänge.

Semantische Muster abstrahieren von der Syntax einer bestimmten Repräsentationssprache und sind damit sprachunabhängig. Dadurch sind sie breiter einsetzbar als syntaktische Muster.

Entwurfsmuster bestehen aus einer kleinen Zahl semantischer Muster, die als generisches Konstrukt zur Ontologieentwicklung verwendet werden können. Nach ihrer Instantiierung stellen sie einen Teil der vollständigen Ontologie dar. Sie sind also im Wesentlichen verallgemeinerte und wiederverwendbare Teile von Ontologien.

Architekturmuster beschreiben, auf welche Weise Entwurfsmuster zu einer Gesamtstruktur einer Ontologie kombiniert werden können, um ein Entwurfsziel zu erreichen. Derartige Muster sind in der Regel domänenunabhängig. Bisher gibt es nur sehr wenige Architektur-Muster. Vor allem Muster zur Modularisierung von Ontologien können hier genannt werden [Rec03, Stu03].

Anwendungsmuster beziehen sich auf Zweck, Umfang und Nutzungsumfeld einer Ontologie. Sie bieten bewährte Leitlinien für die Nutzung einer Ontologie für eine bestimmte Anwendung oder in einem bestimmten Kontext. Bisher wurden keine Muster für diese Klasse vorgeschlagen. Sie werden domänenabhängig sein.

Um Domänenexperten bei der Modellierung zu unterstützen, sind also folgende Klassen von Ontologiemustern relevant:

- Zur Modellierung von Infrastrukturzuständen müssen unterschiedliche semantische Zusammenhänge formalisiert werden. Dazu wird ein *Entwurfsmuster* benötigt, das mehrere *semantische Muster* kombiniert. Es wird in Abschnitt 4.4 entwickelt.
- Zur Erstellung einer konsistenten Gesamtontologie müssen die Teilontologien mehrerer Domänenexperten integriert werden. Dazu wird ein *Architekturmuster* benötigt. Dieses wird in Abschnitt 4.5 entwickelt.

4.4. Entwurfsmuster-basierte Modellierung von Infrastrukturzuständen

Zur Entwicklung eines Entwurfsmusters für Infrastrukturzustände müssen zunächst grundlegende Konzepte und Zusammenhänge der Zustandsüberwachung identifiziert werden. Dazu wird das Fachgebiet der Zustandsüberwachung und -diagnostik in Abschnitt 4.4.1 analysiert, um in Abschnitt 4.4.2 ein entsprechendes Entwurfsmuster abzuleiten.

4.4.1. Zustandsüberwachung und -diagnostik

Die Zustandsüberwachung und -diagnostik ist ein weitläufiges Gebiet mit breitem Anwendungsspektrum. Diese Analyse stützt sich deshalb auf Arbeiten, die von spezifischen Anwendungen abstrahieren und allgemeingültige Konzepte entwickeln [Vac06, VRYK03]. Zudem werden internationale Standards und Normen auf diesem Gebiet berücksichtigt.

Begriffe. Zunächst werden die zentralen Begriffe auf diesem Gebiet vorgestellt und definiert. Dazu stützt sich diese Arbeit auf den internationalen und deutschen Standard zur *Zustandsüberwachung und -diagnostik von Maschinen* DIN ISO 17359[2], der

[2] DIN ISO 17359:2003-11

kontinuierlich weiterentwickelt wird und zur Zeit in der Version vom November 2003 vorliegt. Die deutsche Version dieses Standards wird ergänzt um ein *Beiblatt 1* vom August 2007[3], in dem die verwendeten Fachbegriffe erläutert werden.

Auf Basis dieser Standards lassen sich die zentralen Begriffe *Überwachungsobjekt*, *Merkmal*, *Symptom*, *Fehler* und *Zustand* für den Zweck dieser Arbeit wie folgt definieren. (Die Ziffern in eckigen Klammern geben jeweils die Referenz-Ziffer im Standard DIN ISO 17359 *Bbl* 1:2007-08 an.)

Definition 4.4.1 (Überwachungsobjekt, *engl.* monitoring subject). *[5.1] Betrachtungseinheit, deren Zustand durch Überwachung oder Diagnostik festgestellt werden soll.*

In dieser Arbeit stellen die Infrastrukturkomponenten die Überwachungsobjekte dar. Im Schienennetz also z. B. Waggons, Achsen, Gleisabschnitte und Weichen; bei Stromnetzen Leitungen, Transformatoren und Unterbrecher.

Definition 4.4.2 (Merkmal, *engl.* feature). *[7.1] Zahlenwert der durch Signalverarbeitung gewonnenen Kenngröße eines Messsignals oder zahlenmäßiges Beobachtungsergebnis, der bzw. das mit einem Bezugswert verglichen wird. [..]*

Bei der Zustandsüberwachung und -diagnostik werden Merkmale verwendet, die zustandsabhängig sind und Symptome quantitativ beschreiben. Sie können auch quantitative Ausdrücke für Wertebereiche qualitativ beobachteter Symptome (z. B. sehr heiß, heiß, warm, kühl, kalt, sehr kalt) sein.

Diagnosemerkmale stehen über das Diagnosemodell in einem bekannten Zusammenhang mit den Zustandsparametern oder mit der Zugehörigkeit zu Zustandsklassen bzw. Nutzungsklassen.

Bei Infrastrukturnetzen werden Diagnosemerkmale zu Überwachungsobjekten direkt von Überwachungssystemen geliefert, z. B. die Meldung eines Achslastgebers, dass ein bestimmter Zug die Toleranzgrenzen eingehalten hat, oder die Meldung eines Schutzschalters, dass er ausgelöst wurde.

Definition 4.4.3 (Symptom, *engl.* symptom). *[6.14] Mess- oder beobachtbare physikalische Erscheinung, die eine Verhaltensabweichung widerspiegelt und dadurch mit einer gewissen Wahrscheinlichkeit die Existenz eines oder mehrerer Schäden (Anm. Schaden ist als eine Untergruppe von Fehler definiert) oder eines Ausfalls anzeigt. [..]*

Das Symptom wird zum Zweck der Zustandsüberwachung oder Diagnostik durch ein oder mehrere Merkmale quantifiziert. [..]

Im Schienennetz könnte ein Symptom einer Weiche sein, dass das Schalten länger als vorgesehen dauert. Bei Stromnetzen könnte die Gasdichte in einem Isolator unter einen bestimmten Schwellwert gefallen sein. Diese Informationen werden teilweise auch von Überwachungssystemen geliefert oder können aus Merkmalen abgeleitet werden.

[3] DIN ISO 17359 Beiblatt 1:2007-08. Dieses Beiblatt wurde auf Basis mehrerer DIN-Normen, ISO-Normen und VDI-Richtlinien zur Zustandsüberwachung entwickelt. Es ist der internationalen Norm ISO 13372 *Condition monitoring and diagnostics of machines – Vocabulary* vorzuziehen, da es auch Bezug auf die deutschsprachigen Dokumente enthält und Vollständigkeit und Aktualität bei Bezug auf andere Dokumente gewährleistet.

Definition 4.4.4 (Fehler, *engl.* fault). *[6.5] Abweichung des Istwerts einer Eigenschaft (des Zustands, der Funktion) eines Überwachungs- oder Diagnoseobjekts vom Sollwert.*

Nach DIN 31051 *und* DIN EN 13306 *ist Fehler als „Unfähigkeit, eine geforderte Funktion zu erfüllen" definiert.*

Im Schienennetz könnte ein Fehler einer Weiche darin bestehen, dass die Weichenheizung nicht mehr die erforderliche Wärme erzeugt. Bei Stromnetzen ist es möglich, dass ein Transformator nur eingeschränkte Leistung liefert.

Definition 4.4.5 ((technischer, Erhaltungs-) Zustand, *engl.* condition; state). *[6.1] Eigenschaft eines Überwachungs- oder Diagnoseobjekts, die die Funktionsfähigkeit bzw. ihre Einschränkung oder Gefährdung durch Fehler oder Schäden oder ihre Beendigung durch Ausfälle ausdrückt. [..]*

Ein Überwachungs- oder Diagnoseobjekt kann mehrere Fehler oder Schäden gleichzeitig aufweisen. Diese werden durch verschiedene Arten von Zustandsparametern oder -klassen beschrieben.

Im Schienenverkehr können beispielsweise ein Radsatz oder eine Weiche (dringend) instandsetzungsbedürftig sein. Beim Stromnetz kann ein Unterbrecher defekt oder ein Transformator in der Leistungsfähigkeit eingeschränkt sein.

Generischer Zustandsüberwachungs-Prozess. Zusätzlich zu den zentralen Begriffen für die Zustandsüberwachung müssen auch zentrale Schritte beim Prozess der Zustandsüberwachung geklärt werden. Venkatasubramanian et al. charakterisieren Zustandsüberwachung und -diagnostik als Klassifikationsproblem [VRYK03]. Sie strukturieren den Klassifikationsprozess in verschiedene Schritte, die Transformationen von einem Bezugsraum in einen anderen entsprechen. Dabei ist jeweils wichtig, welches A-priori-Domänenwissen bei der Durchführung der Transformation nötig ist. Folgende Schritte werden unterschieden (vergleiche Abbildung 4.3):

1. Der *Messwerte-Raum* umfasst die Menge der Messwerte. Diese ist unabhängig von A-priori-Domänenwissen und entspricht lediglich den mittels Sensorik erfassten Werten.

2. Der *Merkmals-Raum* umfasst die Menge der Merkmale. Merkmale werden mittels A-priori-Domänenwissen auf Basis der Messwerte identifiziert.

3. Der *Entscheidungs-Raum* umfasst die Menge der entscheidungsrelevanten Symptome. Diese werden aus der Menge der Merkmale extrahiert.

4. Der *Fehlerklassen-Raum* umfasst die Menge der Fehlerklassen. Diese werden auf Basis der entscheidungsrelevanten Symptome identifiziert.

Existierende Überwachungssysteme führen die ersten beiden Schritte durch. Je nach Konzeption bieten sie zusätzlich Funktionalität für die weiteren zwei Schritte. Dabei können jedoch nur Fehlerklassen für die jeweils überwachte Domäne erkannt werden. Zur Integration von Kontextinformationen aus unterschiedlichen Quellen muss bei

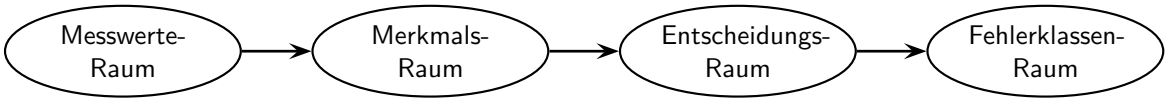

Abbildung 4.3.: Generische Verarbeitungsschritte in einem Zustandsüberwachungssystem.

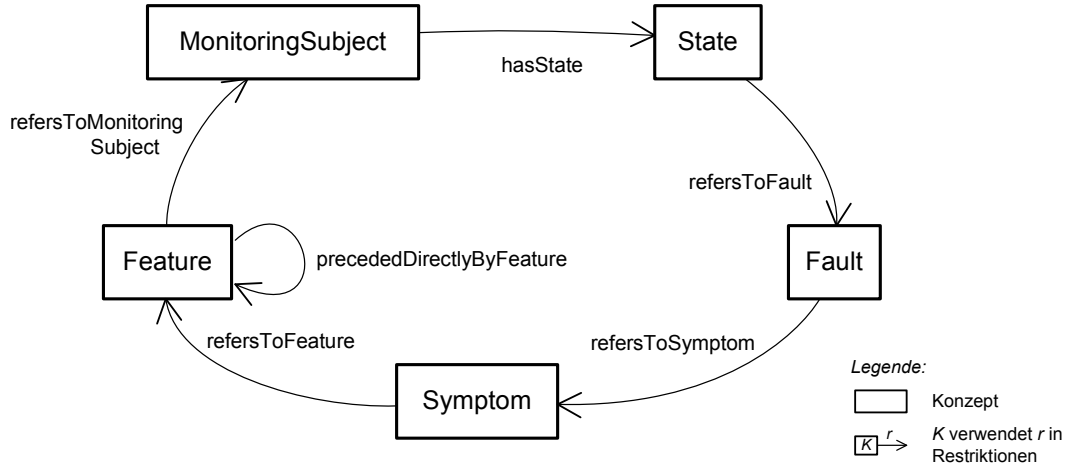

Abbildung 4.4.: Kernkonzepte und Zusammenhänge des Entwurfsmusters für Infrastrukturzustände.

den Schritten drei und vier angesetzt werden und es müssen jeweils Merkmale und Symptome mehrerer Überwachungssysteme und Kontextquellen berücksichtigt werden.

Dieser Abschnitt hat zentrale Begriffe und ein generisches Prozessmodell für die Zustandsüberwachung vorgestellt. Darauf wird im Folgenden aufgebaut, um ein Entwurfsmuster für die beschreibungslogische Modellierung von Infrastrukturzuständen zu entwickeln.

4.4.2. Ontologie-Entwurfsmuster für Infrastrukturzustände

Gesucht ist eine ontologische Modellierung, mit welcher der aktuelle *Zustand* eines *Überwachungsobjekts* aus seinen aktuellen *Merkmalen* abgeleitet werden kann. Dazu benötigen Domänenexperten ein Ontologie-Entwurfsmuster, mit dessen Hilfe sie die Zusammenhänge zwischen Merkmalen (zu diesem Überwachungsobjekt) und den möglichen Zuständen (des Überwachungsobjekts) modellieren können.

Die Kernkonzepte des Entwurfsmusters sind in Abbildung 4.4 illustriert: Merkmale werden als Feature-Konzept repräsentiert. Sie beziehen sich auf Überwachungsobjekte, die als MonitoringSubject-Konzept modelliert sind. Ein Überwachungsobjekt besitzt Zustände, die als State-Konzept repräsentiert werden.

Die Analyse der Zustandsüberwachung im vorangegangenen Abschnitt hat ergeben, dass zusätzlich zwischen *Symptomen* und *Fehlern* unterschieden werden muss. Sie werden deshalb ebenfalls im Entwurfsmuster als Symptom- und Fault-Konzepte repräsentiert. Symptome werden mit Bezug zu Merkmalen, Fehler mit Bezug zu Symptomen modelliert. Auf Basis von Fehlern kann schließlich der Zustand eines Überwachungsob-

jekts spezifiziert werden. Die Einführung separater Symptom- und Fehlerkonzepte hat mehrere Vorteile:

- Domänenexperten sind mit den Begrifflichkeiten vertraut. Es fällt ihnen deshalb leichter, das Entwurfsmuster korrekt anzuwenden.
- Die Wiederverwendbarkeit von Konzeptdefinitionen wird erhöht, da zwischen Merkmalen, Symptomen und Fehlern differenziert wird. So kann ein neues Zustandskonzept beispielweise mit Bezug auf bereits bestehende Fehlerkonzepte definiert werden. Oder neue Merkmals- und Symptomkonzepte können hinzugefügt werden und die neuen Symptomkonzepte in Kombination mit bestehenden Symptomkonzepten genutzt werden, um ein neues Fehlerkonzept zu definieren.
- Die Wartbarkeit der Ontologie wird erhöht, weil Änderungen nur lokal vorgenommen werden müssen, um globale Auswirkungen zu haben. Stellt sich beispielsweise heraus, dass zur Spezifikation eines Symptomkonzepts ein zusätzliches Merkmal berücksichtigt werden sollte, muss nur die Definition dieses Symptomkonzepts adaptiert werden. Diese Änderung wirkt sich zur Laufzeit dann implizit – wie beabsichtigt – auch auf die automatisierte Auswertung derjenigen Fehler und Zustände aus, die mit Bezug auf dieses Symptom definiert sind.

Um eine hohe Schlussfolgerbarkeit zu erreichen, müssen diese Konzepte sowohl mit *notwendigen* als auch mit *hinreichenden* Bedingungen definiert werden. Dadurch wird Semantik formalisiert, die für automatische Ableitungen genutzt werden kann. Beispielsweise ist es dann möglich, eine Symptominstanz mit der zugehörigen Merkmalsinstanz zu assoziieren und mittels der *Konzept-Realisierung* (vergleiche Abschnitt 3.3.3) automatisiert die Art des Symptoms ableiten zu lassen.

Damit erfolgt die Definition von Symptom-, Fehler- und Zustandskonzepten analog zu den oben beschriebenen generischen Verarbeitungsschritten eines Zustandserkennungssystems: In den Konzeptdefinitionen wird das A-priori-Domänenwissen formalisiert, das für die Transformationen zwischen den verschiedenen Bezugsräumen nötig ist.

Die Kernkonzepte des Entwurfsmusters – Feature, Symptom, Fault, State und MonitoringSubject – werden im Folgenden im Detail erläutert. Für ihre Modellierung werden jeweils Beispiele angegeben, die auch in Abbildung 4.5 illustriert sind. Es wird mit dem Feature-Konzept begonnen, da die anderen Konzeptdefinitionen darauf aufsetzen.

4.4.2.1. Feature-Konzept

Das Feature-Konzept repräsentiert Merkmale (siehe Definition 4.4.2). Für jedes Merkmal, das von einer Kontextquelle lieferbar ist, wird eine Konzeptdefinition als Spezialisierung des Feature-Konzepts erstellt. Instanzen dieser Konzepte werden mittels der funktionalen Rolle refersToMonitoringSubject mit dem Überwachungsobjekt assoziiert, auf das sie sich beziehen.

Merkmale können in unterschiedlicher Form von Überwachungssystemen und anderen Kontextquellen geliefert werden: *Qualitative* Merkmale, wie z. B. „erhöhte Temperatur“ oder „kein Wert“, werden direkt durch entsprechende Konzeptdefinitionen (IncreasedTemperature, NoValue) modelliert. *Quantitative* Merkmale, wie z. B. 153 Kilo-Newton oder 235 Volt, müssen zunächst diskretisiert werden (z. B. „niedrige Kraft“,

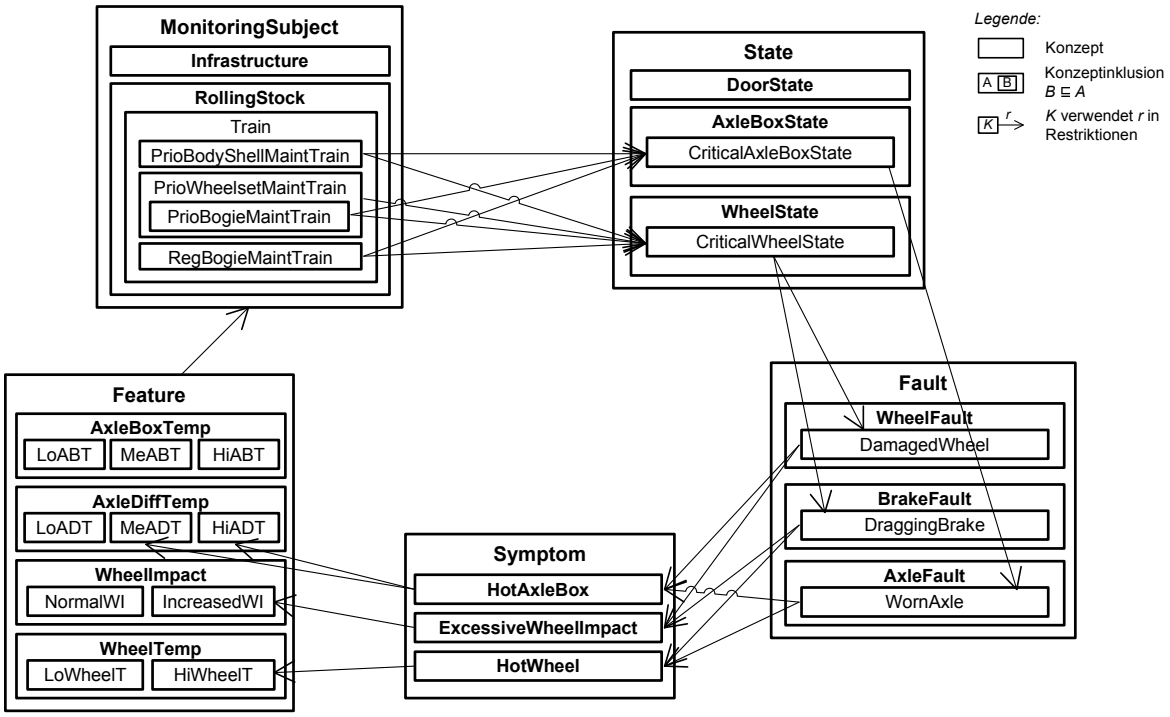

Abbildung 4.5.: Beispiele für den Einsatz des Entwurfsmusters zur Definition domänenspezifischer Merkmale, Symptome, Fehler, Zustände und Überwachungsobjekte.

„mittlere Kraft“, „hohe Kraft“). Für jede dieser Diskretisierungklassen wird anschließend eine entsprechende Konzeptdefinition angelegt (z. B. LowForce, MediumForce, HighForce). Dazu wird das Entwurfsmuster für *value partitions* eingesetzt [Rec05].

Außerdem kann es wichtig sein, die zeitliche Abfolge von Merkmalen festzuhalten. Der *absolute* Zeitpunkt für eine konkrete Merkmalsinstanz wird mittels der funktionalen Rolle hasTimestamp zu einer Timestamp-Instanz modelliert. Dazu wird das Entwurfsmuster zur Modellierung von Zeitpunkten in OWL verwendet [HP06]. Zusätzlich wird auch die *relative* zeitliche Ordnung repräsentiert. Dies geschieht mittels der Rollen precededByFeature und precededDirectlyByFeature, die analog zu dem Entwurfsmuster für Teil-Ganzes-Beziehungen modelliert sind [RW05]: precededByFeature ist transitiv definiert mit precededDirectlyByFeature als Unter-Rolle, die nicht transitiv definiert ist. Dadurch ist es bei der Definition von Symptom-Konzepten möglich, sowohl zu modellieren, dass ein Merkmal irgendwann zuvor aufgetreten sein muss, als auch, dass es in einem gewissen Zeitfenster aufgetreten sein muss.

Einige Beispiele für die Modellierung von Merkmalen sind in Abbildung 4.5 illustriert: Achslagertemperatur (AxleBoxTemp) wird diskretisiert in die Partitionen „hoch“ (Hi), „mittel“ (Me) und „niedrig“ (Lo). Das Überwachungssystem für Achslast (WheelImpactLoad) liefert bereits qualifizierte Merkmale und wird entsprechend repräsentiert als „erhöht“ (Increased) und „normal“ (Normal). Dies wird beschreibungslogisch wie folgt ausgedrückt:

$$\mathsf{AxleBoxTemp} \sqsubseteq \mathsf{Feature},\ \mathsf{AxleDiffTemp} \sqsubseteq \mathsf{Feature},\ \mathsf{WheelImpact} \sqsubseteq \mathsf{Feature},$$

$\mathsf{WheelTemp} \sqsubseteq \mathsf{Feature}$
$\mathsf{AxleBoxTemp}$, $\mathsf{AxleDiffTemp}$, $\mathsf{WheelImpact}$, $\mathsf{WheelTemp}$ sind paarweise disjunkt

$\mathsf{AxleBoxTemp} \equiv \mathsf{LoAxleBoxTemp} \sqcup \mathsf{MeAxleBoxTemp} \sqcup \mathsf{HiAxleBoxTemp}$
$\mathsf{LoAxleBoxTemp}$, $\mathsf{MeAxleBoxTemp}$, $\mathsf{HiAxleBoxTemp}$ sind paarweise disjunkt

$\mathsf{AxleDiffTemp} \equiv \mathsf{LoAxleDiffTemp} \sqcup \mathsf{MeAxleDiffTemp} \sqcup \mathsf{HiAxleDiffTemp}$
$\mathsf{LoAxleDiffTemp}$, $\mathsf{MeAxleDiffTemp}$, $\mathsf{HiAxleDiffTemp}$ sind paarweise disjunkt

$\mathsf{WheelImpact} \equiv \mathsf{IncreasedWheelImpact} \sqcup \mathsf{NormalWheelImpact}$
$\mathsf{IncreasedWheelImpact}$, $\mathsf{NormalWheelImpact}$ sind paarweise disjunkt

$\mathsf{WheelTemp} \equiv \mathsf{LoWheelTemp} \sqcup \mathsf{HiWheelTemp}$
$\mathsf{LoWheelTemp}$, $\mathsf{HiWheelTemp}$ sind paarweise disjunkt

Bei der Instantiierung dieser Merkmalskonzepte für konkrete Daten werden mittels Zusicherungen der Bezug zum Überwachungsobjekt und der Zeitbezug hergestellt.

$\mathsf{IncreasedWheelImpact}(i1)$, $\mathsf{Train}(o1)$, $\mathsf{refersToMonitoringSubject}(i1, o1)$

$\mathsf{hasTimestamp}(i1, t1)$, $\mathsf{Timestamp}(t1)$,
$\mathsf{owl\text{-}time{:}inXSDDateTime}(t1, 2008\text{-}06\text{-}01T10{:}30{:}00\text{-}5{:}00)$

$\mathsf{precededDirectlyByFeature}(i1, i2)$, $\mathsf{NormalWheelImpact}(i2)$, ...

4.4.2.2. Symptom-Konzept

Das Symptom-Konzept repräsentiert Symptome (siehe Definition 4.4.3). Für jedes Symptom wird eine Konzeptdefinition als Spezialisierung des Symptom-Konzepts erstellt. Gemäß Definition spiegelt ein Symptom eine Verhaltensabweichung wider und kann durch ein oder mehrere Merkmale gekennzeichnet sein. Dementsprechend werden Symptomkonzepte mit Bezug auf Merkmalskonzepte definiert. Dieser Bezug wird bei Instanzen mittels der funktionalen Rolle refersToFeature hergestellt.

Zur Modellierung von Symptom-Konzepten steht die volle Ausdrucksstärke der $\mathcal{SHIN}$-Beschreibungslogik zur Verfügung. Beispiele für die Symptome HotWheel, HotAxleBox und ExcessiveWheelImpact sind in Abbildung 4.5 illustriert und werden hier formal vorgestellt:

$\mathsf{HotWheel} \sqsubseteq \mathsf{Symptom}$, $\mathsf{HotAxleBox} \sqsubseteq \mathsf{Symptom}$, $\mathsf{ExcessiveWheelImpact} \sqsubseteq \mathsf{Symptom}$

$\mathsf{HotWheel} \equiv (\exists\ \mathsf{refersToFeature.HiWheelTemp} \sqcap \forall\ \mathsf{refersToFeature.HiWheelTemp})$

$\mathsf{HotAxleBox} \equiv (\exists\ \mathsf{refersToFeature.}(\mathsf{HiAxleBoxTemp} \sqcup \mathsf{MeAxleDiffTemp} \sqcup \mathsf{HiAxleDiffTemp})) \sqcap (\forall\ \mathsf{refersToFeature.}(\mathsf{HiAxleBoxTemp} \sqcup \mathsf{MeAxleDiffTemp} \sqcup \mathsf{HiAxleDiffTemp}))$

$\mathsf{ExcessiveWheelImpact} \equiv (\exists\ \mathsf{refersToFeature.}(\mathsf{IncreasedWheelImpact} \sqcup \exists\ \mathsf{precededDirectlyByFeature.IncreasedWheelImpact}))) \sqcap (\forall\ \mathsf{refersToFeature.}(\mathsf{IncreasedWheelImpact} \sqcup \exists\ \mathsf{precededDirectlyByFeature.IncreasedWheelImpact})))$

Die Symptom-Konzepte werden nicht als disjunkt definiert, da es durchaus möglich sein kann, dass mehrere Symptome gleichzeitig auftreten.

4.4.2.3. Fault-Konzept

Das Fault-Konzept repräsentiert Fehler (siehe Definition 4.4.4). Für jeden Fehler wird eine Konzeptdefinition als Spezialisierung des Fault-Konzepts erstellt. Gemäß Definition zeigt ein Fehler die Abweichung einer Eigenschaft des Überwachungsobjekts vom Sollwert an. Deshalb werden Fehlerkonzepte mit Bezug auf Symptomkonzepte definiert. Zwischen Instanzen wird der Bezug mittels der funktionalen Rolle refersToSymptom hergestellt.

Auch hier steht die Ausdrucksstärke der $\mathcal{SHIN}$-Beschreibungslogik zur Verfügung. Als Beispiele sind WornAxle, DraggingBrake und DamagedWheel in Abbildung 4.5 illustriert und werden hier formal dargestellt:

$$\mathsf{WornAxle} \sqsubseteq \mathsf{Fault},\ \mathsf{DraggingBrake} \sqsubseteq \mathsf{Fault},\ \mathsf{DamagedWheel} \sqsubseteq \mathsf{Fault}$$

$$\begin{aligned}\mathsf{WornAxle} \equiv (&\exists\, \mathsf{refersToSymptom}.(\mathsf{HotAxleBox} \sqcap (\neg \mathsf{HotWheel}) \sqcap \\ &\quad (\exists\, \mathsf{refersToFeature}.(\exists\, \mathsf{precededByFeature}. \\ &\quad\quad (\exists\, \mathsf{indicatesSymptom}.\mathsf{HotAxleBox})))) \sqcap \\ &\forall\, \mathsf{refersToSymptom}.(\mathsf{HotAxleBox} \sqcap (\neg \mathsf{HotWheel}) \sqcap \\ &\quad (\exists\, \mathsf{refersToFeature}.(\exists\, \mathsf{precededByFeature}. \\ &\quad\quad (\exists\, \mathsf{indicatesSymptom}.\mathsf{HotAxleBox})))))\end{aligned}$$

$$\begin{aligned}\mathsf{DraggingBrake} \equiv (&\exists\, \mathsf{refersToSymptom}.((\neg \mathsf{ExcessiveWheelImpact}) \sqcap \mathsf{HotWheel}) \sqcap \\ &\forall\, \mathsf{refersToSymptom}.((\neg \mathsf{ExcessiveWheelImpact}) \sqcap \mathsf{HotWheel}))\end{aligned}$$

$$\begin{aligned}\mathsf{DamagedWheel} \equiv (&\exists\, \mathsf{refersToSymptom}.((\neg \mathsf{HotAxleBox}) \sqcap \mathsf{ExcessiveWheelImpact}) \sqcap \\ &\forall\, \mathsf{refersToSymptom}.((\neg \mathsf{HotAxleBox}) \sqcap \mathsf{ExcessiveWheelImpact}))\end{aligned}$$

Diese Beispiele zeigen, wie unterschiedliche Symptomkonzepte kombiniert werden, um Fehlerkonzepte zu definieren. Fehlerkonzepte werden hier ebenso nicht als disjunkt modelliert, weil es möglich sein soll, dass mehrere Fehler gleichzeitig erkannt werden.

4.4.2.4. State-Konzept

Das State-Konzept repräsentiert technische Zustände (siehe Definition 4.4.5). Für jeden Zustand wird eine Konzeptdefinition als Spezialisierung des State-Konzepts erstellt. Gemäß Definition stellt ein Zustand die (Einschränkung der) Funktionsfähigkeit eines Überwachungsobjekts vor dem Hintergrund von Fehlern und Schäden dar. Dementsprechend werden Zustandskonzepte mit Bezug auf Fehlerkonzepte definiert. Auf Instanzebene wird dieser Bezugs mittels der funktionalen Rolle refersToFault hergestellt.

Auch hier kann wieder die Ausdrucksstärke der $\mathcal{SHIN}$-Beschreibungslogik genutzt werden. In Abbildung 4.5 sind z. B. CriticalWheelState und CriticalAxleBoxState dargestellt und werden im Folgenden formalisiert:

$$\mathsf{CriticalAxleBoxState} \sqsubseteq \mathsf{State},\ \mathsf{CriticalWheelState} \sqsubseteq \mathsf{State}$$

$$\mathsf{CriticalAxleBoxState} \equiv (\exists\, \mathsf{refersToFault}.\mathsf{WornAxle} \sqcap \forall\, \mathsf{refersToFault}.\mathsf{WornAxle})$$

$$\begin{aligned}\mathsf{CriticalWheelState} \equiv (&\exists\, \mathsf{refersToFault}.(\mathsf{DamagedWheel} \sqcup \mathsf{DraggingBrake}) \sqcap \\ &\forall\, \mathsf{refersToFault}.(\mathsf{DamagedWheel} \sqcup \mathsf{DraggingBrake}))\end{aligned}$$

4.4.2.5. MonitoringSubject-Konzept

Das MonitoringSubject-Konzept repräsentiert Überwachungsobjekte (siehe Definition 4.4.1). Je nach Gesamtzustand werden Überwachungsobjekte in unterschiedliche Überwachungsobjektkonzepte klassifiziert, die als Spezialisierungen des MonitoringSubject-Konzept definiert sind. Sie werden deshalb mit Bezug auf Zustandskonzepte modelliert. Der Bezug zwischen Überwachungsobjekt und Zuständen wird auf Instanzebene mittels der Rolle hasState hergestellt. hasState ist dabei nicht als funktional modelliert, weshalb eine Überwachungsobjektinstanz mit beliebig vielen Zustandsinstanzen verbunden sein kann. Wie vorher schon beschrieben, sind außerdem beliebig viele Merkmalsinstanzen mit einer Überwachungsobjektinstanz über die (funktionale) Rolle refersToMonitoringSubject assoziiert.

Zur Definition von MonitoringSubject-Konzepten kann prinzipiell die Ausdruckstärke der $\mathcal{SHIN}$-Beschreibungslogik verwendet werden. (Zur Verwendung in einer eingeschränkt verteilten Wissensbasis müssen jedoch einige Einschränkungen beachtet werden, die in Abschnitt 5.3.3.4 erläutert werden.) Abbildung 4.5 illustriert Regular- und PriorityBogieMaintenanceTrain sowie PriorityBodyShell- und PriorityWheelsetMaintenanceTrain, welche wie folgt formal modelliert sind:

$$\begin{aligned}
&\mathsf{PriorityBogieMaintenanceTrain} \sqsubseteq \mathsf{MonitoringSubject},\\
&\mathsf{RegularBogieMaintenanceTrain} \sqsubseteq \mathsf{MonitoringSubject},\\
&\mathsf{PriorityBodyShellMaintenanceTrain} \sqsubseteq \mathsf{MonitoringSubject}
\end{aligned}$$

$$\mathsf{PriorityBogieMaintenanceTrain} \equiv (\exists\, \mathsf{hasState.CriticalWheelState} \sqcap \exists\, \mathsf{hasState.CriticalAxleBoxState})$$

$$\mathsf{RegularBogieMaintenanceTrain} \equiv (\exists\, \mathsf{hasState.CriticalAxleBoxState} \sqcap \forall\, \mathsf{hasState.}(\neg\mathsf{CriticalWheelState}))$$

$$\mathsf{PriorityWheelsetMaintenanceTrain} \equiv (\exists\, \mathsf{hasState.CriticalWheelState})$$

$$\mathsf{PriorityBodyShellMaintenanceTrain} \equiv (\geqslant_2 \mathsf{hasState} \sqcap \forall\, \mathsf{hasState.}(\neg\mathsf{CriticalWheelState}) \sqcap \forall\, \mathsf{hasState.}(\neg\mathsf{CriticalAxleBoxState}))$$

Auf Basis dieser Konzeptdefinitionen kann also für Überwachungsobjektinstanzen beschreibungslogisch gefolgert werden, zu welchen Konzepten sie gehören und damit, in welchem Zustand sich das beschriebene Überwachungsobjekt befindet.

4.4.3. Anwendung des Entwurfsmusters zur Laufzeit

Die bisherige Beschreibung des Entwurfsmusters bezog sich auf die beschreibungslogische Definition von Konzepten in der TBox. Um die so formalisierte Semantik für automatisches Schlussfolgern nutzen zu können, muss die Wissensbasis zur Laufzeit geeignet aufgebaut werden. Dazu müssen in der ABox geeignete Instanzen erzeugt und mittels Rollen-Zusicherungen in Beziehung gesetzt werden.

Instanzen von Überwachungsobjektkonzepten stellen das Referenzsystem für die Zustandsüberwachung dar. Je nach Überwachungsaufgabe und Domäne müssen folglich

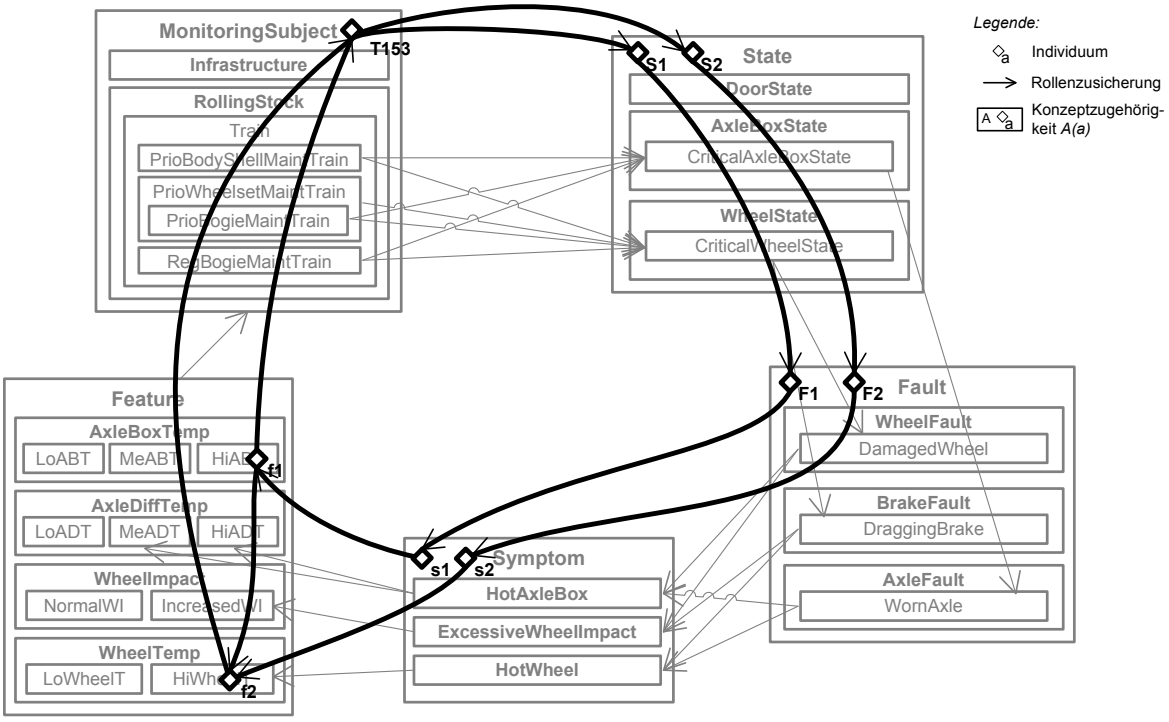

Abbildung 4.6.: Explizite Zusicherungen in der Wissensbasis.

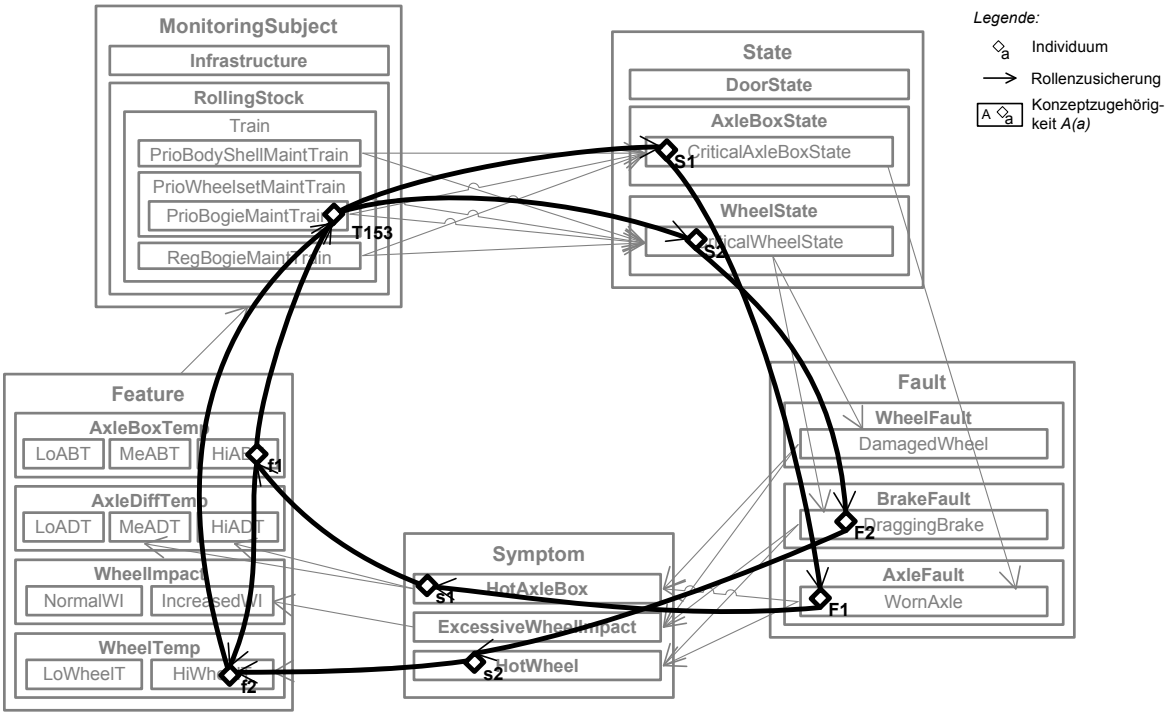

Abbildung 4.7.: Implizite Aussagen in der Wissensbasis, die gefolgert werden können.

Instanzen für Gleisabschnitte, Weichen, Waggons oder Radsätze bzw. Transformatoren, Leitungen oder Schalter angelegt werden. Diese müssen eindeutig referenzierbar sein, z. B. mit einem global eindeutigen Bezeichner. Damit mehrere Kontextquellen Merkmale zu demselben Überwachungsobjekt liefern können, müssen sie denselben Bezeichner verwenden.[4]

Liefert eine Kontextquelle nun ein Merkmal zu einem Überwachungsobjekt, werden folgende Informationen in Form von ABox-Zusicherungen explizit zu der Wissensbasis hinzugefügt (siehe Illustration in Abbildung 4.6): Eine Instanz des entsprechenden Merkmalskonzepts wird erzeugt, z. B. eine HiWheelTemp-Instanz. Sie wird mit der Rolle refersToMonitoringSubject mit der entsprechenden Überwachungsobjektinstanz assoziiert. Zusätzlich wird die Merkmalsinstanz mit einem Zeitstempel versehen und mittels precededDirectlyByFeature mit der vorangegangenen Merkmalsinstanz in eine relative Reihenfolge gebracht. Gleichzeitig werden Symptom-, Fehler- und Zustandsinstanzen erzeugt und mit den Merkmals- und Überwachungsobjektinstanzen assoziiert. Sie werden in der ABox jedoch lediglich als Instanzen der Konzepte Symptom, Fault und State zugesichert, weil nicht bekannt ist, um welche spezifischeren Konzepte es sich handelt. Die Zugehörigkeit zu spezifischeren Konzepten ergibt sich *implizit* aus der in der TBox formalisierten Semantik und kann deshalb automatisiert abgeleitet werden.

Dies wird in Abbildung 4.7 für die Train-Instanz $T153$ illustriert: Für die Symptom-Instanz $s1$ folgt logisch HotAxleBox($s1$), d. h. es kann bewiesen werden, dass $s1$ auch eine Instanz des HotAxleBox-Konzepts ist. Bei der Fault-Instanz $F1$ kann abgeleitet werden, dass sie eine Instanz des WornAxle-Konzepts ist; bei der State-Instanz $S1$, dass sie eine Instanz des CriticalAxleBoxState-Konzepts ist. Daraus folgt schließlich, dass die Train-Instanz $T153$ gleichzeitig eine Instanz des PriorityBogieMaintenanceTrain-Konzepts sein muss. Diese Erkenntnis, die anhand der in der TBox formalisierten Semantik abgeleitet wurde, kann beispielsweise die Grundlage für ein System zur Wartungsoptimierung bilden.

4.4.4. Systematische Dokumentation des Entwurfsmusters

Zur zusammenfassenden Dokumentation des Ontologie-Entwurfsmusters für Infrastrukturzustände wird im Folgenden die Systematik der *Semantic Web Best Practices and Deployment Working Group* des W3C übernommen [SWBP08].

Aspekt	*Inhalt*
Allgemeiner Sachverhalt	Zur Überwachung von Infrastrukturnetzen müssen Infrastrukturzustände mit Bezug zu Kontextinformationen modelliert werden. Die Modellierung soll richtig, verständlich, schlussfolgerbar, erweiterbar und wartbar sein. Das zu modellierende Wissen ist in der Praxis jedoch über mehrere Domänenexperten verteilt, die in der Regel über keine Modellierungserfahrung verfügen. Um dennoch Modellierungen mit den gewünschten

[4] Diese Funktionalität wird von *Namensdiensten*, wie dem im Internet verwendeten *Domain Name System* (DNS), zur Verfügung gestellt. Da zu diesem Gebiet umfangreiche Arbeiten existieren – beispielweise [ABC+05] —, beschränkt sich diese Arbeit darauf, auf diese zu verweisen.

Aspekt	*Inhalt*
...	Eigenschaften zu erhalten, soll Domänenexperten ein geeignetes Entwurfsmuster an die Hand gegeben werden.
Beispiele für Anwendungsfälle	Ein Experte für Heißläuferortungsanlagen möchte auf Basis der Merkmale Achslager- und Radtemperatur, die eine Heißläuferortungsanlage liefert, unterschiedliche Zustände für einen Zug modellieren. Ein Experte für Radsätze modelliert relevante Zustände des Radsatzes eines Waggons und möchte als Kontextinformationen dazu Merkmale, die von Achslastgebern und Heißläuferortungsanlagen geliefert werden, sowie Wetterinformationen berücksichtigen.
Notation	Konzepte werden durch abgerundete Rechtecke repräsentiert, Individuen durch Rauten. Rollen werden durch Pfeile dargestellt. Es wird die übliche Beschreibungslogik-Syntax verwendet (vergleiche Abschnitt 3.3).
Ansatz	Merkmale und Zustände werden als Konzepte mit notwendigen und hinreichenden Bedingungen spezifiziert. Zusätzlich werden Konzepte für Symptome und Fehler eingeführt. Merkmalskonzepte abstrahieren von konkreten Kontextquellen wie z. B. Überwachungssystemen. Für jedes abstrakte Merkmal wird ein spezialisiertes Merkmalskonzept eingeführt. Spezialisierte Symptomkonzepte werden mit Bezug auf Merkmalskonzepte definiert; spezialisierte Fehlerkonzepte mit Bezug auf Symptomkonzepte und spezialisierte Zustandskonzepte mit Bezug auf Fehlerkonzepte. Schließlich werden spezialisierte Überwachungsobjektkonzepte definiert, die sich auf Zustandskonzepte beziehen. Mittels Konzept-Realisierung kann die formalisierte Semantik automatisiert ausgewertet und am Ergebnis der Zustand eines Überwachungsobjekts abgelesen werden.
Abwägungen	Anhand der in Abschnitt 4.4.1 gegebenen Definitionen muss bei der Modellierung klar zwischen Merkmalen, Symptomen und Fehlern unterschieden werden. In der Regel liefern Überwachungssysteme Merkmale. Führt das Überwachungssystem selbst bereits eine Vorverarbeitung durch, kann es sich bei den gelieferten Kontextinformationen jedoch auch um Symptome oder sogar Fehler handeln. Für die Modellierung von Überwachungsobjektkonzepten müssen zur verteilten Auswertbarkeit die Bedingungen für eingeschränkt verteilte Wissensbasen beachtet werden (siehe Abschnitt 5.3.3).

4.5. Ontologie-Architekturmuster zur verteilten Entwicklung

Mit Hilfe des zuvor vorgestellten Entwurfsmusters, können Domänenexperten qualitativ hochwertige Domänenontologien erstellen. Für den Einsatz in einem Zustandsüberwa-

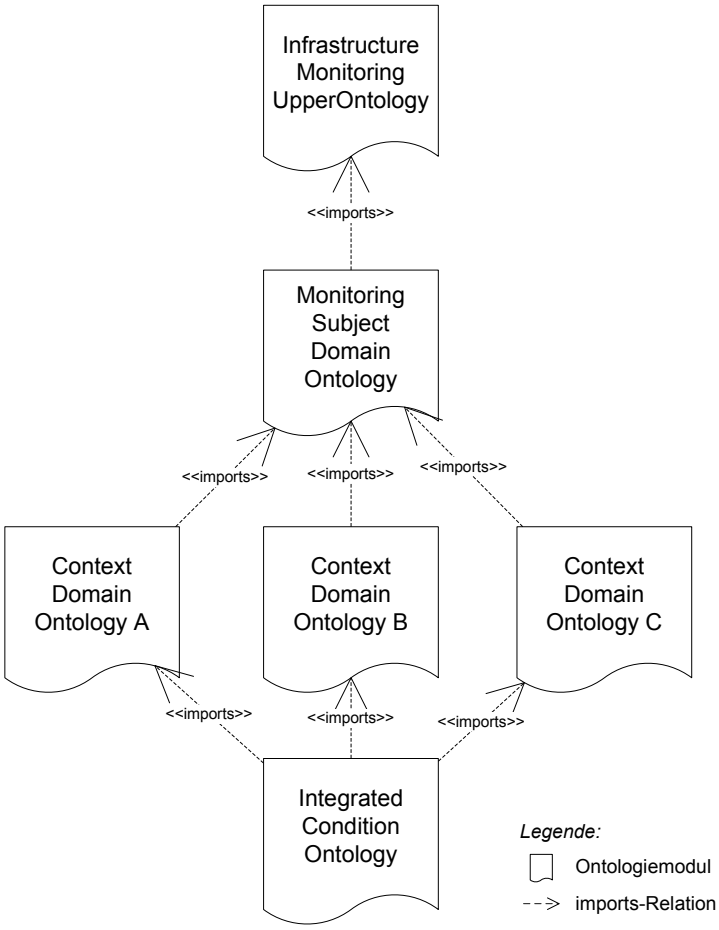

Abbildung 4.8.: Illustration des Architekturmusters zur Integration mehrerer Domänenontologien.

chungssystem, das mehrere Domänen berücksichtigt, müssen diese Domänenontologien zu einer Gesamtontologie integriert werden. Dabei dürfen keine logischen Widersprüche auftreten, und die formalisierte Semantik muss erhalten bleiben. Um dies zu erreichen, wird im Folgenden ein Architekturmuster entwickelt. Abbildung 4.8 illustriert seine Struktur: Die Gesamtontologie wird hierarchisch in Module gegliedert, welche sich gegenseitig importieren. Ontologie A importiert Ontologie B bedeutet, dass A um alle TBox-Axiome und ABox-Zusicherungen von B erweitert wird.

Die Wurzel der Ontologiemodule bildet die *InfrastructureConditionUpperOntology*. Diese definiert die Kernkonzepte und -rollen des zuvor beschriebenen Entwurfsmusters für Infrastrukturzustände und ist unabhängig vom Infrastrukturtyp.

Sie wird ergänzt um eine *MonitoringSubjectDomainOntology*. Diese ist spezifisch für einen bestimmten Infrastrukturtyp, wie z. B. das Schienen- oder das Stromnetz. Sie definiert Überwachungsobjektkonzepte, die für diesen Infrastrukturtyp relevant sind, und spezialisiert dazu das MonitoringSubject-Konzept. Damit bietet sie eine einheitliche Sammlung von Überwachungsobjektkonzepten, auf die sich die verschiedenen Domänenontologien beziehen können.

Domänenexperten erstellen für jede relevante Domäne eine *ContextDomainOntology*.

Diese importiert die *MonitoringSubjectDomainOntology* und bezieht sich somit auf deren einheitliche Sammlung von Überwachungsobjektkonzepten. Da sie außerdem indirekt die *InfrastructureConditionUpperOntology* importiert, wird sichergestellt, dass sie nicht im Widerspruch zum Entwurfsmuster für Infrastrukturzustände steht.

Zur Definition domänenübergreifender Konzepte können schließlich *IntegratedConditionOntologies* erstellt werden. Diese importieren mehrere *ContextDomainOntologies*. Ihre Konzeptdefinitionen können sich deshalb auf die Konzeptdefinitionen *mehrerer* Domänenontologien beziehen. So ist das Überwachungsobjektkonzept RegularBogieMaintenanceTrain in Abschnitt 4.4.2 beispielsweise mit Bezug auf Zustandskonzepte der Achslasten- und der Heißläufer-Domäne definiert.

Für jedes dieser Ontologiemodule (und für jede Kombination von ihnen) gilt, dass die logische Konsistenz automatisiert überprüft werden kann. Dadurch können Modellierungswidersprüche und -fehler frühzeitig erkannt und eingegrenzt werden.

4.6. Bewertung des gewählten Modellierungsansatzes

Anhand der in Abschnitt 4.1 aufgestellten Anforderungen wird im Folgenden die Effektivität des hier gewählten musterbasierten Modellierungsansatzes beurteilt.

A4.1 Genauigkeit. Die Genauigkeit und Korrektheit der modellierten Zusammenhänge muss von den Domänenexperten sichergestellt werden. Das Entwurfsmuster gibt ihnen durch die Differenzierung zwischen Merkmalen, Symptomen, Fehlern und Zuständen jedoch Leitlinien an die Hand, um die Modellierung eines komplexen Infrastrukturzustands systematisch zu strukturieren und damit Modellierungsfehler zu vermeiden. So verhindert die Definition der refersToFault-Rolle beispielsweise, dass ein State-Konzept mit Bezug zu Symptom-Konzepten statt – wie vorgesehen – Fault-Konzepten definiert wird. Zusätzlich kann die beschreibungslogische Grundlage dazu genutzt werden, die *logische* Konsistenz zu prüfen, d. h. logische Widersprüche automatisiert zu erkennen. Dies ist vor allem dann essenziell, wenn mehrere Personen an der Ontologieerstellung beteiligt sind.

A4.2 Verständlichkeit. Die Verständlichkeit der erstellten Ontologie für sowohl Ontologieentwickler als auch -nutzer wird durch das Entwurfsmuster gewährleistet, weil es nur Begriffe und Konzepte verwendet, die aus etablierten, internationalen Standards im Bereich der Zustandsüberwachung abgeleitet wurden.

A4.3 Schlussfolgerbarkeit. Wie die Analyse in Abschnitt 4.3 ergeben hat, kann durch den Einsatz von Entwurfsmustern besonders die Schlussfolgerbarkeit der entstehenden Ontologie erhöht werden. Die Beschreibung des Musters hat gezeigt, wie die Semantik von Symptomen, Fehlern und Zuständen in den Konzeptdefinitionen formalisiert werden kann. Abschnitt 4.4.3 hat dann an einem Beispiel beschrieben, wie die Konzept-Realisierung dieses implizite Wissen zur Laufzeit für automatisches Schlussfolgern nutzt.

A4.4 Erweiterbarkeit. Erweiterbarkeit der entstehenden Ontologie war ein Grund für die Einführung von Symptom- und Fehlerkonzepten im Entwurfsmuster: Beim Hinzukommen zusätzlicher Kontextquellen können so auch zur Laufzeit problemlos neue Merkmalskonzepte ergänzt und in die Definition von Symptomkonzepten übernommen werden. Auf dieselbe Weise können zusätzliche Symptom-, Fehler- und Zustandskonzepte hinzugefügt bzw. eingebunden werden (siehe Beispiel in Abschnitt 4.4.2). Auch hier zahlt sich aus, dass die logische Konsistenz der erweiterten Ontologie automatisiert geprüft und damit Modellierungsfehler rechtzeitig erkannt werden können.

A4.5 Wartbarkeit. Diese Eigenschaft ergibt sich aus den Eigenschaften der Verständlichkeit und der Erweiterbarkeit: Die Verständlichkeit stellt sicher, dass bei der Wartung die Änderungen an der richtigen Stelle vorgenommen werden. Die Erweiterbarkeit ermöglicht es, bestehende Konzeptdefinitionen auch zur Laufzeit des Systems an die neuen Anforderungen anzupassen. Da lediglich die in der TBox formalisierte Semantik geändert wird, können außerdem die bereits vorhanden Zusicherungen in der ABox (Merkmale von Kontextquellen) und bisherige Anfragen weiter genutzt werden. Lediglich die Ergebnisse ändern sich wie beabsichtigt entsprechend der angepassten Semantik (siehe ebenfalls das Beispiel in Abschnitt 4.4.2). Mittels automatisierter Prüfung der logischen Konsistenz können auch hier Modellierungsfehler vermieden werden.

4.7. Zusammenfassung

Dieses Kapitel entwickelte einen Ansatz, der Domänenexperten bei der Erstellung qualitativ hochwertiger Domänenontologien unterstützt. Dazu wurden Kriterien an die zu erstellenden Ontologien formuliert. Als effektivster Ansatz, um diese Eigenschaften zu gewährleisten, wurden Modellierungsmuster identifiziert. Zur Modellierung von Infrastrukturzuständen wurde ein Entwurfsmuster entwickelt. Dieses wurde aus zentralen Begriffen und Zusammenhängen abgeleitet, die in internationalen Standards zur Zustandsüberwachung und -diagnostik definiert sind. Mit diesem Entwurfsmuster können Domänenexperten qualitativ hochwertige Ontologien für Kontextinformationen und Infrastrukturzustände ihrer Domäne erstellen. Zur Integration dieser Teilontologien zu einer konsistenten Gesamtontologie wurde außerdem ein Architekturmuster entwickelt. Dieses ermöglicht es, mehrere Domänenontologien zu einer Systemontologie zu integrieren.

Diese Systemontologie stellt also ein Wissensmodell dar. Es kann für unterschiedlichste Formen des Reasonings genutzt werden (vergleiche Abschnitt 3.1). Bei logikbasierten Reasoning-Formen unterscheidet man zwischen Deduktion, Abduktion und Induktion (vergleiche Abschnitt 3.2.2). Davon ist die Deduktion die einzige Form des sicheren Schließens. Da sicheres Schließen für die Überwachung von Infrastrukturnetzen unverzichtbar ist, wird im Folgenden untersucht, wie deduktive Ableitungen auf Basis einer solchen Systemontologie getroffen werden können.

5. Reasoning über verteilte Kontextinformationen

Im vorherigen Kapitel wurden Entwurfsmuster entwickelt, die Domänenexperten die Erstellung hochwertiger beschreibungslogischer Ontologien für Kontextinformationen und Infrastrukturzustände erleichtern. Ein zentrales Kriterium war dabei Schlussfolgerbarkeit. Dieses Kapitel befasst sich nun mit der Nutzung dieser formalisierten Semantik für deduktive Schlussfolgerungen als eine wichtige Form des Reasonings. Dabei besteht die Kernherausforderung in Infrastrukturnetzen darin, dass die Kontextinformationen verteilt anfallen und es nicht praktikabel ist, sie zu zentralisieren. Trotzdem soll erreicht werden, dass Anfragen beantwortet werden können, als wären die Daten zentral verfügbar.

Dazu werden zunächst die Anforderungen und die formale Problemstellung in Abschnitt 5.1 formuliert und in Abschnitt 5.2 bestehende Reasoningansätze untersucht. Da diese nicht alle Anforderungen erfüllen können, wird in Abschnitt 5.3 das Konzept eines verteilten Reasoning-Systems entworfen. Dieses wurde in [FBLP09, FB08, FB07, FHP+06, FB06] entwickelt. Es wird gezeigt, wie damit – für unterschiedliche Arten der Verteilung einer Wissensbasis – Verfahren zur Beantwortung konjunktiver Anfragen umgesetzt werden. Insbesondere wird für sogenannte eingeschränkt verteilte Wissensbasen ein korrekter und vollständiger Algorithmus zur Beantwortung konjunktiver Anfragen angegeben. Nach der Beschreibung der Implementierung in Abschnitt 5.4 wird der Ansatz in Abschnitt 5.5 zusätzlich experimentell evaluiert.

5.1. Anforderungen an Reasoning über verteilte Kontextinformationen

In Kapitel 2 wurden bereits die wichtigsten Merkmale einer Infrastrukturüberwachung dargestellt: Die Größe von Infrastrukturen sowie ihre Komplexität führen dazu, dass viele verschiedene Organisation an Überwachung und Wartung beteiligt sind. Jede dieser Organisationen erfasst und speichert Daten über bestimmte Teile der Infrastruktur getrennt und unabhängig von den anderen. Zur Erkennung komplexer Infrastrukturzustände müssen die erfassten Daten jedoch in ihrer Gesamtheit ausgewertet werden. Darum werden Verfahren zum Reasoning über verteilte Kontextinformationen benötigt. Der folgende Abschnitt identifiziert im Detail die damit verbundenen Anforderungen. Sie werden anschließend in Abschnitt 5.1.2 formalisiert.

5.1.1. Allgemeine Anforderungen

Bei der Anforderungsanalyse in der Softwaretechnik hat es sich bewährt, zwischen *funktionalen* und *nicht-funktionalen* Anforderungen zu unterschieden [BD04]. Funktionale Anforderungen bezeichnen Aufgaben, die das System für seine Nutzer unterstützen muss, wie z. B. die korrekte Beantwortung bestimmter Anfragen. Hier geht es also darum, *was* das System tut. Im Gegensatz dazu legen nicht-funktionale Anforderungen die Eigenschaften des Systems fest, beispielweise dass die Antwort innerhalb eines bestimmten Zeitrahmens erfolgen muss. Dabei geht es also darum, *wie* das System etwas tut.

Anhand der in Abschnitt 2.3 auf Seite 20 identifizierten Anforderungen an die Zustandsüberwachung für Infrastrukturnetze, werden im Folgenden Anforderungen an das Reasoning über verteilte Kontextinformationen abgeleitet.

Funktionale Anforderungen. Zunächst wird analysiert, welche Funktionen das zu entwickelnde System zur Verfügung stellen muss.

A5.1.1 *Berücksichtigung von verteilt vorliegenden Kontextinformationen zu demselben Überwachungsobjekt*: Zur Zustandsüberwachung von Infrastrukturnetzen müssen Kontextinformationen aus unterschiedlichen Quellen integriert werden (A2.1). Dazu kommt, dass diese Kontextinformationen in der Regel von unterschiedlichen Organisationen erfasst (A2.5) und aufgrund der geographischen Ausdehnung von Infrastrukturnetzen an verschiedenen Orten produziert werden (A2.6).

A5.1.2 *Verwendung einer einheitlichen Systemontologie*: Das Wissen über die relevanten Zusammenhänge ist zwar über mehrere Domänenexperten verteilt (A2.3). Dennoch sollen Kontextinformationen von unterschiedlichen Organisationen auf konsistente Weise integriert (A2.1, A2.5) und dabei auch komplexe Beziehungen berücksichtigt werden (A2.2). Dazu wird eine einheitliche Systemontologie benötigt.

A5.1.3 *Unterstützung für korrektes und vollständiges beschreibungslogisches Schlussfolgern*: Um das Wissen, das in der Systemontologie formalisiert ist, zur Integration und Ableitung komplexer Zustände nutzen zu können (A2.1, A2.2), muss beschreibungslogisches Reasoning unterstützt werden. Dazu müssen die grundlegenden Reasoning-Dienste (siehe Abschnitt 3.3.3) zur Verfügung gestellt werden, und die gelieferten Antworten müssen im logischen Sinne korrekt und vollständig sein (A2.4, A2.7).

A5.1.4 *Beantwortung konjunktiver Anfragen*: Zur Integration und Ableitung komplexer Zustände (A2.1, A2.2) reichen die grundlegenden Reasoning-Dienste nicht aus. Sie sollen in Form von konjunktiven Anfragen kombinierbar sein, um komplexere Anfragen formulieren zu können.

Nicht-funktionale Anforderungen. Aus den in Abschnitt 2.3 identifizierten Anforderungen lassen sich außerdem Eigenschaften des zu entwickelnden Systems in Form von nicht-funktionalen Anforderungen ableiten.

A5.2.1 *Verteilte Haltung der Kontextinformationen ohne zentrale Instanz*: Die zu integrierenden Kontextinformationen werden von vielen verschiedenen, räumlich verteilten Kontextquellen produziert (A2.6). Zudem werden sie von unterschiedlichen Organisationen erfasst und verwaltet (A2.5). Aufgrund der zu übertragenden Datenmengen und strategischer Überlegungen der Organisationen kann nicht angenommen werden, dass Kontextinformationen zentralisiert werden können. Deshalb muss das zu entwickelnde System von einer verteilten Haltung der Kontextinformationen ohne zentrale Instanz ausgehen.

A5.2.2 *Unabhängigkeit von Reasoning-Verfahren und -Implementierungen*: Infrastrukturnetze zeichnen sich durch eine lange Lebensdauer aus, die den permanenten Ausbau und die Anpassung an neue Rahmenbedingung erfordert (A2.7). Dabei eingesetzte Systeme müssen besonders zukunftstüchtig sein. Aufgrund der vielen beteiligten Organisationen (A2.5) ist es außerdem nicht möglich, eine einheitliche Implementierung des Systems durchzusetzen. Anstatt sich auf bestimmte Reasoning-Verfahren und -Implementierungen festzulegen, soll Zukunftstüchtigkeit und Flexibilität dadurch erreicht werden, dass lediglich Komponenten und ihre Schnittstellen spezifiziert werden. So bleibt die konkrete Umsetzung der Komponenten den unterschiedlichen Organisationen überlassen. Sie kann im Laufe der Zeit unter Einhaltung der Schnittstellen auch angepasst werden, ohne Auswirkungen auf andere Komponenten des Systems zu haben.

5.1.2. Formalisierung der Anforderungen

Die im vorigen Abschnitt identifizierten Anforderungen werden im Folgenden mit den in Abschnitt 3.3.2 eingeführten Begriffen eines beschreibungslogisch wissensbasierten Systems formalisiert.

Anforderungen A5.1.1 und A5.1.2 implizieren, dass dem System eine *verteilte Wissensbasis* mit einheitlicher TBox und verteilter ABox zugrunde liegt:

Definition 5.1.1 (Verteilte Wissensbasis $\mathfrak{K}$). *Sei $\mathcal{T}$ eine TBox und $\mathcal{A}_j$ eine (Teil-) ABox für $1 \leq j \leq n$. Eine* verteilte Wissensbasis $\mathfrak{K}$ *ist definiert als $\mathfrak{K} = (\mathcal{T}, \mathfrak{A})$ mit $\mathfrak{A} = \bigcup_j \mathcal{A}_j$ bzw. $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ mit Teil-Wissensbasis $\mathcal{K}_j = (\mathcal{T}, \mathcal{A}_j)$, $1 \leq j \leq n$.*

Eine Interpretation $\mathcal{I}$ ist ein Modell von $\mathfrak{K}$ ($\mathcal{I} \models \mathfrak{K}$), falls gilt $\mathcal{I} \models \mathcal{T}$ und $\mathcal{I} \models \mathcal{A}_j$ für $1 \leq j \leq n$.

Anforderung A5.1.1 hat zur Folge, dass dasselbe Individuum in mehr als einer Teil-ABox vorkommen kann. Ein solches Individuum heißt *verteilt beschriebenes Individuum*:

Definition 5.1.2 (Verteilt beschriebene Individuen). *Sei $\mathsf{Ind}_\cap(\mathcal{K}_k, \mathcal{K}_l) \subseteq N_I$ die Menge der Individuennamen, die jeweils in mindestens einer Zusicherung in $\mathcal{K}_k$ und $\mathcal{K}_l$, $k \neq l$, vorkommen, d. h.*

$$\mathsf{Ind}_\cap(\mathcal{K}_k, \mathcal{K}_l) = \mathsf{Ind}(\mathcal{K}_k) \cap \mathsf{Ind}(\mathcal{K}_l).$$

Sei $\mathsf{Ind}_\cap(\mathfrak{K}) \subseteq N_I$ die Menge der Individuennamen, die bei der verteilten Wissensbasis

$\mathfrak{K} = \bigcup_j \mathcal{K}_j$ *in mehr als einer Teil-Wissensbasis* $\mathcal{K}_j$ *vorkommen, d. h.*

$$\mathit{Ind}_{\cap}(\mathfrak{K}) = \bigcup_{1 \leq k \neq l \leq n} \mathit{Ind}_{\cap}(\mathcal{K}_k, \mathcal{K}_l).$$

Anforderungen A5.1.3 und A5.1.4 erfordern die Definition der Semantik einer konjunktiven Anfrage von beschreibungslogischen Anfrageatomen bezüglich einer verteilten Wissensbasis:

Definition 5.1.3 (Logische Folge einer konjunktiven Anfrage Q für eine verteilte Wissensbasis). *Sei* $\mathfrak{K}$ *eine verteilte Wissensbasis und* Q *eine konjunktive Anfrage. Dann folgt* Q *logisch aus* $\mathfrak{K}$ *(geschrieben* $\mathfrak{K} \models Q$*), falls* $\mathcal{I} \models \mathfrak{K}$ *impliziert* $\mathcal{I} \models Q$*.*

Definition 5.1.4 (Lösungen einer konjunktiven Anfrage Q für eine verteilte Wissensbasis). *Sei* $\mathfrak{K}$ *eine verteilte Wissensbasis und* Q *eine konjunktive Anfrage mit* n *Variablen* $v_1, \ldots, v_n$*, d. h.* $|\mathit{Var}(Q)| = n$*. Seien die Variablen* v_i *für* $1 \leq i \leq m \leq n$ *benannte Variablen und sonst unbenannte Variablen.*

Die Lösungen $M_{Q,\mathfrak{K}}$ *für* Q *bezüglich* $\mathfrak{K}$ *sind alle Tupel* $m_{Q,\mathfrak{K}} = (a_1, \ldots, a_m) \in N_I^m$*, für die es für alle Modelle* $\mathcal{I} \models \mathfrak{K}$ *eine Zuordnung* $\mu(v_i) = a_i^{\mathcal{I}}, 1 \leq i \leq m$*, gibt, so dass* $\mathcal{I} \models_{\mu} Q$*.*

Vor dem Hintergrund der Anforderungen A5.2.1 und A5.2.2 wird schließlich das Konzept eines verteilten Reasoning-Systems eingeführt, das beide Anforderungen erfüllt:

Definition 5.1.5 (Verteiltes Reasoning-System $\mathfrak{R}$). *Sei* $\mathfrak{R}$ *ein verteiltes Reasoning-System für eine verteilte Wissensbasis* $\mathfrak{K}$*.* $\mathfrak{R}$ *besteht aus* n *Reasoning-Knoten* R_j*,* $1 \leq j \leq n$*, denen jeweils die Teil-Wissensbasis* $\mathcal{K}_j$ *zugeordnet ist.*

Jeder Reasoning-Knoten R_j *bietet für* $\mathcal{K}_j$ *die grundlegenden Reasoning-Dienste an (siehe Abschnitt 3.3.3) sowie Instanz-Abfrage, Konzept-Realisierung, Rollen-Abfrage und Rollen-Realisierung.*

In einem verteilten Reasoning-System werden Kontextinformationen verteilt gehalten (A5.2.1), weil jedem Reasoning-Knoten eine Teil-Wissensbasis zugeordnet ist. Dabei trifft ein verteiltes Reasoning-System keine Annahmen zu den verwendeten Reasoning-Verfahren oder -Implementierungen (A5.2.2), weil die Reasoning-Knoten als *Black Box*-Komponenten angesehen werden. Es wird lediglich angenommen, dass sie die wesentlichen beschreibungslogischen Reasoning-Probleme bezüglich ihrer lokalen Teil-Wissensbasis lösen können.

In diesem Kapitel sollen Verfahren entwickelt werden, mit denen eine konjunktive Anfrage Q bezüglich einer verteilten Wissensbasis $\mathfrak{K}$ beantwortet werden kann. Zur Erfüllung der nicht-funktionalen Anforderungen soll dazu auf einem solchen verteilten Reasoning-System aufgebaut werden. Ein Verfahren zur Beantwortung von Q kann dazu nur die Reasoningfähigkeiten der Reasoning-Knoten R_j auf ihren lokalen Teil-Wissensbasen $\mathcal{K}_j$ nutzen. Diese müssen so kombiniert werden, dass die Lösungen für Q bezüglich der verteilten Wissensbasis $\mathfrak{K}$ ermittelt werden.

Es soll gezeigt werden, dass auf Basis eines verteilten Reasoning-Systems auch für unterschiedliche Arten der Verteilung von $\mathfrak{K}$ jeweils ein *vollständiges* und *korrektes* Verfahren zur Beantwortung von Q entwickelt werden kann: Seien $M_{Q,\mathfrak{K}}$ die Lösungen von Q bezüglich $\mathfrak{K}$ und $M'_{Q,\mathfrak{K}}$ die Lösungen, die ein Verfahren liefert. Das Verfahren ist genau dann *korrekt*, wenn $m \in M'_{Q,\mathfrak{K}}$ impliziert $m \in M_{Q,\mathfrak{K}}$. Es ist genau dann *vollständig*, wenn $m \in M_{Q,\mathfrak{K}}$ impliziert $m \in M'_{Q,\mathfrak{K}}$.

5.2. Verwandte Arbeiten

Zunächst wird untersucht, inwieweit bestehende Ansätze in der Lage sind, die identifizierten Anforderungen zu erfüllen. Dazu werden als Erstes Systeme betrachtet, die nur für zentrales Reasoning geeignet sind, auf denen jedoch möglicherweise aufgebaut werden kann. Dann werden verwandte Arbeiten zu Reasoning mit verteilter Wissensbasis analysiert. Diese Klassifikation ist in Tabelle 5.1 dargestellt.

5.2.1. Zentrales Reasoning

Systeme für zentrales Reasoning setzen voraus, dass die Wissensbasis, bestehend aus einer TBox und einer ABox, vollständig lokal verfügbar ist. In Tabelle 5.1 ordnen sich diese Ansätze also oben links ein.

Verfahren für beschreibungslogisches Reasoning werden seit langem erforscht. Die Ansätze reichen von strukturellen Verfahren über Tableaux-, Hypertableaux- [MSH07] und Resolutions-basierte Varianten bis zur Reduktion auf disjunktives Datalog [HMS04]. Für praktische Implementierungen haben sich bisher vor allem Tableaux-Algorithmen bewährt [BCM+03].

5.2.1.1. Vollwertige Beschreibungslogik-Reasoner

Pellet [SPG+07], RacerPro [WM05] und Fact++ [TH06] sind hochoptimierte Implementierungen von Tableaux-Algorithmen. Jedes dieser Systeme unterstützt eine ausdrucksstarke Beschreibungslogik, sie unterscheiden sich jedoch in bestimmten Konstrukten. Beispielsweise war Pellet einige Zeit der einzige Reasoner, der einen vollständigen und korrekten Algorithmus für OWL DL implementierte. Ein anderes Unterscheidungsmerkmal ist, dass Pellet in Java implementiert und der Quellcode frei verfügbar ist, während RacerPro in LISP implementiert ist und seit einiger Zeit kommerziell vertrieben wird. Fact++ ist in C++ implementiert, und der Quellcode ist ebenfalls frei verfügbar.

Es kann nicht pauschal gesagt werden, welche Implementierung die effizienteste ist. Untersuchungen haben gezeigt, dass es individuelle Stärken und Schwächen gibt [MS06c, GTH06, Lie06]. Außerdem werden die Systeme permanent weiterentwickelt (d. h. die Effizienz erhöht bzw. die Ausdrucksstärke erweitert), so dass ein permanenter Wettstreit zwischen den Entwicklergruppen herrscht.

	eine einheitliche TBox	mehrere verknüpfte TBoxen
zentrale ABox	PELLET, RACERPRO, FACT++, KAON2	nicht sinnvoll
verteilte ABox	Pothipruk et al.	DDL, $\mathcal{E}$-connections, P-DL, KAONP2P

Tabelle 5.1.: Einordnung von Arbeiten zum Reasoning mit zentraler und verteilter Wissensbasis.

5.2.1.2. Reasoning-Ansätze für skalierbares ABox-Reasoning

Die bisher vorgestellten Reasoner unterstützen ausdrucksstarke Beschreibungslogiken, die auch die Grundlage für OWL DL sind. Mit zunehmender Größe der ABox zeigt sich jedoch, dass die Reasoningeffizienz stark abnimmt [MHW06, Lie06]. Das lässt sich mit der exponentiellen Worst-Case-Berechnungskomplexität der Reasoning-Probleme erklären (vergleiche Abschnitt 3.3.3). Einige Ansätze versuchen deshalb, durch Einschränkung der Ausdrucksstärke der Beschreibungslogik eine bessere Skalierbarkeit für das ABox-Reasoning zu erreichen [HLMS08].

Eine bekannte Arbeit dazu ist der *Instance Store*, der speziell für Instanz-Abruf auf einer sehr großen ABox optimiert ist, die in einer Datenbank gehalten wird [HLTB04, BHT05]. Zur Realisierung der nachgewiesenen enormen Effizienzgewinne ist jedoch Voraussetzung, dass die ABox keine Rollen enthält, d. h. es können keinerlei Beziehungen zwischen Individuen ausgedrückt werden. Dadurch ist der Instance Store für viele Anwendungen – so auch für diese Arbeit – nicht einsetzbar.

Auch andere Systeme sind auf Skalierbarkeit bezüglich der Größe der Wissensbasis optimiert und nehmen dafür starke Einschränkungen der Ausdrucksstärke in Kauf. Dazu gehören z. B. OWLIM [KOM05] und *Minerva* [ZML+06]. Datenbankhersteller wie *Oracle* erweitern ihre Systeme ebenfalls um Schnittstellen für OWL-basierte Daten (z. B. *Oracle 11g* RDF/OWL).

5.2.1.3. Hybrid-Ansätze

Es wird weiterhin versucht, interessante Kompromisse zwischen diesen zwei Extremen zu finden. So ist SHER ein vollwertiger Beschreibungslogik-Reasoner, der auf PELLET aufsetzt. Bei einer bestimmten Klasse von ABoxen ist er jedoch in der Lage, durch Entfernung quasi-redundanter Zusicherungen die Größe der zu verarbeitenden ABox zu reduzieren ohne die Korrektheit und Vollständigkeit des Reasonings aufzugeben (*Summary Abox* [FKM+06]). Dadurch kann für diese ABoxen das Reasoning wesentlich beschleunigt werden [DFK+07].

5.2.2. Verteiltes Reasoning

Von verteiltem Reasoning spricht man, wenn Reasoning-Probleme für eine verteilte Wissensbasis gelöst werden sollen. Eine Wissensbasis kann auf unterschiedliche Arten verteilt sein: die Teil-Wissensbasen haben unterschiedliche ABoxen, aber verwenden eine einheitliche TBox (in Tabelle 5.1 unten links); oder die Teil-Wissensbasen haben sowohl unterschiedliche ABoxen als auch mehrere unterschiedliche TBoxen, die untereinander verknüpft sind (in Tabelle 5.1 unten rechts).

5.2.2.1. Eine einheitliche TBox

Eine einheitliche TBox für alle Teil-Wissensbasen führt dazu, dass eine Anfrage in der Terminologie dieser TBox an jede beliebige Teil-Wissensbasis gestellt werden kann und keine Abbildungen in andere Terminologien nötig sind. Außerdem kann in jeder Teil-Wissensbasis für die ABox die vollständige Terminologie der TBox verwendet werden. Zur Spezifikation kann natürlich die gesamte Ausdrucksstärke der zugrunde liegenden Beschreibungslogik genutzt werden.

Obwohl es sich logisch um eine einzige TBox handelt, kann ihre Definition je nach zugrunde liegender Sprache dennoch über mehrere Ontologie-Dokumente verteilt sein. In OWL DL kann z. B. in jedem Ontologie-Dokument das Konstrukt `owl:imports` mit einem anderen Ontologie-Dokument als Argument verwendet werden. Die Semantik von A `owl:imports` B ist, dass A alle Definitionen von B übernimmt und damit eine neue TBox entsteht, die aus der Vereinigung der Definitionen von A und B besteht.

Da die ABoxen der Teil-Wissensbasen unterschiedlich sind, ergeben sich jedoch erhebliche Schwierigkeiten bei der Lösung von ABox-Reasoning-Problemen wie Instanz-Prüfung und Konzept-Realisierung. So kann es sein, dass in mehreren Teil-Wissensbasen Aussagen über dasselbe Individuum gemacht werden. Zur Lösung dieser Probleme müssen jedoch alle diese Aussagen im Zusammenhang berücksichtigt werden. Je nach Komplexität der Konzeptdefinitionen in der TBox kann es sogar nötig sein, dafür die gesamte verteilte ABox einzubeziehen.

Pothipruk und Governatori haben einen Ansatz für einen Spezialfall von verteiltem Reasoning mit einheitlicher TBox entwickelt [PG05]: Sie erkennen automatisiert Teil-Wissensbasen, die mit keiner anderen Teil-Wissensbasis ein gemeinsames Individuum besitzen. Dann sind diese Teil-Wissensbasen logisch unabhängig: Wenn ABox-Reasoning auf jeder dieser Teil-Wissensbasen separat durchgeführt wird, ist die Vereinigung der Teil-Ergebnisse äquivalent zu den Ergebnissen desselben Reasonings auf der *Vereinigung* der Teil-Wissensbasen. Sind keine der Teil-Wissensbasen logisch unabhängig, muss jedoch weiterhin eine Gesamt-Wissensbasis gebildet und darauf das Reasoning durchgeführt werden. Der Ansatz bringt also nur dann Vorteile, wenn die verteilte Wissensbasis *partitioniert verteilt* ist, d. h. es gibt keine Individuen, die in mehr als einer Teil-Wissensbasis erwähnt werden. In der Praxis ist diese Voraussetzung jedoch in der Regel nicht gegeben. Auch im Rahmen dieser Arbeit kann diese Annahme nicht getroffen werden. Denn eine zentrale Anforderung ist gerade, dass Aussagen zu demselben Überwachungsobjekt in mehreren Teil-Wissensbasen vorhanden sein können.

5.2.2.2. Mehrere verknüpfte TBoxen

Verwendet jede Teil-Wissensbasis eine eigene TBox, muss zur Lösung von Reasoning-Problemen über die verteilte Wissensbasis geklärt werden, welcher Zusammenhang zwischen diesen TBoxen besteht. Dabei kann entweder von einer *fiktiven globalen TBox* ausgegangen werden, zu der alle TBoxen der Teil-Wissensbasen in Beziehung stehen, oder von mehreren unabhängigen TBoxen mit *paarweisen Abbildungen* zwischen ihnen.

Fiktive globale TBox. Wird ein fiktives globales Schema angenommen, ergibt sich ein klassisches Problem der Informationsintegration. Dabei werden prinzipiell zwei Ansätze unterschieden:

- *Global-as-View* (*GaV*): Jede Relation des globalen Schemas ist definiert als Sicht (*view*) über die verschiedenen zu integrierenden Schemata [GMPQ+97].
- *Local-as-View* (*LaV*): Jedes zu integrierende Schema ist definiert als Sicht (*view*) des globalen Schemas [LRO+96].

Entsprechende Algorithmen werden in [Hal01] untersucht. Eine Übersicht über derartige Ansätze auf Basis von Ontologien gibt [WVV+01].

Paarweise Abbildungen zwischen TBoxen. Manchmal kann nicht von einem globalen Schema ausgegangen werden, sondern es müssen paarweise Abbildungen zwischen TBoxen vorgenommen werden. Das ist zum Beispiel der Fall, wenn die Menge der potentiellen Teil-Wissensbasen sehr groß ist und eine Integration deshalb nur bei Bedarf und ad-hoc stattfindet [SHG05].

Bei der logischen Interpretation dieser TBoxen werden in der Regel lokale Modelle angenommen, die jeweils disjunkt zu den Modellen der anderen TBoxen sind. So beruht das Reasoning-System DRAGO auf dem Formalismus der *Distributed Description Logics* (DDL) [ST05, BS03]. Auch PELLET bietet Unterstützung für Reasoning mit mehreren TBoxen und verwendet dafür den $\mathcal{E}$*-connections* Formalismus, der um einiges ausdrucksstärker ist als DDL [SPG+07, KLWZ04]. (Eine Reduktion von DDL auf $\mathcal{E}$-connections ist in [KLWZ04] angegeben.) Andererseits ist selbst die Ausdrucksstärke der $\mathcal{E}$-connections begrenzt: Es ist beispielsweise nicht möglich, Subsumptionsbeziehungen zwischen Konzepten oder Rollen *unterschiedlicher* TBoxen zu definieren. Konzepte und Rollen „fremder“ TBox können zudem nicht instantiiert werden.

Package-based Description Logics (P-DL) wurden als Kompromiss zwischen den eingeschränkten Ausdrucksmöglichkeiten der DDL und $\mathcal{E}$-connections sowie der vollständigen Integration zu einer TBox mittels `owl:import` entwickelt [BCH06]. P-DL verwenden deshalb keine Abbildungen zwischen TBoxen, sondern importieren *gezielt* Konzepte aus anderen TBoxen. Dafür wird die strikte Disjunktheit der lokalen Modelle aufgegeben. (DDL und $\mathcal{E}$-connections können teilweise auf P-DL reduziert werden [BCH06].) Für verschiedene Beschreibungslogiken (z. B. $\mathcal{ALC}$, $\mathcal{SHOIQ}$) wurden Algorithmen für das Reasoning mit P-DL entwickelt [Bao07]. Es sind jedoch keine Implementierungen bekannt.

Alle diese Ansätze haben als Schwerpunkt TBox-Reasoning, also Reasoning-Probleme, die sich alleine auf die TBox beziehen. Dazu gehören Konzept-Erfüllbarkeit und

Anforderung	einheitliche TBox, zentrale ABox	einheitliche TBox, verteilte ABox	verknüpfte TBoxen, verteilte ABox
A5.1.1 Kontextinformationen in verschiedenen Teil-Wissensbasen	−	−	−
A5.1.2a Einheitliche Systemontologie	+	+	−
A5.1.2b Ausdrucksstärke der Systemontologie	+	−	+/−
A5.1.3 Korrektes und vollständiges Reasoning	+	+	+/−
A5.1.4 Beantwortung konjunktiver Anfragen	+	−	+/−
A5.2.1 Verteilte Datenhaltung	−	+	+
A5.2.2 Unabhängigkeit von Reasoner-Implementierungen	−	+	−

Tabelle 5.2.: Bewertung verwandter Arbeiten zum verteilten Reasoning.

Subsumptions-Test. Im Rahmen dieser Arbeit kann auf ABox-Reasoning jedoch nicht verzichtet werden. Das sind Reasoning-Probleme, die sich sowohl auf die TBox als auch die ABox beziehen, wozu Instanz-Abruf und Konzept-Realisierung gehören. ABox-Reasoning wird von den existierenden Systemen jedoch nur eingeschränkt angeboten. Unterstützung für konjunktive Anfragen fehlt vollständig.

Im Gegensatz dazu steht bei KAONp2p das ABox-Reasoning im Vordergrund: KAONp2p verwendet für das Reasoning mit mehreren TBoxen paarweise Abbildungen zwischen diesen, die als Korrespondenzen zwischen konjunktiven Anfragen definiert sind [HW07, HM05]. Eine beliebige Definition von Korrespondenzen führt in der Regel zu Unentscheidbarkeit des Reasonings. Es wurden jedoch zwei Klassen von Abbildungen identifiziert, die entscheidbar sind. Die entwickelten Algorithmen wurden auf Basis des KAON2-Reasoners umgesetzt. Eine Evaluierung des Systems zeigt, dass sich die Verteilung der Wissensbasis nur unwesentlich auf die Antwortzeit auswirkt [HW07].

5.2.3. Zusammenfassende Bewertung

Tabelle 5.2 fasst die Eigenschaften der verwandten Arbeiten zum zentralen und verteilten Reasoning bezüglich der Anforderungen zusammen. (Bei Anforderung A5.1.2 wird zusätzliche zwischen der Existenz einer einheitlichen Systemontologie und ihrer Ausdrucksstärke differenziert.) Dazu wird die in Tabelle 5.1 eingeführte Klassifikation übernommen.

Für die Ansätze mit einer einheitlichen TBox und zentraler ABox gilt, dass sie die höchste Ausdrucksstärke (A5.1.2b) und komplexesten Reasoning-Möglichkeiten (A5.1.4) unterstützen. Sie sind jedoch darauf angewiesen, dass die gesamte Wissensbasis zentral vorliegt (A5.2.1). Um diese Verfahren zum Reasoning über verteilte Kontextinformationen einzusetzen, muss die verteilte Wissensbasis also zunächst aufwändig zentralisiert werden.

Bestehende Ansätze mit einer einheitlichen TBox und verteilter ABox gehen zwar davon aus, dass die ABox verteilt ist (A5.2.1). Damit die Reasoning-Verfahren vollständig und korrekt sind, wird jedoch angenommen, dass jedes Individuum nur in einer Teil-Wissensbasis vorkommt. Dadurch ist es nicht möglich, dass Kontextinformationen zu demselben Überwachungsobjekt über mehrere Teil-Wissensbasen verteilt sind (A5.1.1). Die Ausdrucksstärke der Systemontologie ist bei Pothipruk et al. zudem auf $\mathcal{ALC}$ beschränkt (A5.1.2). Auch konjunktive Anfragen werden nicht unterstützt (A5.1.4). Vorteilhaft ist jedoch, dass dieser Ansatz keine Annahmen über das zugrunde liegende Reasoning-Verfahren trifft (A5.2.2).

Bei Ansätzen mit mehreren verknüpften TBoxen und einer verteilten ABox ist ebenfalls vorgesehen, dass die ABox verteilt ist (A5.2.1). Im Vergleich zu Ansätzen mit einer *einheitlichen TBox* und verteilter ABox können über die Verknüpfung der TBoxen zwar Zusammenhänge zwischen den Teil-Wissensbasen ausgedrückt werden. Die Ausdrucksmöglichkeiten sind jedoch sehr beschränkt (A5.1.2b). Sie reichen nicht aus, um Kontextinformationen zu demselben Überwachungsobjekt über mehrere Teil-Wissensbasen zu verteilen (A5.1.1). Diese beschränkte Ausdrucksstärke führt auch zu beschränkten Reasoning-Möglichkeit (A5.1.3). Die Verfahren werden häufig nicht für konjunktive Anfragen verallgemeinert (A5.1.4). Alle diese Verfahren beruhen außerdem auf der *Modifikation* eines bestehenden Reasoning-Verfahrens – häufig eines Tableaux-Algorithmus. Darum können sie nicht für andere Reasoning-Verfahren und -Implementierungen eingesetzt werden (A5.2.2).

Insgesamt muss also festgestellt werden, dass keiner der vorgestellten Ansätze alle in Abschnitt 5.1 identifizierten Anforderungen für Reasoning über verteilte Kontextinformationen erfüllt. Ansätze mit einheitlicher TBox und verteilter ABox sind jedoch ein vielversprechender Ausgangspunkt, weil sie die beiden zentralen Anforderung der verteilten Datenhaltung (A5.2.1) und Unabhängigkeit von Reasoning-Verfahren (A5.2.2) erfüllen. Eine einheitliche TBox ist wünschenswert, weil auf diese Weise eine hohe Ausdrucksstärke erreicht werden kann. In Kapitel 4 wurde gezeigt, wie eine entsprechende Systemontologie für die Zustandsüberwachung in Infrastrukturnetzen auch dann erstellt werden kann, wenn mehrere Domänenexperten daran beteiligt sind. Auch aus organisatorischer Sicht ist eine einheitliche Systemontologie umsetzbar, weil zentrale technische Standards in den verschiedenen Infrastrukturdomänen von internationalen Verbänden vorgegeben werden. Beim Schienenverkehr übernimmt diese Aufgabe beispielsweise der *Internationale Eisenbahnverband* (UIC).

Für den in dieser Arbeit entwickelten Ansatz wird somit eine einheitliche TBox und eine verteilte ABox angenommen. Damit ordnet er sich in Tabelle 5.1 links unten ein. Im Folgenden müssen neue Konzepte entwickelt werden, um damit Kontextinformationen berücksichtigen zu können, die sich auf dasselbe Überwachungsobjekt beziehen, jedoch in unterschiedlichen Teil-Wissensbasen liegen (A5.1.1). Außerdem muss die Ausdrucksstärke der Systemontologie erhöht werden (A5.1.2), für die korrektes und vollständiges

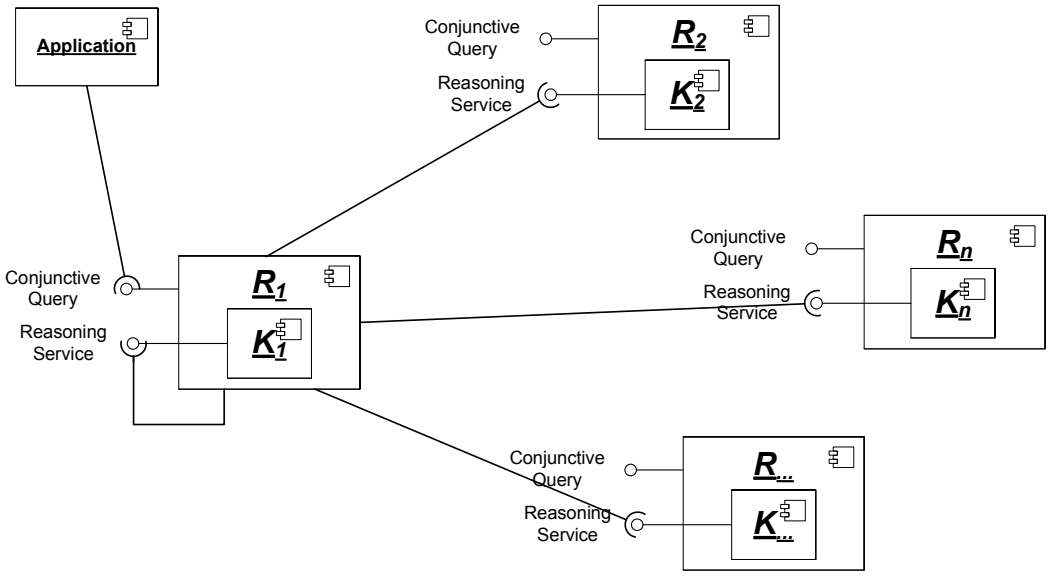

Abbildung 5.1.: Verteiltes Reasoning-System $\mathfrak{R}$ mit Reasoning-Knoten R_j und Teil-Wissensbasen $\mathcal{K}_j$.

Reasoning unterstützt wird. Schließlich müssen auch konjunktive Anfragen beantwortet werden können (A5.1.4).

5.3. Anfragebeantwortung in einem verteilten Reasoning-System

Dieser Abschnitt entwickelt Verfahren zur Anfragebeantwortung in einem verteilten Reasoning-System $\mathfrak{R}$ mit einer verteilten Wissensbasis $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ (vergleiche Abschnitt 5.1.2). Laut Definition ist jedem Reasoning-Knoten R_j die Teil-Wissensbasis $\mathcal{K}_j$ zugeordnet. Außerdem bietet jeder Reasoning-Knoten R_j eine Schnittstelle an, über die die wesentlichen Reasoning-Probleme auf $\mathcal{K}_j$ gelöst werden können. Gesucht ist ein korrektes und vollständiges Verfahren, um für eine konjunktive Anfrage Q die Lösungen $M_{Q,\mathfrak{K}}$ bezüglich $\mathfrak{K}$ zu ermitteln. Dabei ist eine Lösung $m \in M_{Q,\mathfrak{K}}$ definiert als ein Tupel, das alle Anfrageatome $q \in Q$ erfüllt (siehe Definition 5.1.4).

Eine konjunktive Anfrage Q kann an jeden beliebigen Reasoning-Knoten R_j gestellt werden. Ein einzelner Reasoning-Knoten kann sie in der Regel jedoch nicht beantworten, da ihm nur eine Teil-Wissensbasis zur Verfügung steht. Deshalb müssen zunächst Lösungen für jede Unteranfrage $\{q\}$ ermittelt werden. Auch dazu ist ein einzelner Reasoning-Knoten im Allgemeinen nicht in der Lage, weil sich relevante Informationen in mehreren Teil-Wissensbasen befinden können. Deshalb besteht die Herausforderung für den Reasoning-Knoten R_j darin, die Schnittstelle anderer Reasoning-Knoten $R_k,\ 1 \leq\ k \leq\ n$, so zu nutzen, dass vollständige und korrekte Lösungen für Q ermittelt werden (siehe die Illustration in Anlehnung an UML-Komponentendiagramme in Abbildung 5.1). Dabei sind prinzipiell zwei Schritte zu unterscheiden:

1. *Prüfung der logischen Konsistenz*: Um sinnvolle Lösungen für Q ermitteln zu

können, muss sichergestellt sein, dass $\mathfrak{K}$ logisch konsistent ist, denn aus einer inkonsistenten Wissensbasis folgt logisch jede beliebige Aussage (vergleiche Abschnitt 3.2.1).

2. *Beantwortung beschreibungslogischer Anfrageatome*: Die Beantwortung von Q lässt sich auf die Beantwortung der Anfrageatome $q \in Q$ zurückführen. Jedes q ist ein beschreibungslogisches *Anfrageatom* und entweder von der Form $C(t)$ (*Konzept-Anfrageatom*) oder von der Form $r(t, t')$ (*Rollen-Anfrageatom*) mit $t, t' \in N_V \cup N_I$. Zu ihrer Beantwortung muss beschreibungslogisches Reasoning über $\mathfrak{K}$ durchgeführt werden.

Ziel ist es, die *Beantwortung von Q* auf die Beantwortung der einzelnen Anfrageatome $q \in Q$ zurückzuführen. Gelingt dies, können konjunktive Anfragen effizient bearbeitet werden, weil die Anfrageatome q teilweise unabhängig voneinander ausgewertet werden können. Außerdem kann dann ein Verfahren angegeben werden, das unabhängig von der Verteiltheit von $\mathfrak{K}$ ist (siehe Abschnitt 5.3.4).

Um die Beantwortung von Q auf die Beantwortung der $q \in Q$ zurückführen zu können, wird angenommen, dass alle Variablen in Q benannte Variablen sind, d. h. alle Variablen in Q können nur an benannte Individuen $a \in N_I$ gebunden werden (vergleiche Abschnitt 3.3.3.3). Dies entspricht dem Verhalten aktueller Reasoner-Implementierungen und wird bei RacerPro beispielsweise als *active domain assumption* bezeichnet [WM05]. Damit wird zwar ein Teil der Ausdrucksstärke konjunktiver Anfragen aufgegeben, im Gegenzug jedoch eine höhere Effizienz bei der Anfragebearbeitung erreicht. Bezogen auf die in Kapitel 4 vorgestellten Entwurfsmuster sind weiterhin alle Anfragen ausdrückbar, da dort alle Individuen benannt sind.

Sowohl für die *Prüfung der logischen Konsistenz* als auch die *Beantwortung der beschreibungslogischen Anfrageatome* muss jedoch die Art der Verteiltheit von $\mathfrak{K}$ berücksichtigt werden, damit die Lösungen korrekt und vollständig sind. Deshalb definiert diese Arbeit folgenden Arten der Verteiltheit von $\mathfrak{K}$ (siehe Illustration in Abbildung 5.2):

- *Beliebig verteilte Wissensbasis* $\mathfrak{K}$. ABox-Zusicherungen können beliebig über die Teil-Wissensbasen $\mathcal{K}_j$ verteilt sein. Insbesondere kann $\mathsf{Ind}_\cap(\mathfrak{K}) \neq \emptyset$ gelten. Dieser Fall wird in Abschnitt 5.3.1 behandelt.

- *Partitioniert verteilte Wissensbasis* $\mathfrak{K}$. ABox-Zusicherungen sind so über die Teil-Wissensbasen $\mathcal{K}_j$ verteilt, dass jedes Individuum nur in höchstens einer Teil-Wissensbasis erwähnt wird, d. h. $\mathsf{Ind}_\cap(\mathfrak{K}) = \emptyset$. Dieser Fall wird in Abschnitt 5.3.2 behandelt.

- *Eingeschränkt verteilte Wissensbasis* $\mathfrak{K}$. ABox-Zusicherungen können so über die Teil-Wissensbasen $\mathcal{K}_j$ verteilt sein, dass dasselbe Individuum in mehr als einer Teil-Wissensbasis vorkommt, d. h. möglicherweise $\mathsf{Ind}_\cap(\mathfrak{K}) \neq \emptyset$. Es gelten jedoch gewisse Einschränkungen für Zusicherungen zu den verteilt beschriebenen Individuen $\mathsf{Ind}_\cap(\mathfrak{K})$, welche in Abschnitt 5.3.3 formuliert und untersucht werden. Wie gezeigt wird, sind diese Einschränkungen vereinbar mit den Entwurfsmustern aus Kapitel 4, weshalb eingeschränkt verteilte Wissensbasen zur Integration von Kontextinformationen genutzt werden können.

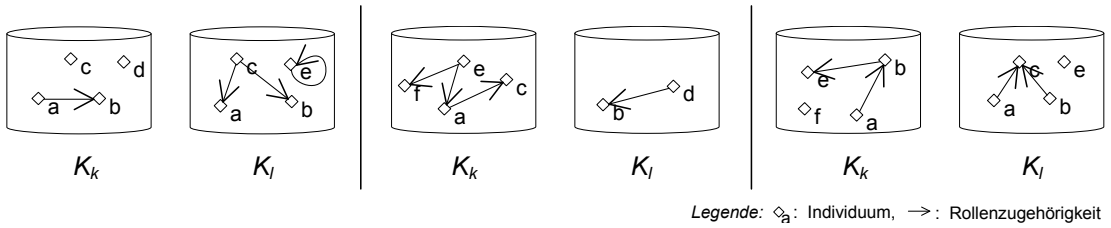

Abbildung 5.2.: Vereinfachte Illustration unterschiedlich verteilter Wissensbasen: *beliebig verteilt* (links), *partitioniert verteilt* (Mitte), *eingeschränkt verteilt* (rechts).

Dabei stellen partitioniert und eingeschränkt verteilte Wissensbasen Spezialfälle von beliebig verteilten Wissensbasen dar. Gibt es Verfahren zur Prüfung der logischen Konsistenz und Beantwortung von Anfrageatomen bei beliebig verteilten Wissensbasen, können diese also auch bei partitioniert und eingeschränkt verteilten Wissensbasen eingesetzt werden. Darüber hinaus können für diese Spezialfälle jedoch auch wesentlich effizientere Verfahren entwickelt werden.

Ziel ist es, in den Abschnitten 5.3.1, 5.3.2 und 5.3.3 entsprechende Verfahren für alle drei Arten der Verteiltheit von $\mathfrak{K}$ entwickelt. Sie sollen in einem verteilten Reasoning-System umgesetzt werden. Aufbauend auf diesen Verfahren zur Beantwortung von Anfrageatomen soll in Abschnitt 5.3.4 ein Verfahren zu Beantwortung von *Konjunktionen* von Anfrageatomen entwickelt werden, das *unabhängig* von der Verteiltheit der Wissensbasis ist. Dieses soll ebenfalls in einem verteilten Reasoning-System umgesetzt werden.

5.3.1. Beliebig verteilte Wissensbasis

Bei einer beliebig verteilten Wissensbasis sind ABox-Zusicherungen beliebig über die Teil-Wissensbasen $\mathcal{K}_j, 1 \leq j \leq n$, verteilt, d. h. es kann gelten $\mathsf{Ind}_\cap(\mathfrak{K}) \neq \emptyset$.

In diesem Fall kann im Allgemeinen weder bei der Konsistenz-Prüfung noch bei der Beantwortung der Anfrageatome vermieden werden, die verteilte Wissensbasis zu zentralisieren. Dies wird im Folgenden gezeigt.

5.3.1.1. Prüfung der logischen Konsistenz

Um die logische Konsistenz von $\mathfrak{K}$ zu prüfen, muss entschieden werden, ob eine Interpretation $\mathcal{I}$ existiert, die $\mathfrak{K}$ modelliert, d. h. ob $\mathcal{I} \models \mathfrak{K}$. Wenn $\mathfrak{K}$ vollständig bei einem Reasoning-Knoten R_j vorhanden ist, kann dazu der Reasoning-Dienst genutzt werden, den R_j für $\mathcal{K}_j$ zur Verfügung stellt.

Das ist im Allgemeinen – wenn $\mathfrak{K}$ beliebig verteilt ist – jedoch nicht der Fall: Es lässt sich leicht eine Verteiltheit von $\mathfrak{K}$ konstruieren, bei der die Konsistenz von $\mathfrak{K}$ nur entschieden werden kann, wenn $\mathfrak{K}$ vollständig in einem Reasoning-Knoten zentralisiert wird.

Hier ein einfaches Beispiel für eine verteilte Wissensbasis $\mathfrak{K}$ mit zwei Teil-Wissensbasen $\mathcal{K}_1$ und $\mathcal{K}_2$:

Seien $\mathcal{K}_1 = (\emptyset, \{r(a, b_1), r(a, b_2), b_1 \neq b_2, (\leqslant_2 r)(a)\})$ *und* $\mathcal{K}_2 = (\emptyset, \{r(a, b_3), b_3 \neq b_2, b_3 \neq b_1\})$. *Sowohl* $\mathcal{K}_1$ *als auch* $\mathcal{K}_2$ *sind für sich genommen konsistent.* $\mathfrak{K} = \mathcal{K}_1 \cup \mathcal{K}_2$ *ist jedoch inkonsistent, da das Individuum* a *in* $\mathfrak{K}$ *insgesamt drei unterschiedliche* r*-Nachbarn zugesichert hat, obwohl* $\mathcal{K}_1$ *definiert, dass* a *höchstens zwei hat.*

Ein korrektes Verfahren für die Konsistenz-Prüfung einer beliebig verteilten Wissensbasis erfordert also die Zentralisierung der gesamten verteilten Wissensbasis. Im verteilten Reasoning-System wird das umgesetzt, indem ein beliebiger Reasoning-Knoten R_j mittels der einheitlichen Schnittstelle von allen anderen Reasoning-Knoten $R_k, k \neq j$, die assoziierte Teil-Wissensbasis $\mathcal{K}_k$ abfragt. R_j aggregiert die Teil-Wissensbasen in $\mathcal{K}_j$ und nutzt dann seinen Reasoning-Dienst, um die Konsistenz des modifizierten $\mathcal{K}_j$ zu entscheiden.

5.3.1.2. Beantwortung von Anfrageatomen

Sei die beliebig verteilte Wissensbasis $\mathfrak{K}$ also konsistent. Nun wird ein Verfahren gesucht, um Anfrageatome in Form von Konzept- und Rollen-Anfrageatomen bezüglich $\mathfrak{K}$ zu beantworten.

Beantwortung von Rollen-Anfrageatomen. Sei r zunächst eine nicht-transitive Rolle. Zur Beantwortung eines *Rollen-Anfrageatoms* $r(a, b)$ bezüglich $\mathfrak{K}$ muss nur für Teil-Wissensbasen $\mathcal{K}_k$ mit $a, b \in \mathsf{Ind}(\mathcal{K}_k)$ (und einer Rollenzusicherung zwischen diesen) geprüft werden, ob $\mathcal{K}_k \models r(a, b)$. Das ist ausreichend, da die TBox $\mathcal{T}$ in jeder Teil-Wissensbasis $\mathcal{K}_j$ verfügbar ist und deshalb auch Rollenhierarchie und inverse Rollen berücksichtigt werden. Bei transitiven Rollen kann es sein, dass mehrere Teil-Wissensbasen überprüft werden müssen. Dazu werden Mechanismen benötigt, die in Abschnitt 5.3.4 beschrieben werden.

Beantwortung von Konzept-Anfrageatomen. Für die Beantwortung von *Konzept-Anfrageatomen* $C(a)$ bezüglich $\mathfrak{K}$ ist es in der Regel nicht ausreichend, eine einzelne Teil-Wissensbasis zu prüfen. Denn die Konzeptzugehörigkeit eines Individuums a kann im Gegensatz zur Rollenzugehörigkeit auch von Zusicherungen in einer Teil-Wissensbasis $\mathcal{K}_k$ mit $a \notin \mathsf{Ind}(\mathcal{K}_k)$ abhängen.

In Anlehnung an das vorangegangene Beispiel:

Seien $\mathcal{K}_1 = (\emptyset, \{r(a, b_1), r(a, b_2), b_1 \neq b_2\})$, $\mathcal{K}_2 = (\emptyset, \{r(a, b_3), b_3 \neq b_2, b_3 \neq b_1\})$, $\mathcal{K}_3 = (\emptyset, C(b_2))$ *und* $\mathfrak{K} = \mathcal{K}_1 \cup \mathcal{K}_2 \cup \mathcal{K}_3$. *Das Anfrageatom* $(\geqslant_3 r \sqcap \exists\, r.C)(x)$ *ist eine Instanz-Abfrage für alle Individuen, die mindestens drei unterschiedliche* r*-Nachbarn sowie eine Instanz von* C *als* r*-Nachbarn haben.* $\mathcal{K}_1$ *und* $\mathcal{K}_2$ *liefern für sich genommen und auch in Vereinigung keine Antwort. Erst wenn* $\mathcal{K}_3$ *dazugenommen wird, in der* a *gar nicht vorkommt, ergibt sich* a *als Lösung, was die korrekte Antwort bezüglich* $\mathfrak{K}$ *ist.*

Um die korrekte Beantwortung von Konzept-Anfrageatomen bezüglich einer beliebig verteilten Wissensbasis zu garantieren, muss also ebenfalls die gesamte Wissensbasis $\mathfrak{K}$ zentralisiert werden. Im verteilten Reasoning-System wird das umgesetzt, indem der

Reasoning-Knoten R_j, der die Ausgangsanfrage erhalten hat, mittels der einheitlichen Schnittstelle von allen anderen Reasoning-Knoten $R_k, k \neq j$, die assoziierte Teil-Wissensbasis $\mathcal{K}_k$ abfragt. R_j aggregiert die Teil-Wissensbasen in $\mathcal{K}_j$ und führt dann Rollen-Prüfung, -Abfrage oder -Realisierung bzw. Instanz-Prüfung, -Abfrage oder Konzept-Realisierung auf $\mathcal{K}_j$ durch, um das Anfrageatom zu beantworten.

Muss von einer beliebig verteilten Wissensbasis ausgegangen werden, gibt es also viele Situationen, in denen die gesamte Wissensbasis zentralisiert werden muss, um korrekte und vollständige Ergebnisse zu erhalten. Dies kann mit dem vorgeschlagenen verteilten Reasoning-System zwar umgesetzt werden, in der Regel werden jedoch effizientere Verfahren benötigt. Deshalb werden im Folgenden attraktive Spezialfälle der Verteiltheit von $\mathfrak{K}$ untersucht, um effizientere Verfahren zur Beantwortung von Anfrageatomen zu realisieren.

5.3.2. Partitioniert verteilte Wissensbasis

Bei einer partitioniert verteilten Wissensbasis sind ABox-Zusicherungen so über die Teil-Wissensbasen $\mathcal{K}_j, 1 \leq j \leq n$, verteilt, dass jedes Individuum nur in höchstens einer Teil-Wissensbasis vorkommt, d. h. $\mathsf{Ind}_\cap(\mathfrak{K}) = \emptyset$.

In diesem Fall können die Konsistenz-Prüfung und die Beantwortung von Anfrageatomen verteilt umgesetzt werden:

- Teil-Wissensbasen, die – z. B. aus Gründen der Kontrolle über die Daten – verteilt bleiben sollen, müssen nicht zentralisiert werden.
- Die Effizienz der Verfahren wird erhöht: Für die Konsistenz-Prüfung können Teil-Wissensbasen unabhängig voneinander betrachtet und die Bearbeitung damit parallelisiert werden, wie Abschnitt 5.3.2.1 zeigt. Bei der Beantwortung von Anfrageatomen reicht es zudem aus, jeweils nur eine der Teil-Wissensbasen zu berücksichtigen, wie in Abschnitt 5.3.2.2 gezeigt wird.

Für die einfache Beschreibungslogik $\mathcal{ALC}$ wurde dieser Fall bereits in [PG05] untersucht. Für den Zweck dieser Arbeit muss die Betrachtung jedoch auf die Beschreibungslogik $\mathcal{SHIN}$ erweitert werden.

5.3.2.1. Prüfung der logischen Konsistenz

Bei einer partitioniert verteilten Wissensbasis wird für die Konsistenz-Prüfung ausgenutzt, dass $\mathsf{Ind}_\cap(\mathfrak{K}) = \emptyset$. Denn es gilt folgender Zusammenhang:

Theorem 5.3.1 (Konsistenz einer partitioniert verteilten Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine partitioniert verteilte Wissensbasis. Dann gilt:*

$$\mathcal{K}_j \textit{ ist konsistent für alle } j \iff \mathfrak{K} \textit{ ist konsistent.}$$

Um die Konsistenz von $\mathfrak{K}$ zu prüfen, reicht es also aus, für jede Teil-Wissensbasis $\mathcal{K}_j$, $1 \leq j \leq n$, unabhängig die Konsistenz zu prüfen.

Beweis. Das Theorem folgt aus folgenden Überlegungen: Die Wissensbasis $\mathfrak{K}$ ist genau dann konsistent, wenn der $\mathcal{SHIN}$-Tableaux-Algorithmus T (siehe Definition 3.4.3 auf Seite 46) einen widerspruchsfreien und vollständigen Komplettierungswald $\mathcal{F}$ konstruiert (vergleiche Theorem 3.4.1). Da für $\mathfrak{K}$ gilt, dass $\mathsf{Ind}_{\cap}(\mathfrak{K}) = \emptyset$, kann es keine Rollen-Zusicherung für ein Individuum geben, das in mehr als einer Teil-Wissensbasis $\mathcal{K}_j$ vorkommt. Gemäß Definition von T enthält der initiale Komplettierungswald für $\mathfrak{K}$ deshalb auch keine Kante zwischen Individuen, die in unterschiedlichen Teil-Wissensbasen vorkommen. Da eine derartige Kante auch nicht durch die Expansionsregeln von T erzeugt werden kann, enthält ein Komplettierungswald für $\mathfrak{K}$ nie eine Kante zwischen Individuen, die in unterschiedlichen Teil-Wissensbasen vorkommen; der Komplettierungswald $\mathcal{F}$ für $\mathfrak{K}$ ist also ebenfalls partitioniert.

Die partitioniert verteilte Wissensbasis $\mathfrak{K}$ ist gemäß Theorem 3.4.1 genau dann inkonsistent, wenn T keinen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}$ konstruiert. Ein Widerspruch in $\mathcal{F}$ ergibt sich genau dann, wenn für einen Knoten x in $\mathcal{F}$ gilt, dass $\mathcal{L}(x)$ einen Widerspruch enthält. Aus den Expansionsregeln von T (siehe Tabelle 3.4 auf Seite 47) ist ersichtlich, dass ein anderer Knoten y in $\mathcal{F}$ mit $y \neq x$ nur dann einen Einfluss auf $\mathcal{L}(x)$ haben kann, wenn zwischen x und y eine Kante in $\mathcal{F}$ besteht. Da $\mathcal{F}$ entsprechend der Aufteilung in Teil-Wissensbasen partitioniert ist, gilt insbesondere, dass ein Widerspruch in einer Teil-Wissensbasis $\mathcal{K}_k$ unabhängig von allen anderen Teil-Wissensbasen $\mathcal{K}_j, 1 \leq j \neq k \leq n$, ist. Ein Widerspruch im Komplettierungswald $\mathcal{F}$ für $\mathfrak{K}$ tritt also genau dann auf, wenn für (mindestens) ein k in dem Komplettierungswald $\mathcal{F}_k$ der Teil-Wissensbasis $\mathcal{K}_k$ ein Widerspruch auftritt. □

Im vorgeschlagenen verteilten Reasoning-System wird das Verfahren zur Konsistenz-Prüfung von $\mathfrak{K}$ wie folgt umgesetzt: Jeder Reasoning-Knoten $R_j, 1 \leq j \leq n$, nutzt seinen lokalen Reasoning-Dienst, um die Konsistenz von $\mathcal{K}_j$ zu prüfen. Genau dann, wenn alle Teil-Wissensbasen $\mathcal{K}_j$ konsistent sind, ist auch $\mathfrak{K}$ konsistent.

5.3.2.2. Beantwortung von Anfrageatomen

Sei die partitioniert verteilte Wissensbasis $\mathfrak{K}$ also konsistent. Nun sollen Rollen- und Konzept-Anfrageatome bezüglich $\mathfrak{K}$ beantwortet werden.

Beantwortung von Rollen-Anfrageatomen. Sei r eine transitive oder nicht-transitive Rolle. Zur Beantwortung eines *Rollen-Anfrageatoms* $r(a, b)$ bezüglich $\mathfrak{K}$ müssen nur Teil-Wissensbasen $\mathcal{K}_k$ mit $a, b \in \mathsf{Ind}(\mathcal{K}_k)$ berücksichtigt und $\mathcal{K}_k \models r(a, b)$ geprüft werden. Wieder ist die TBox $\mathcal{T}$ in jeder Teil-Wissensbasis $\mathcal{K}_j$ verfügbar und Rollenhierarchie und inverse Rollen werden deshalb berücksichtigt.

Beantwortung von Konzept-Anfrageatomen. Die Beantwortung von *Konzept-Anfrageatomen* $C(a)$ bezüglich einer partitioniert verteilten Wissensbasis $\mathfrak{K}$ kann ebenfalls effizient umgesetzt werden:

Theorem 5.3.2 (Instanz-Prüfung bei partitioniert verteilter Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine partitioniert verteilte und konsistente Wissensbasis, C eine einge-*

schränkte Konzeptbeschreibung und $a \in \mathsf{Ind}(\mathfrak{K})$ *ein Individuum. Dann gilt:*

$$\mathfrak{K} \models C(a) \iff \mathcal{K}_j \models C(a) \text{ mit } a \in \mathsf{Ind}(\mathcal{K}_j) \text{ für ein } 1 \leq j \leq n.$$

Beweis. Die Aussage $\mathfrak{K} \models C(a)$ ist äquivalent zu der Aussage, dass $\mathfrak{K} \cup \{\neg C(a)\}$ inkonsistent ist. Da $\mathfrak{K}$ partitioniert verteilt ist und deshalb $\mathsf{Ind}_\cap(\mathfrak{K}) = \emptyset$ gilt, kann Theorem 5.3.1 angewandt werden. Demnach reicht es zur Entscheidung der Konsistenz von $\mathfrak{K} \cup \{\neg C(a)\}$ aus, die Konsistenz der Teil-Wissensbasen $\mathcal{K}_j \cup \{\neg C(a)\}$ mit $1 \leq j \leq n$ zu prüfen. $\mathcal{K}_j \cup \{\neg C(a)\}$ kann nur inkonsistent sein, falls $a \in \mathsf{Ind}(\mathcal{K}_j)$. Da $\mathsf{Ind}_\cap(\mathfrak{K}) = \emptyset$, reicht es also aus, für dasjenige $\mathcal{K}_k$ mit $a \in \mathsf{Ind}(\mathcal{K}_k)$ zu prüfen, ob $\mathcal{K}_k \cup \{\neg C(a)\}$ inkonsistent ist, d. h. ob $\mathcal{K}_k \models C(a)$. □

Für die Umsetzung im verteilten Reasoning-System bedeutet dies, dass der Reasoning-Knoten R_l, der die Ausgangsanfrage erhält, den Reasoning-Knoten R_k mit $a \in \mathsf{Ind}(\mathcal{K}_k)$ identifiziert. Dann wird die Schnittstelle von R_k genutzt, um die entsprechende Rollen-Prüfung, -Abfrage oder -Realisierung bzw. Instanz-Prüfung, -Abfrage oder Konzept-Realisierung auf $\mathcal{K}_k$ durchzuführen.

Bei einer partitioniert verteilten Wissensbasis $\mathfrak{K}$ können also sowohl Konsistenz-Prüfung als auch die Beantwortung von Anfrageatomen sehr effizient umgesetzt werden, da die Reasoning-Dienste auf den Teil-Wissensbasen unabhängig voneinander und – bei Anfrageatomen – auch nur auf einzelnen Teil-Wissensbasen durchgeführt werden müssen, um dennoch korrekte und vollständige Ergebnisse zu erhalten. Jedoch sind die Ausdrucksmöglichkeiten bei einer partitioniert verteilten Wissensbasis sehr eingeschränkt, da jedes Individuum nur in höchstens einer Teil-Wissensbasis vorkommen darf. Deshalb ist es hier zum Beispiel nicht möglich, Aussagen zu demselben Überwachungsobjekt in mehreren Teil-Wissensbasen zu machen, da das Überwachungsobjekt-Individuum dazu in mehr als einer Teil-Wissensbasis erwähnt werden müsste (vergleiche Anforderung A5.1.1).

Um eine Möglichkeit zu haben, dasselbe Individuum in mehreren Teil-Wissensbasen zu verwenden und Anfrageatome dennoch effizient beantworten zu können, werden im Folgenden Bedingungen für sogenannte eingeschränkt verteilte Wissensbasen entwickelt.

5.3.3. Eingeschränkt verteilte Wissensbasis

Bei einer eingeschränkt verteilten Wissensbasis sind ABox-Zusicherungen so über die Teil-Wissensbasen $\mathcal{K}_j$ verteilt, dass dasselbe Individuum in mehr als einer Teil-Wissensbasis vorkommen kann, d. h. es gilt möglicherweise $\mathsf{Ind}_\cap(\mathfrak{K}) \neq \emptyset$. Es werden jedoch gewisse Einschränkungen für Zusicherungen zu den verteilt beschriebenen Individuen $\mathsf{Ind}_\cap(\mathfrak{K})$ eingeführt (siehe Abschnitt 5.3.3.1).

Dann kann gezeigt werden, dass die Konsistenz-Prüfung und die Beantwortung von Anfrageatomen – trotz der Existenz von verteilt beschriebenen Individuen – verteilt umgesetzt werden können. Somit müssen auch hier Teil-Wissensbasen, die verteilt bleiben sollen, nicht vollständig zentralisiert werden. Gleichzeitig wird eine hohe Effizienz der Verfahren erreicht:

- Bei der Konsistenz-Prüfung können Teil-Wissensbasen wieder unabhängig voneinander betrachtet und die Bearbeitung damit parallelisiert werden, wie Abschnitt 5.3.3.2 zeigt.

- Bei der Beantwortung von Anfrageatomen muss differenziert werden, wie in Abschnitt 5.3.3.3 gezeigt wird:
  - Handelt es sich um ein Rollen-Anfrageatom zu beliebigen Individuen oder um ein Konzept-Anfrageatom zu einem *nicht verteilt beschriebenen* Individuum, so muss erneut nur höchstens eine der Teil-Wissensbasen berücksichtigt werden.
  - Handelt es sich um ein Konzept-Anfrageatom zu einem *verteilt beschriebenen* Individuum, so kann eine Zentralisierung der verteilten Wissensbasis ebenfalls vermieden werden: Auf Basis der relevanten Teil-Wissensbasen wird eine temporäre ABox generiert, die als *Wissensbasis-Kern* bezeichnet wird. Mit ihrer Hilfe kann das Anfrageatom korrekt und vollständig beantwortet werden, ohne die gesamte Wissensbasis zentralisieren zu müssen.

Im Folgenden werden die Eigenschaften einer eingeschränkt verteilten Wissensbasis definiert. Anschließend wird gezeigt, wie diese Annahmen bei der Konsistenz-Prüfung und Beantwortung von Anfrageatomen ausgenutzt werden. Da die in Kapitel 4 entwickelten Entwurfsmuster die Bedingungen an eine eingeschränkt verteilte Wissensbasis erfüllen (siehe Abschnitt 5.3.3.4), können diese Verfahren für Reasoning über verteilte Kontextinformationen in Infrastrukturnetzen eingesetzt werden.

5.3.3.1. Eigenschaften einer eingeschränkt verteilten Wissensbasis

Zur Formulierung der Bedingungen an eine eingeschränkt verteilte Wissensbasis muss zunächst die Konzepthülle $\mathsf{clos}(\mathcal{K})$ der Wissensbasis $\mathcal{K}$ definiert werden. Dabei ist E eine *Unter-Konzeptbeschreibung* von D, wenn sich D aus E entsprechend der rekursiven Definition der $\mathcal{SHIN}$-Konzeptbeschreibungen in Abschnitt 3.4.2 ergibt.

Definition 5.3.1 (Konzepthülle $\mathsf{clos}(\mathcal{K})$ einer Wissensbasis $\mathcal{K}$). *Sei $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ eine Wissensbasis mit Konzeptbeschreibungen in NNF (siehe Definition 3.4.1). Sei C eine Konzeptbeschreibung in NNF. Dann ist $\mathsf{clos}(C)$ definiert als die kleinste Menge, für die gilt:*

$$C \in \mathsf{clos}(C)$$

$$D \in \mathsf{clos}(C) \implies \mathsf{NNF}(\neg D) \in \mathsf{clos}(C)$$

$$E \text{ ist Unter-Konzeptbeschreibung von } D \in \mathsf{clos}(C) \implies E \in \mathsf{clos}(C).$$

Die Definition von $\mathsf{clos}(\mathcal{K})$ baut darauf auf: Sei $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ und sei $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$ das Ergebnis der TBox-Internalisierung von $\mathcal{K}$, d. h. $\mathcal{T}'$ enthält nur Rolleninklusionen (vergleiche Abschnitt 3.4.3). Dann gilt:

$$\mathsf{clos}(\mathcal{K}) = \bigcup_{C(a) \in \mathcal{A}'} \mathsf{clos}(C).$$

Unter Verwendung von $\mathsf{clos}(\mathfrak{K})$ mit $\mathfrak{K} = (\mathcal{T}, \bigcup_j \mathcal{A}_j)$ lassen sich nun verteilt beschreibende Rollen bezüglich $\mathfrak{K}$, eingeschränkte Konzeptbeschreibungen bezüglich $\mathfrak{K}$ und die eingeschränkt verteilte Wissensbasis $\mathfrak{K}$ definieren:

Definition 5.3.2 (Verteilt beschreibende Rollen bezüglich $\mathfrak{K}$). *Sei $\mathfrak{K}$ eine verteilte Wissensbasis, $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ein verteilt beschriebenes Individuum und r eine Rolle.*

- *r heißt* abgehende verteilt beschreibende Rolle *bezüglich $\mathfrak{K}$, falls $i_\cap$ einen r-Nachbarn in $\mathfrak{K}$ hat.*
- *r heißt* eingehende verteilt beschreibende Rolle *bezüglich $\mathfrak{K}$, falls $i_\cap$ ein r-Nachbar in $\mathfrak{K}$ ist.*

Definition 5.3.3 (Eingeschränkte Konzeptbeschreibungen bezüglich $\mathfrak{K}$). *Sei $\mathfrak{K}$ eine verteilte Wissensbasis, $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ein verteilt beschriebenes Individuum und r eine verteilt beschreibende Rolle bezüglich $\mathfrak{K}$. Dann ist eine $\mathcal{SHIN}$-Konzeptbeschreibung C in NNF eine* eingeschränkte Konzeptbeschreibung *bezüglich $\mathfrak{K}$, falls gilt:*

- *Falls r eine abgehende verteilt beschreibende Rolle ist, ist $(\leqslant_n r)$ keine Unter-Konzeptbeschreibung von C.*
- *Falls r eine eingehende verteilt beschreibende Rolle ist, ist $(\forall r.D)$ für kein D eine Unter-Konzeptbeschreibung von C.*

Definition 5.3.4 (Eingeschränkt verteilte Wissensbasis $\mathfrak{K}$). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine verteilte Wissensbasis mit $\mathcal{K}_j = (\mathcal{T}, \mathcal{A}_j), 1 \leq j \leq n$, seien $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ein verteilt beschriebenes Individuum, C eine Konzeptbeschreibung und r eine verteilt beschreibende Rolle bezüglich $\mathfrak{K}$. Dann ist $\mathfrak{K}$ eine* eingeschränkt verteilte Wissensbasis*, wenn gilt:*

$$C(i_\cap) \in \mathcal{A}_i \text{ für ein } 1 \leq i \leq n \Longrightarrow C(i_\cap) \in \mathcal{A}_j \text{ für alle } 1 \leq j \leq n \quad (5.1)$$

$$r \text{ ist abgehende verteilt beschreibende Rolle} \Longrightarrow (\leqslant_n r) \notin \mathsf{clos}(\mathfrak{K}) \quad (5.2)$$

$$r \text{ ist eingehende verteilt beschreibende Rolle} \Longrightarrow (\forall r.C') \notin \mathsf{clos}(\mathfrak{K}) \text{ für alle } C'. \quad (5.3)$$

Bedingungen (5.2) und (5.3) können mittels Definition 5.3.3 auch bezüglich TBox und ABox ausgedrückt werden:

> *In TBox-Axiomen und ABox-Zusicherungen werden nur eingeschränkte Konzeptbeschreibungen bezüglich $\mathfrak{K}$ verwendet.*

Ob eine verteilte Wissensbasis $\mathfrak{K}$ diese Bedingungen erfüllt und damit eingeschränkt verteilt ist, lässt sich also auf Basis von TBox $\mathcal{T}$ und ABoxen $\mathcal{A}_j, 1 \leq j \leq n$, automatisiert überprüfen.

5.3.3.2. Prüfung der logischen Konsistenz

Nachfolgend sollen die Eigenschaften einer eingeschränkt verteilten Wissensbasis $\mathfrak{K}$ ausgenutzt werden, um verteilt und effizient die Konsistenz von $\mathfrak{K}$ zu prüfen.

Theorem 5.3.3 (Konsistenz einer eingeschränkt verteilten Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte Wissensbasis. Dann gilt:*

$$\mathcal{K}_j \text{ ist konsistent für alle } j \iff \mathfrak{K} \text{ ist konsistent.}$$

Um die Konsistenz einer eingeschränkt verteilten Wissensbasis $\mathfrak{K}$ zu prüfen, reicht es somit aus, für jede Teil-Wissensbasis $\mathcal{K}_j, 1 \leq j \leq n$, unabhängig die Konsistenz zu prüfen.

Ähnlich zu partitioniert verteilten Wissensbasen wird das Theorem mit Hilfe des $\mathcal{SHIN}$-Tableaux-Algorithmus T gezeigt: $\mathfrak{K}$ ist genau dann konsistent, wenn T einen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}$ für $\mathfrak{K}$ konstruiert (Theorem 3.4.1). Daraus wird folgendes Korrollar abgeleitet:

Korrollar 5.3.1 (Existenz eines vollständigen und widerspruchsfreien Komplettierungswalds für $\mathfrak{K}$). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte Wissensbasis und seien $\mathcal{F}_1, \ldots, \mathcal{F}_n$ vollständige und widerspruchsfreie Komplettierungswälder, die T für $\mathcal{K}_1, \ldots, \mathcal{K}_n$ konstruiert. Genau dann, wenn diese existieren, konstruiert T auch für $\mathfrak{K}$ einen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}$.*

Für den Beweis des Korrollars sind folgende Vorüberlegungen nötig: Zunächst wird der Komplettierungswald $\mathcal{F}$ charakterisiert. $\mathcal{F}$ wird gemäß Definition 3.4.3 initialisiert. Dann wendet T schrittweise die Expansionsregeln aus Tabelle 3.4 auf Seite 47 an, um $\mathcal{F}$ zu vervollständigen. Dadurch kann der Konzeptmenge $\mathcal{L}(x)$ eines beliebigen Knotens x im Extremfall jede Unter-Konzeptbeschreibung einer Konzeptbeschreibung, die in der Wissensbasis verwendet wird, hinzugefügt werden. Deshalb lässt sich $\mathcal{L}(x)$ sicher charakterisieren mit der Invarianten $\mathcal{L}(x) \subseteq \mathsf{clos}(\mathfrak{K})$.

Ziel ist es, Zusammenhänge zwischen den von T für die Teil-Wissensbasen $\mathcal{K}_j$ konstruierten Komplettierungswäldern $\mathcal{F}_j$ und dem von T für die Gesamt-Wissensbasis $\mathfrak{K}$ konstruierten Komplettierungswald $\mathcal{F}$ festzustellen. Gibt es keine verteilt beschriebenen Individuen, dann entspricht $\mathcal{F}$ der Vereinigung der $\mathcal{F}_j, 1 \leq j \leq n$ (vergleiche Abschnitt 5.3.2).

Sobald es verteilt beschriebene Individuen gibt, gilt dies nicht mehr: Sei $i_\cap \in \mathsf{Ind}(\mathcal{K}_i)$ und $i_\cap \in \mathsf{Ind}(\mathcal{K}_j)$ für beliebige $i \neq j$. Dann kann eine Konzept-Zusicherung $C(i_\cap) \in \mathcal{A}_i$ dazu führen, dass $\mathcal{F}$ nicht mehr der Vereinigung der $\mathcal{F}_j, 1 \leq j \leq n$, entspricht und z. B. $\mathcal{L}(n(i_\cap)) \neq \mathcal{L}_j(n_j(i_\cap))$ entsteht, wobei $n(i_\cap)$ bzw. $n_j(i_\cap)$ den Knoten in $\mathcal{F}$ bzw. $\mathcal{F}_j$ bezeichnet, der $i_\cap$ repräsentiert. Dadurch kann es sein, dass $\mathcal{F}$ nicht mehr vollständig und widerspruchsfrei konstruiert wird. Nach Bedingung (5.1) für eine eingeschränkt verteilte Wissensbasis gilt jedoch $C(i_\cap) \in \mathcal{A}_i \Rightarrow C(i_\cap) \in \mathcal{A}_k$ für alle $1 \leq k \leq n$. Deshalb gilt bei Initialisierung $\mathcal{L}(n(i_\cap)) = \mathcal{L}_j(n_j(i_\cap))$.

Die Anwendung von Expansionsregeln kann in Kombination mit einer Rollen-Zusicherung $r(a, b)$ mit $a \in \mathsf{Ind}_\cap(\mathfrak{K}) \vee b \in \mathsf{Ind}_\cap(\mathfrak{K})$ wiederum dazu führen, dass $\mathcal{F}$ nicht mehr der Vereinigung der $\mathcal{F}_j, 1 \leq j \leq n$, entspricht. Dies wird im Folgenden genauer untersucht. Dazu wird bei den Expansionsregeln von T (siehe Tabelle 3.4 auf Seite 47) zwischen rollen-relevanten und rollen-irrelevanten Expansionsregeln unterschieden.

Rollen-relevante Expansionsregeln sind dadurch charakterisiert, dass sie in Kombination mit einer Rollen-Zusicherung Auswirkungen auf das Ergebnis von T haben können. Das sind genau die Expansionsregeln $\forall$, $\forall_+$, $\leqslant$ und $\leqslant_s$, denn die $\forall$- und $\forall_+$-Regeln

propagieren über Rollenzusicherungen neue Konzept-Zusicherungen zu den $\mathcal{L}$'s anderer Individuen. Die $\leqslant$- und $\leqslant_s$-Regeln hingegen nutzen Rollenzusicherungen, um einen Widerspruch zu erkennen, der durch die Verletzung einer Anzahlrestriktion entsteht.

Im Gegensatz dazu sind die $\exists$- und $\geqslant$-Expansionsregeln *rollen-irrelevant*: Sie verwenden keine bestehenden Rollen-Zusicherungen, sondern generieren neue Rollen-Zusicherungen mit anonymen Individuen.

Zusammengefasst heißt das für eine Rollenzusicherung in $\bigcup_j \mathcal{A}_j$, dass sie das Ergebnis einer Ausführung des Tableaux-Algorithmus nur dann beeinflussen kann, wenn sie von einer rollen-relevanten Expansionsregel genutzt wird. Mit diesen Vorüberlegungen kann nun das Korrollar gezeigt werden.

Beweis. T initialisiert die Komplettierungswälder $\mathcal{F}_j, 1 \leq j \leq n$, und $\mathcal{F}$ gemäß Definition 3.4.3. Weil $\mathfrak{K}$ eingeschränkt verteilt ist, gilt dann mit Bedingung (5.1), dass $\mathcal{L}(n(a)) = \mathcal{L}_j(n_j(a))$ für alle Individuen $a \in \mathsf{Ind}(\mathcal{K}_j), 1 \leq j \leq n$.

Die Expansionsregeln von T werden solange auf $\mathcal{L}$ in den Komplettierungswäldern angewendet, bis keine weitere Anwendung mehr möglich ist oder ein Widerspruch entsteht. Dabei muss zwischen verteilt und nicht verteilt beschriebenen Individuen unterschieden werden.

Sei $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$. $\mathcal{L}(n(i_\cap))$ kann nur dann erweitert werden, wenn es ein r' gibt, für das $i_\cap$ ein r'-Nachbar von einem $b \in \mathsf{Ind}(\mathfrak{K}), b \neq i_\cap$, ist und auf $\mathcal{L}(n(b))$ eine rollen-relevante Expansionsregel bezüglich r' angewendet werden kann. Weil $\mathfrak{K}$ eingeschränkt verteilt ist, gilt nach Bedingung (5.3) aber, dass kein $(\forall r'.D)$ in $\mathsf{clos}(\mathfrak{K})$ vorkommen kann. Deshalb kann es auch zu keiner Anwendung der $\forall$- und $\forall_+$-Regeln kommen. Auch nach Anwendung aller Expansionsregeln in $\mathcal{F}$ gilt also $\mathcal{L}(n(i_\cap)) = \mathcal{L}_j(n_j(i_\cap))$ mit $i_\cap \in \mathsf{Ind}(\mathcal{K}_j)$. Zusätzlich kann ein Widerspruch für $\mathcal{L}(n(i_\cap))$ bei Verletzung einer Anzahlrestriktion entstehen. Bedingung (5.2) stellt jedoch sicher, dass $(\leqslant_n\ r) \notin \mathsf{clos}(\mathfrak{K})$, falls $i_\cap$ r-Nachbarn hat. Deshalb tritt ein Widerspruch in $\mathcal{L}(n(i_\cap))$ genau dann auf, wenn ein Widerspruch in $\mathcal{L}_j(n_j(i_\cap))$ mit $i_\cap \in \mathsf{Ind}(\mathcal{K}_j)$ für ein j auftritt.

Sei $b \in \mathsf{Ind}(\mathcal{K}_j) \setminus \mathsf{Ind}_\cap(\mathfrak{K})$. $\mathcal{L}(n(b))$ kann nur dann von $\mathcal{L}_j(n_j(b))$ abweichen, wenn b ein r-Nachbar von einem $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ist und auf $\mathcal{L}(n(i_\cap))$ eine rollen-relevante Expansionsregel bezüglich r anwendbar ist. Da oben gezeigt wurde, dass immer $\mathcal{L}(n(i_\cap)) = \mathcal{L}_j(n_j(i_\cap))$, gilt nach Anwendung aller Expansionsregeln jedoch ebenfalls $\mathcal{L}(n(b)) = \mathcal{L}_j(n_j(b))$. Zusätzlich kann ein Widerspruch für $\mathcal{L}(n(b))$ bei Verletzung einer Anzahlrestriktion entstehen. Rollen-Zusicherungen zu b in $\mathcal{F}_j$ entsprechen genau den Rollenzusicherungen zu b in $\mathcal{F}$, da $b \notin \mathsf{Ind}_\cap(\mathfrak{K})$. Deshalb verletzt $\mathcal{L}(n(b))$ eine Anzahlrestriktion genau dann, wenn $\mathcal{L}_j(n_j(b))$ sie verletzt.

Für alle $a \in \mathsf{Ind}(\mathfrak{K})$ gilt also $\mathcal{L}(n(a)) = \mathcal{L}_j(n_j(a))$ mit $a \in \mathsf{Ind}(\mathcal{K}_j)$, und Widersprüche in einem $\mathcal{F}_j$ treten genau dann auf, wenn sie in $\mathcal{F}$ auftreten. Folglich konstruiert T genau dann für alle $\mathcal{K}_j$ einen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}_j$, wenn T für $\mathfrak{K}$ einen vollständigen und widerspruchsfreien Komplettierungswald $\mathcal{F}$ konstruiert. □

Wie sich im Verlauf des Beweises gezeigt hat, können durch die geschickte Wahl der leicht überprüfbaren Bedingungen an eine eingeschränkt verteilte Wissensbasis also alle Fälle, die ein weniger effizientes Verfahren erfordern würden, vermieden werden. Mit Korrollar 5.3.1 ist gleichzeitig auch Theorem 5.3.3 gezeigt. Im vorgeschlagenen

verteilten Reasoning-System wird das Verfahren zur Konsistenz-Prüfung von $\mathfrak{K}$ deshalb wie folgt umgesetzt: Jeder Reasoning-Knoten $R_j, 1 \leq j \leq n$, nutzt seinen lokalen Reasoning-Dienst, um die Konsistenz von $\mathcal{K}_j$ zu prüfen. Genau dann, wenn alle Teil-Wissensbasen $\mathcal{K}_j$ konsistent sind, ist $\mathfrak{K}$ ebenfalls konsistent.

5.3.3.3. Beantwortung von Anfrageatomen

Nachdem nun sichergestellt werden kann, dass die eingeschränkt verteilte Wissensbasis $\mathfrak{K}$ konsistent ist, beschäftigt sich dieser Abschnitt mit der Beantwortung von Anfrageatomen.

Beantwortung von Rollen-Anfrageatomen. Sei r eine nicht-transitive Rolle. Zur Beantwortung eines *Rollen-Anfrageatoms* $r(a, b)$ bezüglich $\mathfrak{K}$ muss dann nur für Teil-Wissensbasen $\mathcal{K}_k$ mit $a, b \in \mathsf{Ind}(\mathcal{K}_k)$ (und einer Rollenzusicherung zwischen diesen) geprüft werden, ob $\mathcal{K}_k \models r(a, b)$. Das ist ausreichend, da die TBox $\mathcal{T}$ in jeder Teil-Wissensbasis $\mathcal{K}_j$ verfügbar ist und deshalb auch die Rollenhierarchie und inverse Rollen berücksichtigt werden. Bei transitiven Rollen kann es sein, dass mehrere Teil-Wissensbasen überprüft werden müssen. Dazu werden Mechanismen benötigt, die in Abschnitt 5.3.4 beschrieben werden.

Für die Beantwortung von *Konzept-Anfrageatomen* $C(a)$ bezüglich $\mathfrak{K}$ muss zwischen nicht verteilt beschriebenen Individuen $a \in \mathsf{Ind}(\mathfrak{K}) \setminus \mathsf{Ind}_\cap(\mathfrak{K})$ und verteilt beschriebenen Individuen $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ unterschieden werden:

Beantwortung von Konzept-Anfrageatomen für nicht verteilt beschriebene Individuen. Bei einem nicht verteilt beschriebenen Individuum a reicht es zur Instanz-Prüfung bezüglich $\mathfrak{K}$ aus, eine Instanz-Prüfung in der Teil-Wissensbasis $\mathcal{K}_k$ mit $a \in \mathsf{Ind}(\mathcal{K}_k)$ durchzuführen.

Theorem 5.3.4 (Instanz-Prüfung bei eingeschränkt verteilter Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte und konsistente Wissensbasis, C eine eingeschränkte Konzeptbeschreibung und $a \in \mathsf{Ind}(\mathfrak{K}) \setminus \mathsf{Ind}_\cap(\mathfrak{K})$ ein nicht verteilt beschriebenes Individuum. Dann gilt:*

$$\mathfrak{K} \models C(a) \iff \mathcal{K}_j \models C(a) \text{ mit } a \in \mathsf{Ind}(\mathcal{K}_j) \setminus \mathsf{Ind}_\cap(\mathfrak{K}) \text{ für ein } 1 \leq j \leq n.$$

Da die Aussage $\mathfrak{K} \models C(a)$ äquivalent ist zu der Aussage, dass $\mathfrak{K} \cup \{\neg C(a)\}$ inkonsistent ist, wird daraus folgendes Korrollar abgeleitet:

Korrollar 5.3.2 (Instanz-Prüfung bei eingeschränkt verteilter Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte und konsistente Wissensbasis, C eine eingeschränkte Konzeptbeschreibung und $a \in \mathsf{Ind}(\mathfrak{K}) \setminus \mathsf{Ind}_\cap(\mathfrak{K})$ ein nicht verteilt beschriebenes Individuum. Dann konstruiert der Tableaux-Algorithmus T genau dann keinen vollständigen und widerspruchsfreien Komplettierungswald für $\mathfrak{K} \cup \{\neg C(a)\}$, wenn T auch keinen vollständigen und widerspruchsfreien Komplettierungswald für $\mathcal{K}_j \cup \{\neg C(a)\}$ mit $a \in \mathsf{Ind}(\mathcal{K}_j)$ konstruiert.*

Beweis. Da $\mathfrak{K}$ konsistent ist, konstruiert T einen vollständigen und widerspruchsfreien Komplettierungswald für $\mathfrak{K}$ und $\mathcal{K}_j$. Damit T für $\mathfrak{K} \cup \{\neg C(a)\}$ keinen vollständigen und widerspruchsfreien Komplettierungswald konstruiert, muss also durch die Anwendung von Expansionsregeln auf $\mathcal{L}(n(a)) \cup \{\neg C\}$ ein Widerspruch ausgelöst werden. Da C eine eingeschränkte Konzeptbeschreibung ist, kann sich dadurch jedoch keine Auswirkung auf ein $\mathcal{L}(i_\cap)$ mit $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ergeben (vergleiche Beweis zu Korrollar 5.3.1). Der Widerspruch muss deshalb in dem Teil des Komplettierungswaldes auftreten, der die Teil-Wissensbasis $\mathcal{K}_j$ repräsentiert. Deshalb tritt dieser Widerspruch genau dann auf, wenn T auch für $\mathcal{K}_j \cup \{\neg C(a)\}$ keinen vollständigen und widerspruchsfreien Komplettierungswald konstruiert. □

Aus Korrollar 5.3.2 folgt unmittelbar Theorem 5.3.4.

Beantwortung von Konzept-Anfrageatomen für verteilt beschriebene Individuen. Für ein verteilt beschriebenes Individuum $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ reicht es zur Beantwortung von Konzept-Anfrageatomen nicht aus, nur eine Teil-Wissensbasis zu betrachten. Stattdessen müssen Zusicherungen zu $i_\cap$ aus allen Teil-Wissensbasen $\mathcal{K}_j$ mit $i_\cap \in \mathsf{Ind}(\mathcal{K}_j)$ integriert werden.

Im Folgenden wird ein Ansatz entwickelt, bei dem für das verteilt beschriebene Individuum zunächst eine temporäre ABox aus den verteilt vorliegenden Zusicherungen generiert wird. Diese wird als *Wissensbasis-Kern* bezeichnet. Zur Beantwortung des Konzept-Anfrageatoms reicht es dann aus, lediglich den Wissensbasis-Kern zu betrachten.

Definition 5.3.5 (Wissensbasis-Kern $[\mathfrak{K}]_{i_\cap}$). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte und konsistente Wissensbasis und $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ein verteilt beschriebenes Individuum. Dann ist der* Wissensbasis-Kern $[\mathfrak{K}]_{i_\cap}$ *von $i_\cap$ eine ABox, die folgendermaßen definiert ist:*

$$
\begin{aligned}
[\mathfrak{K}]_{i_\cap} = \{\ & \mathsf{msc}^{\mathcal{K}_j}_{i_\cap}(i_\cap) \mid i_\cap \in \mathsf{Ind}(\mathcal{K}_j)\ \} \cup \\
& \{\ \mathsf{msr}^{\mathcal{K}_i}_{(i_\cap,b)}(i_\cap, b), \mathsf{msc}^{\mathcal{K}_i}_{b}(b) \mid b \textit{ ist Nachbar von } i_\cap \textit{ in } \mathcal{K}_i\ \} \cup \\
& \{\ b \neq b' \mid b \textit{ ist Nachbarn von } i_\cap \textit{ in } \mathcal{K}_i, b \neq b' \in \mathcal{A}_i\}.
\end{aligned}
$$

Jeder Wissensbasis-Kern bezieht sich also auf ein bestimmtes verteilt beschriebenes Individuum $i_\cap$. Zusicherungen zu $i_\cap$ in den einzelnen Teil-Wissensbasen $\mathcal{K}_j$, $1 \leq j \leq n$, werden in $[\mathfrak{K}]_{i_\cap}$ konzentriert, indem zu jedem Nachbarn b das spezifischste Konzept $\mathsf{msc}^{\mathcal{K}_j}_b$ mittels Konzept-Realisierung (siehe Definition 3.3.1) und zu jeder Rollen-Zusicherung die spezifischste Rolle $\mathsf{msr}^{\mathcal{K}_j}_{(i_\cap,b)}$ mittels Rollen-Realisierung (siehe Definition 3.3.2) zugesichert werden. Wie man leicht sieht, ist für ein konsistentes $\mathfrak{K}$ auch der Wissensbasis-Kern $[\mathfrak{K}]_{i_\cap}$ für ein $i_\cap$ konsistent. Auf Basis eines derartigen Wissensbasis-Kerns können Konzept-Anfrageatome zu dem verteilt beschriebenen Individuum effizient beantwortet werden.

Theorem 5.3.5 (Verteilte Instanz-Prüfung bei eingeschränkt verteilter Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte und konsistente Wissensbasis und $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ein verteilt beschriebenes Individuum. Seien $C_\cap$ und $D_\cap$ wie folgt*

definierte Konzeptbeschreibungen (mit $A \in N_C$, r eine Rolle, die nicht eingehend verteilt beschreibend bezüglich $\mathfrak{K}$ ist, und s eine einfache Rolle):

$$
\begin{array}{lll}
C_\cap, D_\cap & \rightarrow & A \mid \quad \textit{(Konzeptname)} \\
 & & C_\cap \sqcap D_\cap \mid \quad \textit{(Schnittmenge)} \\
 & & C_\cap \sqcup D_\cap \mid \quad \textit{(Vereinigung)} \\
 & & \neg A \mid \quad \textit{(Negation)} \\
 & & \forall r.A \mid \quad \textit{(Wertebeschränkung)} \\
 & & \exists r.A \mid \quad \textit{(Existenzquantifizierung)} \\
 & & \leqslant_n s.\top \mid \quad \textit{(obere Anzahlrestriktion)} \\
 & & \geqslant_n s.\top \quad \textit{(untere Anzahlrestriktion).}
\end{array}
$$

Dann gilt:

$$\mathfrak{K} \models C_\cap(i_\cap) \iff [\mathfrak{K}]_{i_\cap} \models C_\cap(i_\cap).$$

Auch hier lässt sich wieder folgendes Korrollar ableiten.

Korrollar 5.3.3 (Instanz-Prüfung bei eingeschränkt verteilter Wissensbasis). *Sei $\mathfrak{K} = \bigcup_j \mathcal{K}_j$ eine eingeschränkt verteilte und konsistente Wissensbasis, $C_\cap$ eine wie in Theorem 5.3.5 definierte Konzeptbeschreibung und $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ ein verteilt beschriebenes Individuum. Dann konstruiert der Tableaux-Algorithmus T genau dann keinen vollständigen und widerspruchsfreien Komplettierungswald für $\mathfrak{K} \cup \{\neg C_\cap(i_\cap)\}$, wenn T auch keinen vollständigen und widerspruchsfreien Komplettierungswald für $[\mathfrak{K}]_{i_\cap} \cup \{\neg C_\cap(i_\cap)\}$ konstruiert.*

Beweis. Da $\mathfrak{K}$ konsistent ist, konstruiert T einen vollständigen und widerspruchsfreien Komplettierungswald für $\mathfrak{K}$. Das Gleiche gilt für $[\mathfrak{K}]_{i_\cap}$ nach Definition 5.3.5.

Damit T keinen vollständigen und widerspruchsfreien Komplettierungswald für $\mathfrak{K} \cup \{\neg C_\cap(i_\cap)\}$ konstruiert, muss die Anwendung von Expansionsregeln auf $\mathcal{L}(n(i_\cap)) \cup \{\neg C_\cap\}$ einen Widerspruch auslösen. Dabei wird nach den unterschiedlichen Formen von $C_\cap$ unterschieden:

- Sei $C_\cap$ von der Form A. Dann muss der Widerspruch für $\mathcal{L}(n(i_\cap)) \cup \{\neg A\}$ auftreten, es muss also gelten $A \in \mathcal{L}(n(i_\cap))$. Für alle $D \in \mathcal{L}(n(i_\cap))$ gilt $\mathfrak{K} \models D(i_\cap)$ und deshalb insbesondere $\mathfrak{K} \models A(i_\cap)$. Mit $\mathcal{L}(n(i_\cap)) = \mathcal{L}_j(n_j(i_\cap))$ gilt auch $\mathcal{K}_j \models A(i_\cap)$. Für $\mathsf{msc}_{i_\cap}^{\mathcal{K}_j}$ gilt nach Definition 3.3.1, dass $\mathcal{K}_j \models D(i_\cap) \Rightarrow \mathcal{K}_j \models \mathsf{msc}_{i_\cap}^{\mathcal{K}_j} \sqsubseteq D$, also insbesondere $\mathcal{K}_j \models \mathsf{msc}_{i_\cap}^{\mathcal{K}_j} \sqsubseteq A$. Bezeichne mit $[\mathcal{L}]([n](a))$ die Menge der Konzeptbeschreibungen für ein Individuum a im Komplettierungswald für $[\mathfrak{K}]_{i_\cap}$. Nach Definition 5.3.5 gilt $\mathsf{msc}_{i_\cap}^{\mathcal{K}_j} \in [\mathcal{L}]([n](i_\cap))$. Mit $\mathcal{K}_j \models \mathsf{msc}_{i_\cap}^{\mathcal{K}_j} \sqsubseteq A$ tritt genau dann auch für $[\mathcal{L}]([n](i_\cap)) \cup \{\neg A\}$ ein Widerspruch auf.

- Der Fall $C_\cap$ von der Form $\neg A$ ist analog zu $C_\cap$ von der Form A.

- Sei $C_\cap$ von der Form $C'_\cap \sqcap C''_\cap$. Dann muss der Widerspruch durch $\mathcal{L}(n(i_\cap)) \cup \{\neg C'_\cap\}$ oder $\mathcal{L}(n(i_\cap)) \cup \{\neg C''_\cap\}$ ausgelöst werden und lässt sich auf diese Fälle zurückführen.

- Sei $C_\cap$ von der Form $C'_\cap \sqcup C''_\cap$. Dann muss der Widerspruch durch $(\mathcal{L}(n(i_\cap)) \cup \{\neg C'_\cap\}) \cup \{\neg C''_\cap\}$ ausgelöst werden und lässt sich ebenfalls auf diese Fälle zurückführen.

- Sei $C_\cap$ von der Form $\exists\, r.A$. Dann muss der Widerspruch durch $\mathcal{L}(n(i_\cap)) \cup \{\forall\, r.(\neg A)\}$ ausgelöst werden, also bei mindestens einem r-Nachbarn a durch $\mathcal{L}(n(a)) \cup \{\neg A\}$. Folglich gilt $\mathfrak{K} \models A(a)$. Da r keine eingehende verteilt beschreibende Rolle ist, kann a nicht verteilt beschrieben sein, d. h. $a \in \mathsf{Ind}(\mathcal{K}_j) \setminus \mathsf{Ind}_\cap(\mathfrak{K})$ für ein j. Nach Theorem 5.3.4 gilt deshalb auch $\mathcal{K}_j \models A(a)$. Für $\mathsf{msc}_a^{\mathcal{K}_j}$ gilt nach Definition 3.3.1 für alle D, dass $\mathcal{K}_j \models D(a) \Rightarrow \mathcal{K}_j \models \mathsf{msc}_a^{\mathcal{K}_j} \sqsubseteq D$, also insbesondere $\mathcal{K}_j \models \mathsf{msc}_a^{\mathcal{K}_j} \sqsubseteq A$. Nach Definition 5.3.5 gilt $\mathsf{msc}_a^{\mathcal{K}_j} \in [\mathcal{L}]([n](a))$. Mit $\mathcal{K}_j \models \mathsf{msc}_a^{\mathcal{K}_j} \sqsubseteq A$ tritt genau dann auch für $[\mathcal{L}]([n](a)) \cup \{\neg A\}$ ein Widerspruch auf.

- Sei $C_\cap$ von der Form $\forall\, r.A$. Dann muss der Widerspruch durch $\mathcal{L}(n(i_\cap)) \cup \{\exists\, r.(\neg A)\}$ ausgelöst werden. Die $\exists$-Regel führt dazu, dass ein anonymer r-Nachbar von $i_\cap$ erzeugt wird. Dann kann nur die Anwendbarkeit von $\forall$- und $\forall_+$-Regeln auf $\mathcal{L}(n(i_\cap))$ einen Widerspruch für den anonymen r-Nachbarn auslösen. Die $\leqslant$- und $\leqslant_s$-Regeln können nicht anwendbar sein, weil $\mathfrak{K}$ eingeschränkt verteilt und $i_\cap$ verteilt beschrieben ist. Deshalb tritt ein Widerspruch im anonymen r-Nachbarn genau dann auf, wenn der Widerspruch in $[\mathfrak{K}]_{i_\cap}$ auftritt.

- Sei $C_\cap$ von der Form $\leqslant_n \; r$. Dann muss der Widerspruch durch $\mathcal{L}(n(i_\cap)) \cup \{\geqslant_{n+1} \; r\}$ entstehen, d. h. $i_\cap$ hat höchstens n r-Nachbarn. Da jeder r-Nachbar von $i_\cap \in \mathsf{Ind}_\cap(\mathfrak{K})$ gemäß Definition 5.3.5 auch ein r-Nachbar von $i_\cap$ in $[\mathfrak{K}]_{i_\cap}$ ist, tritt derselbe Widerspruch genau dann auch für $[\mathcal{L}]([n](i_\cap))$ auf.

- Der Fall $C_\cap$ von der Form $\geqslant_n \; r$ ist analog zu $C_\cap$ von der Form $\leqslant_n \; r$.

Damit wurde gezeigt, dass T für alle Formen von $C_\cap$ genau dann keinen vollständigen und widerspruchsfreien Komplettierungswald für $\mathfrak{K} \cup \{\neg C_\cap(i_\cap)\}$ konstruiert, wenn T keinen vollständigen und widerspruchsfreien Komplettierungswald für $[\mathfrak{K}]_{i_\cap} \cup \{\neg C_\cap(i_\cap)\}$ konstruiert. □

Aus Korrollar 5.3.3 folgt unmittelbar Theorem 5.3.5.

Diese vorgestellten Zusammenhänge zur Beantwortung von Anfrageatomen werden im verteilten Reasoning-System wie folgt umgesetzt: Bei einem Rollen-Anfrageatom $r(a, b)$ identifiziert der Reasoning-Knoten R_k, der die Ausgangsanfrage erhält, die Reasoning-Knoten R_l mit $a, b \in \mathsf{Ind}(\mathcal{K}_l)$ und einer Rollen-Zusicherung zwischen diesen und nutzt ihre Schnittstelle, um Rollen-Prüfung, -Abfrage oder -Realisierung auf $\mathcal{K}_l$ durchzuführen. Bei Konzept-Anfrageatomen muss R_k zwischen verteilt und nicht verteilt beschriebenen Individuen unterscheiden: Für ein nicht verteilt beschriebenes Individuum a identifiziert R_k den Reasoning-Knoten R_l mit $a \in \mathsf{Ind}(\mathcal{K}_l)$ und nutzt die Schnittstelle von R_l, um eine Instanz-Prüfung, -Abfrage oder Konzept-Realisierung auf $\mathcal{K}_l$ durchzuführen. Für ein verteilt beschriebenes Individuum $i_\cap$ muss R_k alle Reasoning-Knoten R_l mit $i_\cap \in \mathsf{Ind}(\mathcal{K}_l)$ identifizieren. Dann nutzt R_k die Schnittstelle der R_l, um mittels Rollen-Abfrage, Konzept- und Rollen-Realisierung das $[\mathfrak{K}]_{i_\cap}$ zu generieren. Schließlich wendet R_k seinen lokalen Reasoning-Dienst auf $[\mathfrak{K}]_{i_\cap}$ an, um das Konzept-Anfrageatom zu $i_\cap$ zu beantworten.

Für eine eingeschränkt verteilte Wissensbasis $\mathfrak{K}$ kann also ein Kompromiss zwischen den verteilt durchführbaren und effizienten Verfahren bei partitioniert verteilten Wissensbasen und der Ausdrucksstärke bei beliebig verteilten Wissensbasen gefunden werden: Die Konsistenz-Prüfung, die Beantwortung von Rollen-Anfrageatomen und die Beantwortung von Konzept-Anfrageatomen für nicht verteilt beschriebene Individuen kann wie bei partitioniert verteilten Wissensbasen umgesetzt werden; die Reasoning-Dienste können unabhängig voneinander auf den Teil-Wissensbasen durchgeführt werden. Für die Beantwortung von Konzept-Anfrageatomen für verteilt beschriebene Individuen muss mehr Aufwand betrieben werden. In diesem Fall muss aus allen Teil-Wissensbasen, in denen das verteilt beschriebene Individuum vorkommt, der Wissensbasis-Kern generiert werden. Trotzdem kann damit vermieden werden, die gesamte Wissensbasis zu zentralisieren, wie es bei einer beliebig verteilten Wissensbasis nötig wäre.

Insgesamt stellen eingeschränkt verteilte Wissensbasen also eine Möglichkeit dar, die wichtige Anforderung A5.1.1 zu erfüllen, nach der Kontextinformationen zu demselben Überwachungsobjekt über mehrere Teil-Wissensbasen verteilt sein können. Der folgende Abschnitt führt im Detail aus, wie die in Kapitel 4 entwickelten Entwurfsmuster mit eingeschränkt verteilten Wissensbasen umgesetzt werden können.

5.3.3.4. Vereinbarkeit der Entwurfmuster mit eingeschränkt verteilten Wissensbasen

Damit die Vorteile von eingeschränkt verteilten Wissensbasen bei der Zustandsüberwachung für Infrastrukturnetze ausgenutzt werden können, muss gezeigt werden, dass die in Abschnitt 4.4.2 vorgeschlagenen Entwurfsmuster damit vereinbar sind. Dort entsprechen die verteilt beschriebenen Individuen den Instanzen der Überwachungsobjektkonzepte. Instanzen von Merkmals-, Symptom-, Fehler- und Zustandskonzepten sind hingegen nicht verteilt beschriebene Individuen.

Die in Definition 5.3.4 für eine eingeschränkt verteilte Wissensbasis formulierten Bedingungen beziehen sich alle auf verteilt beschriebene Individuen, also Überwachungsobjektinstanzen. Sie sind mit dem Entwurfsmuster wie folgt vereinbar:

Bedingung (5.1). $C(i_\cap) \in \mathcal{A}_i$ für ein $1 \leq i \leq n \implies C(i_\cap) \in \mathcal{A}_j$ für ein $1 \leq j \leq n$.

Für dieselbe Überwachungsobjektinstanz müssen also in allen Teil-Wissensbasen dieselben Konzept-Zusicherungen bekannt sein. Ist beispielsweise für eine Instanz i in einer Teil-Wissensbasis $\mathcal{K}_j$ mit $\mathsf{Train}(i)$ zugesichert, dass i einen Zug repräsentiert, muss dies auch in allen anderen Teil-Wissensbasen $\mathcal{K}_k, k \neq j$, zugesichert sein.

Das in Abschnitt 4.5 beschriebene Architektur-Entwurfsmuster stellt sicher, dass dazu einheitliche Konzeptbeschreibungen verwendet werden: Diese werden im Ontologiemodul *MonitoringSubjectDomainOntology* spezifiziert. In Abschnitt 4.4.3 wurde außerdem beschrieben, wie ein Namensdienst zur Laufzeit gewährleisten kann, dass einheitliche Instanzbezeichner und Konzept-Zusicherungen verwendet werden.

Bedingungen (5.2) und (5.3). *In TBox-Axiomen und ABox-Zusicherungen werden nur eingeschränkte Konzeptbeschreibungen bezüglich $\mathfrak{K}$ verwendet.*

Die Einschränkung von Konzeptbeschreibungen bezieht sich nur auf verteilt beschreibende Rollen – also Rollen, für die ein verteilt beschriebenes Individuum einen Nachbar hat bzw. ein Nachbar ist (siehe Definition 5.3.3). Bezogen auf das Entwurfsmuster bedeutet dies:

- $\forall\, \mathsf{refersToMonitoringSubject}.C$ darf kein Unterkonzept einer Konzeptbeschreibung in ABox und TBox sein.
- $\leqslant_n\ \mathsf{hasState}$ darf kein Unterkonzept einer Konzeptbeschreibung in ABox und TBox sein.

Gemäß seiner Definition in Abschnitt 4.4.2 verwendet das Entwurfsmuster keine derartigen Konzeptbeschreibungen. Somit kann es in einer eingeschränkt verteilten Wissensbasis umgesetzt werden.

Durch den Einsatz der in Kapitel 4 vorgeschlagenen Entwurfsmuster wird also sichergestellt, dass die resultierende Wissensbasis eingeschränkt verteilt ist. Zur Konsistenz-Prüfung und korrekten und vollständigen Beantwortung von Anfrageatomem können deshalb die in diesem Abschnitt beschriebenen verteilt durchführbaren und effizienten Verfahren eingesetzt werden.

5.3.4. Beantwortung konjunktiver Anfragen

Bisher wurde gezeigt, wie bei unterschiedlich verteilten Wissensbasen einzelne Anfrageatome beantwortet werden können. Dies wird nun erweitert auf die Beantwortung von Konjunktionen von Anfrageatomen, also konjunktiven Anfragen. Wie in Abschnitt 5.3 erläutert wird dabei angenommen, dass alle Variablen benannte Variablen sind.

Syntax und Semantik konjunktiver Anfragen bezüglich verteilter Wissensbasen wurde bereits in den Definitionen 5.1.3 und 5.1.4 eingeführt. Dort werden die Lösungen einer konjunktiven Anfrage *deklarativ* definiert. Damit kann für einen gegebenen Lösungskandidaten überprüft werden, ob es sich um eine Lösung handelt; die Definition gibt jedoch keinen Anhaltspunkt dafür, wie eine Lösung *prozedural* berechnet werden kann.

Dies ist vergleichbar mit der Anfragebeantwortung im klassischen relationalen Datenmodell: Dort wird der *Relationenkalkül* verwendet, um die gesuchten Lösungen *deklarativ* zu beschreiben. Sichere Kalkülausdrücke lassen sich äquivalent dazu mit der gleichmächtigen *relationalen Algebra* ausdrücken. Die relationale Algebra beruht auf der Verknüpfung von Operatoren auf Relationen und beschreibt dadurch *prozedural*, wie die Lösungen zu einer gegebenen Anfrage berechnet werden können [RG03].

Zur Anfragebearbeitung der hier behandelten konjunktiven Anfragen muss also zusätzlich zur deklarativen Definition der Lösung eine entsprechende *logische Algebra* [Gra93] angegeben werden. Diese kann analog zu der Untermenge der relationalen Algebra gewählt werden, die nur die Operatoren *Selektion*, *Projektion* und *Join* umfasst.

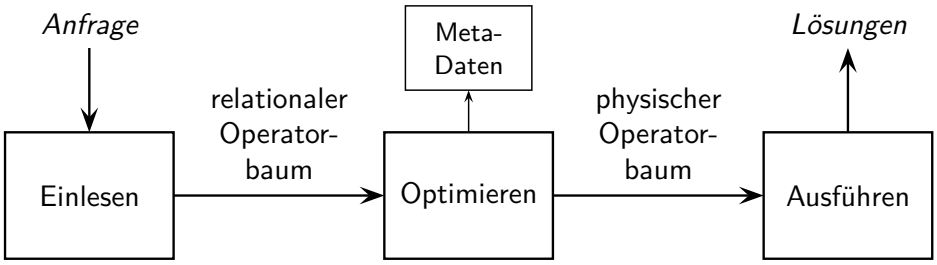

Abbildung 5.3.: Grundlegende Phasen der Anfragebearbeitung (nach [RG03]).

Formal kann die logische Algebra für die hier untersuchten konjunktiven Anfragen wie folgt definiert werden: Die Semantik der vorgestellten Anfrageatome ist äquivalent zu der Semantik der entsprechenden Anfrageatome in SPARQL-DL [SP07]. Deshalb kann die in [SP07] definierte abstrakte Sytax und ihre Abbildung in SPARQL-Syntax zur Repräsentation der hier relevanten konjunktiven Anfragen genutzt werden. Die dafür relevante Untermenge der SPARQL-Syntax lässt sich analog zu [Cyg05] vollständig in die relationale Algebra transformieren. Jede hier untersuchte konjunktive Anfrage lässt sich also in dieser Untermenge der relationalen Algebra darstellen. Die Komplexität der Evaluierung eines SPARQL-Patterns wurde als PSPACE-vollständig analysiert [PAG06].

Phasen der Anfragebearbeitung. Die Beantwortung einer konjunktiven Anfrage kann dann in drei grundlegende Phasen gegliedert werden [RG03, Gra93] (siehe auch Abbildung 5.3):

1. Einlesen der Anfrage
2. Generierung und Optimierung des Anfrageplans
3. Ausführung des Anfrageplans

Übertragen auf das verteilte Reasoning im vorgestellten verteilten Reasoning-System lassen sich diese Phasen umsetzen wie im Folgenden beschrieben.

5.3.4.1. Einlesen der Anfrage

Die konjunktive Anfrage wird eingelesen und in relationaler Algebra repräsentiert. Der Ausdruck der relationalen Algebra kann aus Gründen der Übersichtlichkeit auch als sogenannter relationaler Operatorbaum dargestellt werden (siehe Abbildung 5.4 für ein Beispiel): Jedes Blatt entspricht einer Menge von ABox-Zusicherungen (also einer ABox); die inneren Knoten entsprechen den relationalen Operatoren *Selektion* (σ, unär), *Projektion* (π, unär) oder *Join* ($\bowtie$, binär). Kanten geben die Verknüpfung der Operatoren an und illustrieren damit den Datenfluss.

Zum Beispiel wird ein Konzept-Anfrageatom $C(t)$ mit $t \in N_V$ als Variable relational repräsentiert als $\pi_{t \leftarrow i}(\sigma_{i \in C}(\mathcal{A}))$. Die konjunktive Anfrage $C(t) \wedge r(t, i15)$ entspricht beispielsweise $\pi_{t \leftarrow i}(\sigma_{i \in C}(\mathcal{A})) \bowtie \pi_{t \leftarrow i}(\sigma_{(i,i15) \in r}(\mathcal{A}))$

Ergebnis dieser Phase ist also eine Repräsentation der gegebenen konjunktiven Anfrage als relationaler Operatorbaum.

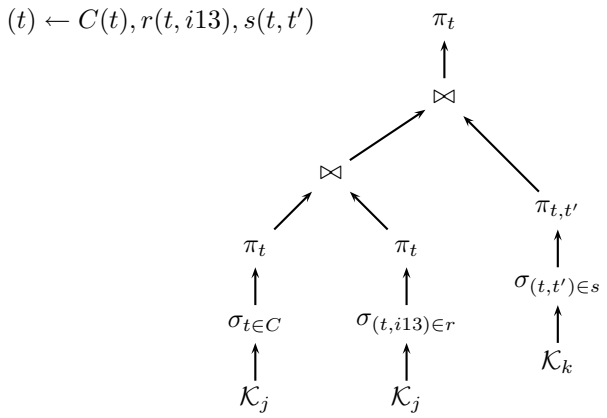

Abbildung 5.4.: Beispiel für einen relationalen Operatorbaum.

5.3.4.2. Generierung und Optimierung des Anfrageplans

In dieser Phase wird aus dem relationalen Operatorbaum ein Anfrageplan (d. h. physischer Operatorbaum) generiert. Der generierte Anfrageplan ist nicht eindeutig, da es für einen gegebenen relationalen Operatorbaum eine Vielzahl unterschiedlicher Anfragepläne geben kann. Dies liegt daran, dass (i) jeder relationale Operatorbaum in viele andere, logisch äquivalente relationale Operatorbäume transformiert werden kann (z. B. aufgrund der Assoziativität und Kommutativität des Join-Operators) und (ii) für jeden Operator im relationalen Operatorbaum mehrere physische Operatoren in Frage kommen können (z. B. unterschiedliche Möglichkeiten, den Join-Operator umzusetzen).

Die Generierung eines Anfrageplans aus einem gegebenen relationalen Operatorbaum kann also auch als Suchproblem betrachtet werden, das zum Ziel hat, einen optimalen Anfrageplan zu finden. Die Optimalität kann sich dabei auf verschiedene Kriterien oder Kombination von diesen beziehen wie z. B. minimale Antwortzeit für alle Lösungen, minimale Wartezeit bis zur ersten Lösung, minimale Anzahl der Kommunikationsoperationen, minimaler Speicherbedarf für die Anfragebearbeitung usw. Zur Lösung dieses Suchproblems werden häufig Heuristiken eingesetzt, wobei in der Regel auch Meta-Daten über die zugrunde liegenden Daten berücksichtigt werden (vergleiche Abbildung 5.3).

Sei also ein relationaler Operatorbaum gegeben. Für den Join-Operator können klassische Implementierungen wie z. B. *Nested-Loop* oder *Sort-Merge* verwendet werden. Auch für den Projektions-Operator eigenen sich klassische Implementierungen. Besondere Sorgfalt erfordert jedoch der Selektions-Operator.

Sowohl Konzept- als auch Rollen-Anfrageatome werden in der relationalen Algebra – wie in der vorherigen Phase beschrieben – als Selektions-Operatoren dargestellt. Die Umsetzung der logischen Selektions-Operatoren als physische Selektions-Operatoren hängt von der Verteiltheit der Wissensbasis ab. Für das eingeführte verteilte Reasoning-System wurden die Beantwortung von Anfrageatomen für beliebig, partitioniert und eingeschränkt verteilte Wissensbasen untersucht. Für jede dieser Arten von Verteiltheit werden nun physische Selektions-Operatoren vorgestellt:

Beliebig verteilte Wissensbasis. Der Selektions-Operator muss auf der gesamten verteilten Wissensbasis arbeiten, um korrekte und vollständige Ergebnisse sicherzu-

stellen (siehe Abschnitt 5.3.1).

Aus diesem Grund aggregiert der physische Selektions-Operator für beliebig verteilte Wissensbasen $\sigma^{\text{bel.}}$ zunächst alle Teil-Wissensbasen und führt die Selektion dann mit dem lokalen Reasoning-Dienst auf der aggregierten Gesamt-Wissensbasis durch.

Partitioniert verteilte Wissensbasis. Ist ein Individuum im Anfrageatom gegeben, dann muss der Selektions-Operator nur auf die Teil-Wissensbasis angewendet werden, in der das Individuum vorkommt. Ist kein Individuum angegeben, muss jede Teil-Wissensbasis separat bearbeitet werden (siehe Abschnitt 5.3.2).

Der physische Selektions-Operator für partitionierte Verteilung $\sigma^{\text{part.}}$ muss zunächst die relevanten Teil-Wissensbasen ermitteln (siehe dazu den folgenden Abschnitt). Dann delegiert er die Auswertung der Selektion an jede dieser Teil-Wissensbasen. Da jede dieser Selektionen unabhängig voneinander ist, können sie auch parallelisiert werden. Jede Teil-Wissensbasis führt die Selektion mit dem lokalen Reasoning-Dienst aus und liefert die selektierten Individuen als Lösungen für diese Selektion zurück. Diese werden vereinigt und Duplikate entfernt.

Eingeschränkt verteilte Wissensbasis. Hier wird angenommen, dass ein Individuum i gegeben ist. Es muss entschieden werden, ob es sich um ein verteilt oder nicht verteilt beschriebenes Individuum handelt. Dann werden die Anfrageatome wie in Abschnitt 5.3.3 gezeigt beantwortet:

Für ein nicht verteilt beschriebenes Individuum ist der physische Selektions-Operator für eingeschränkte Verteilung $\sigma^{\text{e. vert.}}_{\text{nicht-vert.}}$ identisch mit dem Operator $\sigma^{\text{part.}}$ für partitioniert verteilte Wissensbasen.

Für ein verteilt beschriebenes Individuum wird $\sigma^{\text{e. vert.}}_{\text{vert.}}$ als physischer Selektions-Operator für eingeschränkte Verteilung eingesetzt. Entsprechend Theorem 5.3.5 wird zunächst der Wissensbasis-Kern $[\mathfrak{K}]_i$ aggregiert und die Selektion schließlich mit dem lokalen Reasoning-Dienst auf $[\mathfrak{K}]_i$ durchgeführt, um die Lösungen zu ermitteln.

Bei der Anfragebearbeitung kann davon ausgegangen werden, dass die Art der Verteiltheit der Wissensbasis bekannt ist. Somit können die im relationalen Operatorbaum enthaltenen Operatoren für Selektion, Projektion und Join alle in einen physischen Operatorbaum übersetzt werden.

Nachfolgend wird beschrieben, wie für einen gegebenen logischen Selektions-Operator die relevanten Teil-Wissensbasen ermittelt werden. Dies ist notwendig für partitioniert und eingeschränkt verteilte Wissensbasen.

Identifikation der relevanten Teil-Wissensbasen. Zur Ermittlung der relevanten Teil-Wissensbasen werden Meta-Daten über die Teil-Wissensbasen benötigt. Es wird angenommen, dass diese von einer logisch gesehen zentralen Registratur (oft auch Verzeichnis genannt) bereitgestellt werden. Meta-Daten zu Teil-Wissensbasen werden *Profile* genannt und für die Teil-Wissensbasis $\mathcal{K}_j$ z. B. als $\mathsf{Profile}(\mathcal{K}_j)$ bezeichnet.

Profile können unterschiedlich spezifiziert werden. Dabei ergibt sich ein prinzipieller Konflikt zwischen dem Aufwand, der zur *Generierung* der Profile nötig ist, und dem Aufwand, der bei der *Nutzung* der Profile für die Anfragebeantwortung entsteht.

Das eine Extrem ist, dass ein Profil vollständige Informationen über die Teil-Wissensbasis enthält. Dadurch wird es möglich, bei der Anfragebearbeitung optimale Entscheidungen bezüglich der relevanten Teil-Wissensbasen zu treffen. Andererseits ist die Generierung der Profile sehr aufwändig, weil sie viele Ressourcen benötigt und jede Veränderung an der Teil-Wissensbasis auch eine Veränderung am Profil nach sich zieht. Das andere Extrem ist es, fast gar keine Informationen ins Profil zu übernehmen. Dadurch ist die Generierung von Profilen sehr effizient, und es sind selten Aktualisierungen nötig. Andererseits ist es mit derartigen Profilen nur sehr ungenau möglich, relevante Teil-Wissensbasen zu identifizierten, und die Anfragebeantwortung kann dadurch ineffizient werden.

Hier wird ein Kompromiss vorgeschlagen, der dazu führt, dass Profile sowohl schnell generiert und selten aktualisiert werden müssen als auch effektiv zur Generierung effizienter Anfragepläne genutzt werden können. Dazu wird $\mathsf{Profile}(\mathcal{K}_j)$ für die Teil-Wissensbasis $\mathcal{K}_j$ wie folgt definiert:

Definition 5.3.6 (Profil $\mathsf{Profile}(\mathcal{K}_j)$ einer Wissensbasis $\mathcal{K}_j$).

$$\begin{aligned} \mathit{Profile}(\mathcal{K}_j) \; = & \{\; \mathit{msc}_i^{\mathcal{K}_j} \mid i \in \mathit{Ind}(\mathcal{K}_j)\} \cup \\ & \{\; \mathit{msr}_{(i_1,i_2)}^{\mathcal{K}_j} \mid i_1, i_2 \in \mathit{Ind}(\mathcal{K}_j)\} \end{aligned}$$

Intuitiv formuliert enthält $\mathsf{Profile}(\mathcal{K}_j)$ also die Terminologie, die in $\mathcal{K}_j$ verwendet wird, und davon jeweils nur die spezifischsten Konzepte und Rollen. Diese Definition des Profils ist durch die Annahme motiviert, dass unterschiedliche Teil-Wissensbasen unterschiedliche Domänen abdecken und damit unterschiedliche Untermengen von N_C und N_R für ABox-Zusicherungen verwendet werden: Auch wenn einer Teil-Wissensbasis $\mathcal{K}_j$ permanent neue Kontextinformationen in Form von Individuen hinzugefügt werden, erfordert dies in der Regel *keine* Aktualisierung des Profils, da die sich aus einem neuen Individuum i ergebenden $\mathsf{msc}_i^{\mathcal{K}_j}$ und $\mathsf{msr}_{(i,i')}^{\mathcal{K}_j}$ bereits in $\mathsf{Profile}(\mathcal{K}_j)$ enthalten sind.

Bei der Generierung eines Anfrageplans wird $\mathsf{Profile}(\mathcal{K}_j)$ nun wie folgt verwendet, um relevante Teil-Wissensbasen für Selektions-Operatoren zu identifizieren:

$$\begin{aligned} \mathcal{K}_j \text{ ist relevant für } \sigma_{C(t)} &\iff \exists D \in \mathsf{Profile}(\mathcal{K}_j) : \mathcal{T} \models D \sqsubseteq C \\ \mathcal{K}_j \text{ ist relevant für } \sigma_{r(t,t')} &\iff \exists s \in \mathsf{Profile}(\mathcal{K}_j) : \mathcal{T} \models s \sqsubseteq r \end{aligned}$$

Optimierung des Anfrageplans. Häufig kann ein auf diese Weise generierter Anfrageplan in einen logisch äquivalenten Anfrageplan transformiert werden, der wesentlich effizienter ausgeführt werden kann.

Da die Anfragebeantwortung im verteilten Reasoning-System als Grundlage eine Untermenge der relationalen Algebra verwendet, kann hierfür auf einer großen Zahl an Arbeiten in diesem Gebiet aufgesetzt werden (siehe z. B. [Cha98] für einen Überblick). Dazu wird beispielsweise die *Reihenfolge* der physischen Operatoren im Anfrageplan geändert. Prinzipiell gilt, dass Selektoren, die viele Kandidaten aussortieren und wenige

potentielle Variablenbindungen als Ergebnis haben, möglichst früh ausgeführt werden sollen.

Außerdem können Heuristiken entwickelt werden, die spezifisch für die Anfragebeantwortung im verteilten Reasoning-System sind: Selektionen, die die Realisierung oder Instanz-Prüfung für Individuen erfordern, sind aufwändiger auszuwerten als Selektionen, die Rollenzusicherungen prüfen. Auch sind Selektionen für verteilt beschriebene Individuen aufwändiger als Selektionen für nicht verteilt beschriebene Individuen, die zudem parallelisiert werden können.

Derartige Heuristiken werden also verwendet, um aus einem Anfrageplan für den gegebenen relationalen Operatorbaum einen effizient ausführbaren Anfrageplan als Ergebnis dieser Anfragebeantwortungs-Phase zu generieren.

5.3.4.3. Ausführung des Anfrageplans

In dieser Phase wird der zuvor generierte Anfrageplan ausgeführt, um die Lösungen der konjunktiven Anfrage zu ermitteln. Auch hier sind wieder unterschiedliche Ansätze denkbar [Gra93, GMUW01]. Die Verfahren im verteilten Reasoning-System machen dazu keine Vorgaben, sondern überlassen die konkrete Umsetzung der jeweiligen Implementierung. Abschnitt 5.4.4.1 stellt eine Realisierung mittels Iteratoren vor.

5.3.5. Bewertung der Konzepte zum verteilten Reasoning

Die hier vorgeschlagenen Verfahren zur Anfragebeantwortung in einem verteilten Reasoning-System werden im Folgenden anhand der Anforderungen aus Abschnitt 5.1 bewertet.

Funktionale Anforderungen.

A5.1.1 Berücksichtigung von verteilt vorliegenden Kontextinformationen zu demselben Überwachungsobjekt. Da die Verfahren zur Anfragebeantwortung lediglich ein verteiltes Reasoning-System $\mathfrak{R}$ annehmen, in dem jeder Reasoning-Knoten R_j, $1 \leq j \leq n$, eine Schnittstelle zu seinen Reasoning-Diensten zur Verfügung stellt, können sie sehr flexibel eingesetzt werden. So können für unterschiedliche Arten der Verteiltheit der zugrunde liegenden Wissensbasis unterschiedlich effiziente Verfahren zur Beantwortung von Anfragen angegeben werden. Im Rahmen dieser Arbeit wurden beliebig und eingeschränkt verteilte Wissensbasen untersucht, bei denen Kontextinformationen zu demselben Überwachungsobjekt in mehreren Wissensbasen hinterlegt sein können (Abschnitte 5.3.1 und 5.3.3).

A5.1.2 Verwendung einer einheitlichen Systemontologie. Im verteilten Reasoning-System wird davon ausgegangen, dass jeder Reasoning-Knoten $R_j, 1 \leq j \leq n$, über dieselbe TBox $\mathcal{T}$ der zugrunde liegenden verteilten Wissensbasis $\mathfrak{K}$ verfügt (Abschnitt 5.1.2). Für die unterschiedlichen Arten der Verteiltheit von $\mathfrak{K}$ wurden Verfahren entwickelt, mit denen Anfragen über $\mathfrak{K}$ bezüglich $\mathcal{T}$ beantwortet werden.

A5.1.3 Unterstützung für korrektes und vollständiges beschreibungslogisches Schlussfolgern. Für beliebig, partitioniert und eingeschränkt verteilte Wissensbasen wurden Verfahren entwickelt, mit denen beschreibungslogische Anfrageatome beantwortet werden können. Korrektheit und Vollständigkeit dieser Verfahren wurde jeweils nachgewiesen (Abschnitte 5.3.1, 5.3.2 und 5.3.3).

Besonders wichtig ist, dass die Korrektheit der Antworten auch beim Ausfall von Reasoning-Knoten gewährleistet ist: Der Ausfall eines Reasoning-Knotens führt dazu, dass seine Teil-Wissensbasis nicht mehr zur Verfügung steht, d. h. weniger ABox-Zusicherungen berücksichtigt werden. Aufgrund der *open world assumption* (siehe Abschnitt 3.2.1) wird aus dem *Fehlen* einer ABox-Zusicherung jedoch nicht darauf geschlossen, dass diese ABox-Zusicherung nicht gilt! Stattdessen wird über die Gültigkeit einer fehlenden ABox-Zusicherung keine Annahme getroffen. Dadurch wird verhindert, dass falsche Schlüsse gezogen werden. Beispielsweise wird bei der Nichtverfügbarkeit einer Teil-Wissensbasis mit Statusinformationen zum Bremssystem nicht gefolgert, dass eine Bremse defekt ist, nur weil die explizite Zusicherung fehlt, dass sie *nicht* defekt ist.

Die *open world assumption* hat auch zur Folge, dass auf Basis der verfügbaren ABox-Zusicherungen immer die bestmögliche, korrekte Ableitung getroffen wird: Wenn für einen Zug eine kritische Achslast gemessen wird, jedoch keine Informationen zur Achslagertemperatur verfügbar sind, kann gemäß Definition in Abschnitt 4.4.2 *nicht* geschlossen werden, dass es sich um einen PriorityBogieMaintenanceTrain handelt. Da jedoch bekannt ist, dass die Achslast kritisch ist, kann sehr wohl gefolgert werden, dass es sich um einen PriorityWheelsetMaintenanceTrain handelt.

A5.1.4 Beantwortung konjunktiver Anfragen. Aufbauend auf der Beantwortung von Anfrageatomen wurde für ein verteiltes Reasoning-System ein allgemeingültiges Verfahren entwickelt, um Konjunktionen von Anfrageatomen zu beantworten (Abschnitt 5.3.4). Dieses Verfahren sieht außerdem einen Optimierungsschritt vor, mit dem anwendungsspezifische Optimierungen integriert werden können, falls es möglich ist, zusätzliche Annahmen zu der Verteiltheit der Wissensbasis zu treffen.

Nicht-funktionale Anforderungen.

A5.2.1 Verteilte Haltung der Kontextinformationen ohne zentrale Instanz. Ein verteiltes Reasoning-System $\mathfrak{R}$ besteht aus gleichwertigen Reasoning-Knoten mit zugeordneten Teil-Wissensbasen und sieht keine zentrale Instanz vor (siehe Abschnitt 5.3). Zur effizienten Beantwortung von Anfrageatomen ist es bei partitioniert und eingeschränkt verteilten Wissensbasen jedoch nötig, potentiell relevante Teil-Wissensbasen zu identifizieren (Abschnitte 5.3.2 und 5.3.3) und damit möglichst viele Teil-Wissensbasen von vorne herein auszuschließen. Dazu wird ein Verzeichnisdienst benötigt. Dieser ist logisch gesehen zwar eine zentrale Komponente. Es gibt jedoch eine Vielzahl von Ansätzen, derartige Verzeichnisdienste vollständig dezentral umzusetzen. Beispiel dafür sind *strukturierte Peer-to-Peer-Netze* [MKL+03, RM06] auf Basis *verteilter Hash-Tabellen* (*Distributed Hash Tables*, DHT) [FBT07, GTPZ05, BBK02].

Kriterium	DIG	OWL-QL	nRQL	SPARQL	SPARQL-DL
Konjunktive Anfragen	nein	ja	ja	ja	ja
DL-Realisierung	ja	nein	ja	nein	ja
Reifegrad	+	−	+/−	+	+/−

Tabelle 5.3.: Vergleich von Anfragesprachen für beschreibungslogische Systeme.

A5.2.2 Unabhängigkeit von Reasoner-Verfahren und -Implementierungen. Die vorgeschlagenen Verfahren zur Anfragebeantwortung nehmen lediglich ein verteiltes Reasoning-System $\mathfrak{R}$ an. Dieses besteht aus Reasoning-Knoten $R_j, 1 \leq j \leq n$, welche alleine durch ihre Schnittstelle spezifiziert sind. Es wird kein Annahme getroffen, wie ein Reasoning-Knoten diese Schnittstelle implementiert (siehe Abschnitt 5.3). Deshalb sind die vorgeschlagenen Verfahren zum Reasoning über verteilte Kontextinformationen unabhängig von der verwendeten Reasoner-Implementierung.

Insgesamt zeigt sich also, dass der in dieser Arbeit vorgeschlagene Ansatz zum Reasoning über verteilte Kontextinformationen alle aufgestellten Anforderungen erfüllt.

5.4. Implementierung der Verfahren als Framework

Zur Umsetzung der verschiedenen Verfahren zur Anfragebeantwortung in einem verteilten Reasoning-System wurde ein Framework implementiert. Nach Erläuterung der Auswahlkriterien für Ontologie- und Anfragesprachen, stellt dieser Abschnitt die Komponenten des Frameworks und ihre Schnittstellen sowie einige Implementierungsdetails vor.

5.4.1. Auswahl der Ontologiesprache

Für die Wissensrepräsentation mit Ontologien wurden verschiedene Sprachen vorgeschlagen. Zu den ersten gehören das *Knowledge Interchange Format* (KIF) und die daran angelehnte *CycL*, die Sprache des Cyc-Projekts. Sie orientieren sich noch an der Prädikatenlogik erster Ordnung und können deshalb auch Aussagen repräsentieren, die nicht entscheidbar sind. Aufbauend auf den entscheidbaren Beschreibungslogiken wurden in amerikanischen und europäischen Forschungsprojekten die Sprachen DARPA *Agent Markup Language* (DAML) und *Ontology Inference Layer* (OIL) entwickelt. Diese bildeten schließlich die Grundlage für die Spezifikation der *Web Ontology Language* (OWL) des W3C (vergleiche Abschnitt 3.4.1). Da diese Arbeit auf den Standards des *Semantic Web* aufbauen soll und mit $\mathcal{SHIN}$ auch eine entsprechende Beschreibungslogik gewählt wurde, verwendet die Implementierung OWL DL als Ontologiesprache.

5.4.2. Auswahl der Anfragesprache

Für die Anfragesprache gibt es mehrere Kandidaten (siehe auch Tabelle 5.3). Bereits vor einigen Jahren wurde der Pseudo-Standard DIG der *Description Logic Implementation Group* entwickelt, der von allen wichtigen Reasoner-Implementierungen unterstützt

Anfrageatom	SPARQL-Syntax	Beschreibung
$r(a,b)$	`{ a r b. }`	Rollen-Prüfung
$r(t,b)$	`{ ?t r b. }`	Rollen-Abfrage
$r(a,t)$	`{ a r ?t. }`	Rollen-Abfrage
$r(t,t')$	`{ ?t r ?t2. }`	Rollen-Abfrage
$\mathsf{msr}^{\mathcal{K}}_{(a,b)}$	`{ a ?r b. }`	Rollen-Realisierung
$C(a)$	`{ a rdf:type C. }`	Instanz-Prüfung
$C(t)$	`{ ?t rdf:type C. }`	Instanz-Abfrage
$\mathsf{msc}^{\mathcal{K}}_{a}$	`{ a distrea:msc ?C. }`	Konzept-Realisierung

Tabelle 5.4.: Abbildung der beschreibungslogischen Anfrageatome auf die SPARQL-Syntax ($a, b \in N_I, t, t' \in N_V$).

wird. Seine Ausdrucksstärke ist jedoch sehr beschränkt, da DIG zunächst rein auf TBox-Reasoning ausgerichtet war. Deshalb wird zur Zeit die Erweiterung DIG *2.0* entwickelt. Die Zukunftsaussichten dieser Erweiterung sind jedoch unklar, da sich in der Zwischenzeit Alternativen etabliert haben.

Für OWL-QL ist die Situation ähnlich. Die Sprache wurde vom *Stanford Knowledge Systems Laboratory* als OWL-Variante der DAML *Query Language* (DQL) vorgeschlagen [FHH04]. Bisher existieren jedoch höchstens Teilimplementierungen.

nRQL ist die proprietäre Anfragesprache des RacerPro-Reasoners. Sie bietet sehr viele Möglichkeiten, die über reines OWL DL hinausgehen [HMW04]. Jedoch wird sie nur von RacerPro unterstützt.

SPARQL wurde vom W3C als Anfragesprache für das *Resource Description Framwork* (RDF) entwickelt, welches auch der üblichen OWL-Syntax zugrunde liegt. Die Spezifikation wurde über mehrere Jahre hinweg diskutiert und liegt seit Januar 2008 als W3C *Recommendation* vor [PS08]. Bereits vor Abschluss des Standardisierungsverfahrens hatte sich SPARQL als De-Facto-Standard für die Abfrage von *Semantic Web*-Inhalten etabliert und wird von allen wesentlichen Implementierungen unterstützt. Zwar ist SPARQL nur auf RDF ausgerichtet (und berücksichtigt nicht die Semantik von OWL). Jedoch wurde mit SPARQL-DL auch bereits eine Erweiterung von SPARQL für OWL DL vorgeschlagen, mit der beispielsweise auch die beschreibungslogische Konzept- und Rollen-Realisierung ausgedrückt werden können [SP07].

Insgesamt ist zu erwarten, dass sich analog zu den Ontologiesprachen im Laufe der Zeit auch bei den Anfragesprachen ein De-Facto-Standard etablieren wird. Als wesentlicher Bestandteil der *Semantic Web*-Initiative des W3C kristallisiert sich dafür immer klarer SPARQL heraus. Deshalb setzt die hier beschriebene Implementierung auf SPARQL und der beschreibungslogischen Interpretation SPARQL-DL auf. Um Rollen- und Konzept-Realisierung ausdrücken zu können, wird zusätzlich das Konstrukt (in SPARQL: *property*) distrea:msc eingeführt und eine Rollenvariable als Rollen-Realisierung interpretiert. Tabelle 5.4 zeigt die Abbildung der beschreibungslogischen Anfrageatome auf die SPARQL-Syntax.

```
PREFIX distrea:<http://www.distrea.org/core#>
PREFIX domain:<http://www.infra.org/domain#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CONSTRUCT {
   ?x domain:hasObservation ?obs.
   ?x rdf:type ?c
}
WHERE {
   ?x domain:hasObservation ?obs.
   ?x distrea:msc ?c
}
```

Listing 5.1: Beispiel für eine SPARQL-Anfrage.

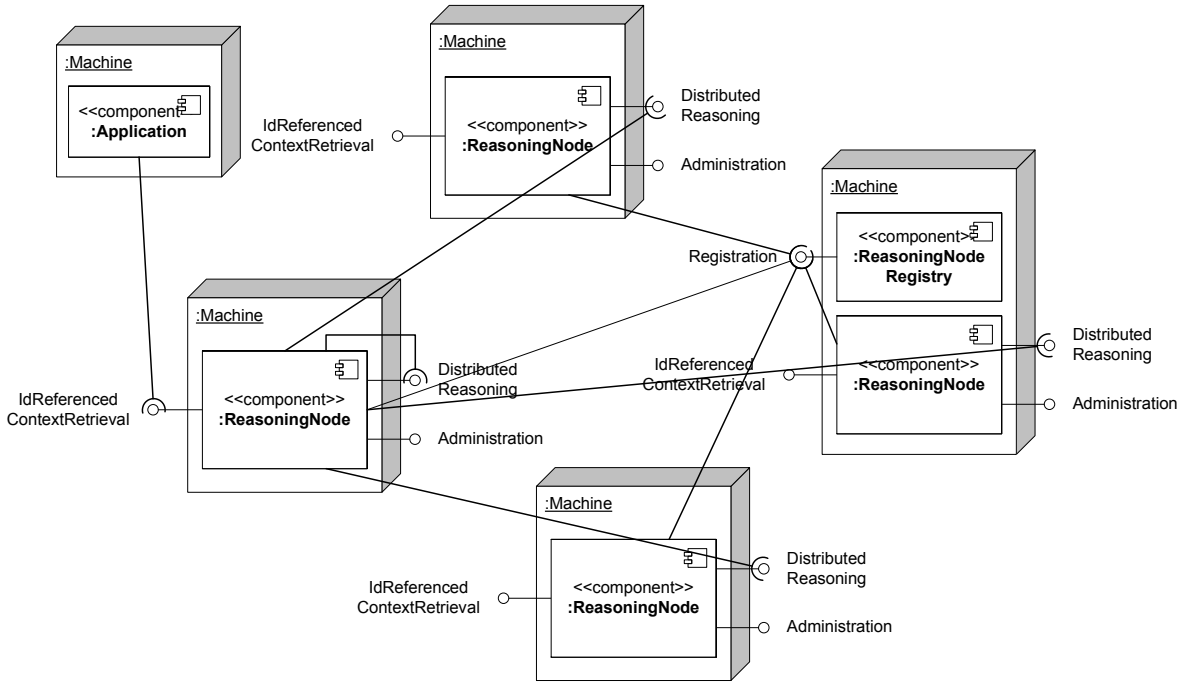

Abbildung 5.5.: Mögliche Verteilung der Komponenten zum Reasoning über verteilte Kontextinformationen.

Ein Beispiel für eine SPARQL-Anfrage mit den Schlüsselwort `CONSTRUCT` ist in Listing 5.1 dargestellt. Im Unterschied zu `SELECT`, dessen Ergebnis eine Menge von Lösungstupeln ist, antwortet `CONSTRUCT` mit einem RDF-Dokument und erlaubt es, die Struktur dieses Dokuments zu vorzugeben [PS08]. Bei entsprechender Spezifikation kann eine SPARQL-`CONSTRUCT`-Anfrage deshalb auch ein gültiges OWL-Dokument liefern, das direkt weiterverarbeitet werden kann.

5.4.3. Komponenten und Schnittstellen des Frameworks

Für die Implementierung wird das verteilte Reasoning-System als Framework umgesetzt. Dazu wird eine `ReasoningNode`-Komponente entwickelt, die einen Reasoning-Knoten

```
public SparqlPOut query( SparqlPIn query );
public SparqlPOut queryArbitrary( SparqlPIn query );
public SparqlPOut queryPartitioned( SparqlPIn query );
```

Listing 5.2: Schnittstelle `IdReferencedContextRetrieval`.

```
public SparqlPOut answerSparqlQuery( SparqlPIn query );
public SparqlPOut retrieveSparqlQuery( SparqlPIn query );
```

Listing 5.3: Schnittstelle `DistributedReasoning`.

realisiert und entsprechend mehrfach instantiiert wird. Zusätzlich wird eine `ReasoningNodeRegistry`-Komponente benötigt, die für ein Anfrageatom die relevanten Reasoning-Knoten bestimmt (vergleiche Abschnitt 5.3.4). Abbildung 5.5 illustriert in Anlehnung an UML-Verteilungsdiagramme eine mögliche Verteilung der Komponenten des Frameworks.

Die Komponenten des Frameworks sind durch ihre Schnittstellen spezifiziert. Es wird keine Annahme darüber getroffen, wie diese Schnittstellen implementiert sind.

Eine `ReasoningNode`-Komponente bietet folgende drei Schnittstellen an:

IdReferencedContextRetrieval Dies ist die für Anwendungen sichtbare Schnittstelle (siehe Listing 5.2). Sie besteht im Wesentlichen aus einer `query`-Operation, die als Eingabe eine SPARQL-`CONSTRUCT`-Anfrage erwartet und als Ausgabe ein entsprechendes RDF-Dokument liefert. Zur Kommunikation wird das vom W3C standardisierte SPARQL-*Protocol* eingesetzt, das ein einfaches Anfrage-Antwort-Muster mit den Nachrichtentypen `SparqlPIn` und `SparqlPOut` umsetzt [CFT08]. Die Operation `query` implementiert dabei das Verfahren für eingeschränkt verteilte Wissensbasen. Zusätzlich wurden die Operationen `queryArbitrary` und `queryPartitioned` spezifiziert, mit denen alternativ die Verfahren für beliebig bzw. partitioniert verteilte Wissensbasen nutzbar sind.

DistributedReasoning Diese Schnittstelle wird nur von `ReasoningNode`-Komponenten verwendet (siehe Listing 5.3). Sie bietet Zugriff auf Reasoning-Dienste für die lokale Teil-Wissensbasis, für den Abruf eines Wissensbasis-Kerns und für den Abruf der gesamten Teil-Wissensbasis. Die Operation `answerSparqlQuery` liefert die beschreibungslogisch implizierte Antwort auf die gestellte SPARQL-Anfrage, während `retrieveSparqlQuery` nur die explizit zugesicherten Antworten für die SPARQL-Anfrage liefert. Damit ist es beispielsweise möglich, bei beliebig verteilten Wissensbasen die vollständige ABox abzurufen. Außerdem können Wissensbasis-Kerne generiert werden.

Administration Diese Schnittstelle wird benutzt, um eine `ReasoningNode`-Komponente zur Laufzeit zu administrieren (siehe Listing 5.4). Sie umfasst Operationen zur Konfiguration der zu verwendenden `ReasoningNodeRegistry`-Komponente, des Inhalts der lokalen Teil-Wissensbasis und des verwendeten Protokolls zur Kommunikation zwischen den `ReasoningNode`-Komponenten.

Eine `ReasoningNodeRegistry`-Komponente bietet folgende Schnittstelle an:

```
public void setRegistry( URL registryEndpoint );
public URL getRegistry();

public void setKnowledgeBase( URI knowledgeBase );
public void addKnowledgeBase( URI knowledgeBase );
public URI[] getKnowledgeBaseUris();

public void setCommunicationProtocol( String protocol );
public String getCommunicationProtocol();
```

Listing 5.4: Schnittstelle `Administration`.

```
public void registerReasoningNode(URL endpoint, RDFDocument profile);
public void deregisterReasoningNode(URL endpoint);
public void reset();

public URL[] getRelevantReasoningNodes(URI subj, URI pred, URI obj);
public boolean isShared(URI indiv);
```

Listing 5.5: Schnittstelle `Registration`.

Registration `ReasoningNode`-Komponenten verwenden diese Schnittstelle, um sich mit ihrem Profil bei der `ReasoningNodeRegistry`-Komponente zu registrieren. Sie nutzen diese Schnittstelle außerdem, um für ein gegebenes Anfrageatom abzufragen, welche `ReasoningNode`-Komponenten (und damit Teil-Wissensbasen) für seine Beantwortung relevant sind. Schließlich können sie damit entscheiden, ob ein gegebenes Individuum verteilt oder nicht verteilt beschrieben ist.

Ein typischer Ablauf zur Beantwortung einer konjunktiven Anfrage ist in Abbildung 5.6 dargestellt: Eine Anwendung, die das verteilte Reasoning nutzen möchte, stellt ihre Anfrage über die `IdReferencedContextRetrieval`-Schnittstelle an eine beliebige `ReasoningNode`-Komponente. Diese `ReasoningNode`-Komponente übernimmt dann die Koordination der Anfragebeantwortung, indem sie einen Anfragebearbeitungsplan erstellt, der bei Bedarf andere `ReasoningNode`-Komponenten einbezieht. Dazu wird eine (logische) `ReasoningNodeRegistry`-Komponente benötigt, die alle `ReasoningNode`-Komponenten identifiziert, die für die Beantwortung einer Anfrage relevant sind.

5.4.4. Implementierung der Framework-Komponenten

Alle Komponenten sind als Webanwendung nach der *Java-Servlet*-Spezifikation und mit *Java 5* entwickelt. Sie können in beliebigen Servlet-Containern wie z. B. *Apache Tomcat*[1] oder *Jetty*[2] ausgeführt werden. Ihre Schnittstellen sind mit der *Web Service Description Language* (WSDL) spezifiziert und werden als W3C *Web Service* angeboten. Als Protokoll wird SOAP *über* HTTP verwendet.

[1] Apache Tomcat: http://tomcat.apache.org/
[2] Jetty: http://jetty.mortbay.com/

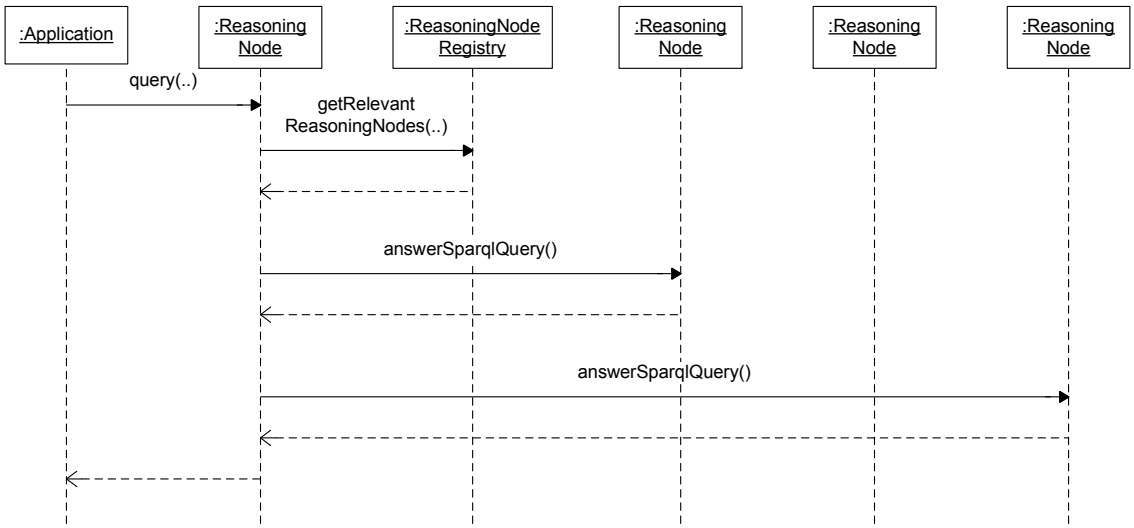

Abbildung 5.6.: Beispiel für Interaktionen zwischen den Komponenten des Frameworks.

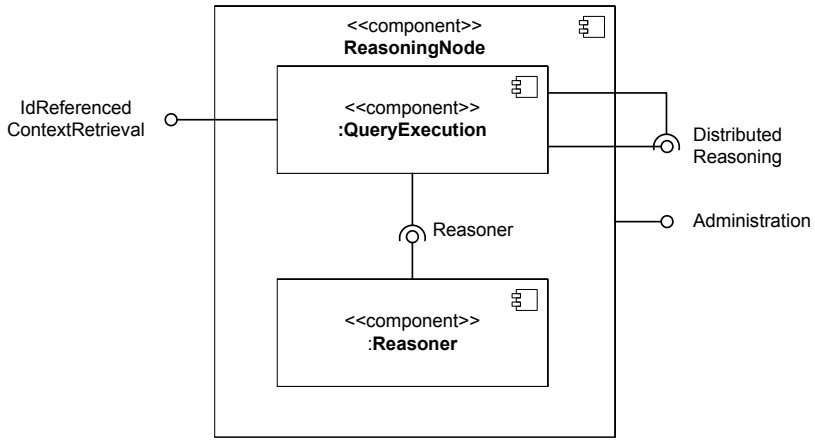

Abbildung 5.7.: Architektur der `ReasoningNode`-Komponente.

5.4.4.1. ReasoningNode-Komponente

Dieser Abschnitt stellt eine mögliche Umsetzung der `ReasoningNode`-Komponente und Implementierung der oben beschriebenen Schnittstellen vor. Abbildung 5.7 zeigt die zwei Sub-Komponenten der `ReasoningNode`-Komponente: die `QueryExecution`-Komponente zur Anfragebearbeitung und die `Reasoner`-Komponente, die die lokale Teil-Wissensbasis verwaltet und Reasoning-Dienste auf ihr anbietet.

QueryExecution-Komponente zur Anfragebearbeitung. Die `QueryExecution`-Komponente nimmt alle Anfragen an die `ReasoningNode`-Komponente entgegen. Anfragen sind konjunktive Anfragen, die wie in Abschnitt 5.4.2 beschrieben formuliert sind. Wird eine Anfrage über die `DistributedReasoning`-Schnittstelle gestellt, soll sie unter Verwendung der lokalen Reasoning-Dienste auf der Teil-Wissensbasis beantwortet

werden. Sie wird deshalb an die `Reasoner`-Komponente weitergeleitet. Wird die Anfrage über die `IdReferencedContextRetrieval`-Schnittstelle gestellt, soll sie bezüglich der Gesamt-Wissensbasis beantwortet werden. Dazu wird ein verteilter Anfragebearbeitungsplan generiert und ausgeführt wie in Abschnitt 5.3.4 beschrieben.

Ein verteilter Anfragebearbeitungsplan für eine konjunktive Anfrage legt für jedes Anfrageatom fest, welche `ReasoningNode`-Komponenten zur Beantwortung nötig sind. Dabei müssen Abhängigkeiten zwischen Anfrageatomen berücksichtigt werden, die gemeinsame Variablen haben und deren Teilergebnisse deshalb mittels der Join-Operation kombiniert werden müssen. Dabei ist die Beantwortungsreihenfolge der Anfrageatome ausschlaggebend für die Effizienz der Anfragebearbeitung.

Die Implementierung der Ausführung von Anfragebearbeitungsplänen wurde bereits ausführlich für relationale Datenbanken untersucht. Dabei hat sich das *Iterator-Modell* als besonders speichereffizient und flexibel herausgestellt [Gra93, GMUW01]. Es beruht auf dem Software-Entwurfsmuster des *Iterators* [GHJV95]. Dieses bietet eine *einheitliche* Schnittstelle für den sequentiellen Zugriff auf die Elemente einer aggregierten Struktur an, welche unabhängig von der internen Repräsentation dieser Struktur ist. Die Iterator-Schnittstelle umfasst die drei Methoden `open`, `next` und `close`: `open` initialisiert eine Iteration; `next` greift auf das aktuelle Element zu und schaltet anschließend zum nächsten Element weiter; `close` überprüft, ob die Iteration vollständig durchlaufen wurde. Jeder Operator eines Anfragebearbeitungsplans kann als Iterator implementiert werden. Die Funktionalität des Operators wird in der Methode `next` umgesetzt, also beispielsweise die Ermittlung eines Lösungselements für ein Anfrageatom. Da alle Iteratoren dieselbe einheitliche Schnittstelle besitzen, können sie in beliebiger Kombination verkettet werden. Dadurch können beliebig komplexe Anfragebearbeitungspläne ausgeführt werden.

Zur Implementierung der `QueryExecution`-Komponente ist es also nötig, aus einer SPARQL-Anfrage einen Anfragebearbeitungsplan zu generieren und daraus eine iterator-basierte Anfrageausführung zu instantiieren. Dafür setzt die hier vorgestellte Implementierung auf ARQ auf, der SPARQL-Komponente des *Jena Semantic Web Frameworks*[3] von *Hewlett-Packard*. ARQ umfasst einen SPARQL-Parser, der einen relationalen Operatorbaum generiert, und eine Ausführungskomponente, die daraus einen Anfrageplan generiert und den Anfrageplan mit dem Iterator-Modell ausführt. Mit ARQ können jedoch nur Anfragen auf einer lokalen Wissensbasis beantwortet werden.

Zur Realisierung der `QueryExecution`-Komponente wurde ARQ deshalb an mehreren Stellen wie folgt erweitert.

Generierung des Anfrageplans. Bei der Analyse jedes Anfrageatoms muss im Unterschied zu ARQ zusätzlich entschieden werden, welche `ReasoningNode`-Komponenten zu seiner Beantwortung relevant sind. Es werden vier Operatoren unterschieden:

- *Lokales Reasoning*: Das Anfrageatom kann mit der lokalen `Reasoner`-Komponente beantwortet werden. Dies entspricht der Ausgangs-Implementierung von ARQ.

[3] Jena Semantic Web Framework: http://jena.sourceforge.net/

- *Entferntes Reasoning*: Das Anfrageatom muss von einer entfernten `ReasoningNode`-Komponente beantwortet werden.

- *Mehrfaches entferntes Reasoning*: Das Anfrageatom muss von mehreren entfernten `ReasoningNode`-Komponenten beantwortet werden, was unabhängig voneinander geschehen kann.

- *Verteiltes Reasoning*: Weil sich das Anfrageatom auf ein verteilt beschriebenes Individuum bezieht, muss von mehreren entfernten `ReasoningNode`-Komponenten ein Wissensbasis-Kern generiert und das Anfrageatom auf Basis dieses Wissensbasis-Kerns beantwortet werden.

Optimierung des Anfrageplans. Aufbauend auf der in [SS07] vorgestellten Implementierung wurden Optimierungs-Heuristiken umgesetzt wie in Abschnitt 5.3.4.2 beschrieben. Diese berücksichtigen die neu hinzugekommenen Operatoren für entferntes und verteiltes Reasoning. Beispielsweise ist die Auswertung des Operators für verteiltes Reasoning wesentlich aufwändiger als die Auswertung des Operators für entferntes Reasoning. Deshalb sorgt eine Heuristik dafür, dass Operatoren für verteiltes Reasoning möglichst spät ausgeführt werden, damit die zu bearbeitende Kandidatenmenge möglichst klein ist.

Ausführung des Anfrageplans. Auch hier musste das bestehende iterator-basierte Ausführungs-Modell von ARQ erweitert werden. Entsprechend der neu hinzugefügten Operatoren wurden die folgenden zusätzlichen Iteratoren implementiert:

- *Lokales Reasoning*: Das Anfrageatom wird als Anfrage an die lokale `Reasoner`-Komponente weitergeleitet.

- *Entferntes Reasoning*: Auf Basis des Anfrageatoms wird eine Anfrage an die `DistributedReasoning`-Schnittstelle der entfernten `ReasoningNode`-Komponente generiert.

- *Mehrfaches entferntes Reasoning*: Auf Basis des Anfrageatoms wird eine Anfrage generiert und parallel an die `DistributedReasoning`-Schnittstellen der entfernten `ReasoningNode`-Komponenten gesendet. Die unterschiedlichen Ergebnismengen werden vereinigt und Duplikate entfernt.

- *Verteiltes Reasoning*: Gemäß des in Abschnitt 5.3.3.3 vorgestellten Verfahrens wird über die `DistributedReasoning`-Schnittstellen der relevanten `ReasoningNode`-Komponenten der Wissensbasis-Kern generiert und das Anfrageatom mit der lokalen `Reasoner`-Komponente bezüglich dieses Wissensbasis-Kerns beantwortet.

Auf diese Weise kann das vorgeschlagene Framework beliebige, in SPARQL formulierte konjunktive Anfragen beantworten.

```
public RDFDocument answerSparqlQuery(Sparql query);
public RDFDocument retrieveSparqlQuery(Sparql query);

public void setKnowledgeBase(URI kbUri);
public void setKnowledgeBase(RDFDocument kb);
public void addKnowledgeBase(URI kbUri);
public RDFDocument getKnowledgeBase();
```

Listing 5.6: Schnittstelle `Reasoner`.

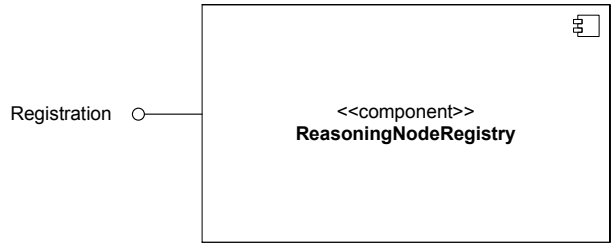

Abbildung 5.8.: Architektur der `ReasoningNodeRegistry`-Komponente.

Reasoner-Komponente. Die `Reasoner`-Komponente verwaltet die lokale Teil-Wissensbasis. Auf sie wird zugegriffen, wenn lokales Reasoning durchgeführt werden muss. Dazu bietet sie die Schnittstelle `Reasoner` an, über die beschreibungslogische Anfrageatome bezüglich der Teil-Wissensbasis beantwortet werden können (siehe Listing 5.6). Auch hier wird wieder unterschieden zwischen `answerSparqlQuery`, womit Anfrageatome beschreibungslogisch beantwortet werden, und `retrieveSparqlQuery`, womit lediglich explizite Zusicherungen in der Teil-Wissensbasis abgefragt werden. Zusätzlich gibt es Methoden zur Verwaltung der lokalen Teil-Wissensbasis.

Diese Schnittstelle kann mit Hilfe bestehender Reasoner-Implementierungen zum zentralen Reasoning (vergleiche Abschnitt 5.2.1) realisiert werden. Bei der hier vorgestellten Implementierung wurden sowohl RacerPro als auch Pellet als `Reasoner`-Komponenten eingesetzt und aus Effizienzgründen jeweils über ihre proprietären Schnittstellen angesprochen. Es wurde RacerPro in der Version 1.9.0 und Pellet in der Version 1.5-RC verwendet.

5.4.4.2. ReasoningNodeRegistry-Komponente

Die `ReasoningNodeRegistry`-Komponente (siehe Abbildung 5.8) wird bei der Generierung des Operatorbaums benötigt, um zu entscheiden, ob ein Anfrageatom lokal, entfernt oder verteilt beantwortet werden muss. Sie bietet die Schnittstelle `Registration` an (siehe Listing 5.5), über die sich `ReasoningNode`-Komponenten mit ihrem Profil registrieren und für ein Anfrageatom abfragen können, welche `ReasoningNode`-Komponenten für die Beantwortung relevant sind. Dafür wird das in Abschnitt 5.3.4.2 vorgestellte Verfahren eingesetzt. Außerdem kann herausgefunden werden, ob ein Individuum ein verteilt beschriebenes Individuum ist und deshalb verteiltes Reasoning nötig ist.

Dabei ist wichtig, dass die `ReasoningNodeRegistry`-Komponente lediglich in der

hier vorgestellten Implementierung eine zentrale Komponente darstellt und damit die Dezentralität des Frameworks einschränkt. Sie kann jedoch auch problemlos dezentral realisiert werden (vergleiche Abschnitt 5.3.5).

5.5. Experimentelle Evaluierung

Nachdem der vorgestellte Ansatz für Reasoning über verteilte Kontextinformationen auf Basis eines verteilten Reasoning-Systems in Abschnitt 5.3.5 bereits theoretisch bewertet wurde, werden im Folgenden noch experimentelle Evaluierungsergebnisse auf Basis der in Abschnitt 5.4 vorgestellten Implementierung als Framework vorgestellt. Dies dient der besseren Beurteilung des praktischen Verhaltens der entwickelten Verfahren.

5.5.1. Ziele der Evaluierung

Es wurde bereits theoretisch nachgewiesen, dass die vorgestellten Verfahren vollständige und korrekte Lösungen liefern. Auch wurde gezeigt, dass selbst bei Ausfall einzelner Teil-Wissensbasen auf Grundlage des vorhandenen Wissens vollständige und – aufgrund der *open world assumption* von Beschreibungslogiken – vor allem korrekte Antworten ermittelt werden (vergleiche Abschnitt 5.3.5).

Durch die experimentelle Evaluierung soll nun der Ressourcenbedarf für Reasoning über verteilte Kontextinformationen untersucht und mit dem alternativen Ansatz der Zentralisierung der verteilten Wissensbasis verglichen werden.

Aufschlussreich ist vor allem die Veränderung des Ressourcenbedarfs entlang unterschiedlicher Dimensionen. Die zwei wichtigsten Kennzahlen bei einer verteilten Wissensbasis sind die *Größe* der Wissensbasis und die *Anzahl* der Teil-Wissensbasen. Deshalb sollen mit der experimentellen Evaluierung folgende Zusammenhänge untersucht werden:

Veränderung des Ressourcenbedarfs bei steigender Größe der Wissensbasis. Die unterschiedlichen Kontextquellen liefern fortlaufend neue Kontextinformationen, die als Individuen in den Teil-Wissensbasen repräsentiert werden. Zwar muss der Inhalt der Wissensbasen verwaltet, also z. B. veraltete Informationen entfernt werden. Dennoch müssen große Datenmengen bewältigt werden. Für die experimentelle Evaluierung ist relevant, wie sich die Größe der Wissensbasis auf die Antwortzeit und den Speicherbedarf des Reasonings auswirkt.

Veränderung des Ressourcenbedarfs bei steigender Zahl der Reasoning-Knoten. Eine Hauptmotivation für das Reasoning über verteilte Kontextinformationen ist die Integration unterschiedlicher, verteilt vorliegender Kontextquellen. Die Hinzunahme neuer Kontextquellen vergrößert in der Regel die Zahl der Teil-Wissensbasen. Für die experimentelle Evaluierung ist deshalb auch relevant, wie sich die Zahl der Teil-Wissensbasen auf die Antwortzeit und den Speicherbedarf des Reasonings auswirkt.

Im Folgenden werden geeignete Test-Wissensbasen und -Anfragen zur Evaluierung dieser beiden Aspekte ausgewählt.

5.5.2. Test-Wissensbasen und -Anfragen

Die Entwicklung von Benchmark-Ontologien ist schon seit mehreren Jahren ein Thema für den Vergleich unterschiedlicher Reasoner-Implementierungen. Einer der ersten Benchmarks für OWL-Beschreibungslogik-Reasoner war der *LeHigh University Benchmark* (LUBM) [GPH05]. Vor allem aus Mangel an Alternativen wurde er von vielen Reasoner-Implementierern übernommen und hat deshalb den Vorteil, dass für ihn mehrere Vergleichsdaten verfügbar sind. Andererseits wurde häufig kritisiert, dass LUBM nur Teile der Ausdrucksmöglichkeiten von OWL DL nutzt. Darum wurde der *University Ontology Benchmark* (UOBM) als umfassendere Variante des LUBM vorgeschlagen [MYQ+06]. Andere häufiger verwendete Benchmarks sind außerdem der *Semintec Benchmark* [MS06b], die *Gene Ontology*[4] und die *Airport Codes Ontology*[5] (siehe [GHT06] für eine umfangreiche Auflistung).

Aktuelle Arbeiten zum Leistungsvergleich von Reasoner-Implementierungen zeigen jedoch, dass sich all diese Benchmarks nur bedingt auf realistische Einsatzszenarien übertragen lassen, da sie häufig von idealisierten Voraussetzungen ausgehen. Gleichzeitig wird festgestellt, dass ein realistischer Benchmark zur Zeit jedoch auch nicht existiert [WLL+07, WLLB06].

Für den Zweck dieser Arbeit stellt sich zusätzlich folgendes Problem: Alle genannten Benchmarks wurden nur für Reasoning mit zentralisierter Wissensbasis entworfen. Für die Evaluierung von Reasoning mit verteilten Kontextinformationen müssen die Benchmark-Wissensbasen zunächst künstlich verteilt werden. Deshalb werden zur Evaluierung nachfolgend zwei Ansätze parallel verfolgt:

- Einerseits wird eine Evaluierung auf Basis von LUBM durchgeführt, da für LUBM die meisten Vergleichsergebnisse zur Verfügung stehen. Dazu muss die LUBM-Wissensbasis künstlich in Teil-Wissensbasen aufgeteilt werden.

- Andererseits wird mit RailBM ein eigener Benchmark entwickelt, der die Domäne der Schienennetzüberwachung realistisch repräsentiert und bei dem die Wissensbasis inhärent verteilt ist.

5.5.2.1. LeHigh University Benchmark (LUBM)

Der *LeHigh University Benchmark* (LUBM) besteht aus einer Ontologie, die die Universitäts-Domäne modelliert, 14 Test-Anfragen und einem Testdaten-Generator [GPH05].

Die Ontologie umfasst 43 Konzepte und 32 Rollen. Sie lässt sich mit OWL lite (siehe Abschnitt 3.4.1) ausdrücken und verwendet z. B. inverse und transitive Rollen sowie Konjunktionen von Konzepten.

Bei der Wahl der Test-Anfragen wurde sowohl auf Realismus als auch Repräsentativität geachtet. Die vollständige Liste der Anfragen findet sich in Anhang A. Beispielhaft werden hier Anfragen 2 und 6 herausgegriffen (wobei das `rdf`-Präfix dem üblichen RDF-Namespace entspricht):

```
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x, ?y, ?z
```

[4] Gene Ontology: http://archive.godatabase.org/

[5] Airport Codes Ontology: http://www.daml.ri.cmu.edu/ont/AirportCodes.daml

```
WHERE { ?x rdf:type ub:GraduateStudent.
        ?y rdf:type ub:University.
        ?z rdf:type ub:Department.
        ?x ub:memberOf ?z.
        ?z ub:subOrganizationOf ?y.
        ?x ub:undergraduateDegreeFrom ?y.
}
```

Listing 5.7: LUBM-Anfrage 2.

```
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:Student. }
```

Listing 5.8: LUBM-Anfrage 6.

Anfrage 2 umfasst sechs Anfrageatome mit gegenseitigen Abhängigkeiten. Dabei werden drei Konzepte und drei Rollen verwendet. Im Gegensatz dazu besteht Anfrage 6 aus lediglich einem Anfrageatom. Zu seiner Beantwortung sind jedoch Inferenzen nötig, da UndergraduateStudent ein explizites und GraduateStudent ein implizites Unterkonzept von Student sind. Zudem ist die Zahl der potentiellen Lösungen sehr groß und die Selektivität der Anfrage sehr gering.

Der Testdaten-Generator ermöglicht es, unterschiedlich große ABoxen für Wissensbasen mit der LUBM-Ontologie als TBox zu erzeugen. Die Zahl der zu generierenden Universitäten kann vorgegeben werden, wobei jede Universität zwischen 15 und 25 Departments enthält und mit realistischen Zahlen für Professoren, Studenten usw. erzeugt wird. Jede Universität wird dann mit etwa 140.000 RDF-Tripeln beschrieben. Für diese Arbeit mussten mehrere verteilte Wissensbasen erzeugt werden, wobei jede Teil-Wissensbasis etwa gleich groß ist. Die Test-Wissensbasen sind in Tabelle 5.5 aufgelistet.

5.5.2.2. Benchmark für Zustandsüberwachung im Schienennetz (RailBM)

Da für das Reasoning über eine verteilte Wissensbasis kein Benchmark existiert, wird ein entsprechender Benchmark RailBM hier selbst entwickelt. Als Ausgangsbasis dient eine realistische Ontologie, die im Rahmen dieser Arbeit für die Zustandsüberwachung bei Schienennetzen erstellt wurde (vergleiche Abschnitt 7.3).

Die RailBM-Ontologie modelliert Kontextinformationen und Infrastrukturzustände für Achslastgeber, Heißläuferdetektoren, Tür- und Zugsystem-Überwachungssysteme. Sie wurde auf einen Kern von 31 Konzepten und 12 Rollen reduziert. Im Gegensatz zu

Name	Zwei Teil-WB	Drei Teil-WB	Vier Teil-WB
n1	15248	21610	28117
n2	29262	41419	55409
n3	42766	62072	83295
n4	57085	81859	108932
n5	71321	104161	137476

Tabelle 5.5.: Name und Größe der Test-Wissensbasen (WB) auf Basis von LUBM in Anzahl der Tripel.

LUBM nutzt RailBM die Ausdrucksstärke von $\mathcal{SHIN}$ bzw. $\mathcal{SHIF}$ komplett aus, was vollständigem OWL DL ohne Nominale entspricht.

Mit Blick auf das Reasoning über verteilte Daten wurden acht Test-Anfragen gewählt, die jeweils ein anderes Reasoning-Problem abdecken. Im Detail sehen sie wie folgt aus (wobei das `rdf`-Präfix dem üblichen RDF-Namespace entspricht):

```
PREFIX core:<http://www.infra.org/core#>
SELECT *
WHERE { ?status core:isStatusOf Train001. }
```

Listing 5.9: RailBM-Anfrage 1.

Anfrage 1 fragt eine explizit zugesicherte Rolle ab.

```
PREFIX core:<http://www.infra.org/core#>
SELECT *
WHERE { Train001 core:hasObservation ?obs.
   ?obs core:refersToSymptom ?symp. }
```

Listing 5.10: RailBM-Anfrage 2.

Anfrage 2 fragt eine Kette explizit zugesicherter Rollen ab.

```
PREFIX core:<http://www.infra.org/core#>
SELECT *
WHERE { ?x core:hasCondition ?y. }
```

Listing 5.11: RailBM-Anfrage 3.

Anfrage 3 fragt eine implizit zugesicherte Rolle ab und nutzt dazu die Rollenhierarchie aus.

```
PREFIX core:<http://www.infra.org/core#>
SELECT *
WHERE { Train001 core:hasStatus ?status. }
```

Listing 5.12: RailBM-Anfrage 4.

Anfrage 4 fragt ebenfalls eine implizit zugesicherte Rolle ab, die als invers definiert ist.

```
PREFIX wilm:<http://www.infra.org/ont/wilm#>
SELECT *
WHERE { Status001 rdf:type wilm:CriticalWILMStatus. }
```

Listing 5.13: RailBM-Anfrage 5.

Anfrage 5 führt eine entfernte Instanz-Prüfung durch.

```
PREFIX cond:<http://www.infra.org/ont/integrated#>
SELECT *
WHERE { Train001 rdf:type cond:CriticalTrain. }
```

Listing 5.14: RailBM-Anfrage 6.

Anfrage 6 führt eine verteilte Instanz-Prüfung durch.

```
SELECT *
WHERE { Status001 rdf:type ?concept. }
```

Listing 5.15: RailBM-Anfrage 7.

Anfrage 7 führt eine entfernte Konzept-Realisierung durch.

```
SELECT *
WHERE { Train001 rdf:type ?concept. }
```

Listing 5.16: RailBM-Anfrage 8.

Anfrage 8 führt eine verteilte Konzept-Realisierung durch.

Um auch für diese Benchmark-Ontologie Test-Wissensbasen unterschiedlicher Größen zur Verfügung zu haben, wurde ebenfalls ein Testdaten-Generator implementiert. Er erzeugt Daten auf Basis von Zügen und ordnet diesen Kontextinformationen von Heißläuferortungsanlagen, Achslastgebern, Waggontür- und Bereitschaftsstatus-Überwachungssystemen zu (vergleiche die Erläuterungen zum Schienennetz in Abschnitt 2.1). Dabei setzt er das Entwurfsmuster für Infrastrukturzustände aus Abschnitt 4.4 um und erlaubt so, aus Merkmalen – unter Verwendung der in der Ontologie formalisierten Semantik – automatisiert Zustände abzuleiten.

Für die experimentelle Evaluierung wurden sechs verteilte Wissensbasen mit steigender Größe für einen bis 250 Züge generiert. Auch hier sind die Teil-Wissensbasen etwa gleich groß. Als Datenbasis eines Benchmarks für verteiltes Reasoning sind die verteilten Wissensbasen unabhängig von einem bestimmten Szenario. Ein interessanter Vergleichswert ist jedoch, dass die ICE-Flotte der Deutschen Bahn AG im Jahr 2006 etwa 200 Züge umfasste [DB06]. Die Test-Wissensbasen sind in Tabelle 5.6 aufgelistet. Sowohl die Ontologie als auch die verwendeten Wissensbasen können unter `http://www.mobile.ifi.lmu.de/~fuchs/DistRea` heruntergeladen werden.

5.5.3. Evaluierungsergebnisse

Die Benchmark-Experimente wurden in einem lokalen Netz auf dedizierten Rechnern durchgeführt, um kontrollierte Bedingungen zu gewährleisten. Abbildung 5.9 zeigt den Aufbau der Testumgebung. Bei den Rechnern handelt es sich um einfache Desktop-Rechner, die mit einer *Pentium 4* CPU mit 3.2 GHz und 1 GB physischem Arbeitsspeicher ausgestattet sind. Davon standen den Reasoning-Knoten 768 MB zur Verfügung. Als Betriebssystem wird *Microsoft Windows XP Service Pack 2* verwendet. Die Reasoning-Knoten werden als Web-Anwendungen im Applikationsserver *Apache Tomcat 5.5* ausgeführt. Die Rechner sind zusammen mit anderen Rechnern in einem lokalen Netz mit 100 MBit/s vernetzt.

5.5.3.1. LeHigh University Benchmark

Zunächst werden die Ergebnisse für den *LeHigh University Benchmark* vorgestellt. Diese müssen jedoch vor folgendem Hintergrund interpretiert werden und eignen sich deshalb nur eingeschränkt zur Bewertung der vorgeschlagenen Verfahren zur Anfragebeantwortung:

Name	Züge	Zwei Teil-WB	Drei Teil-WB	Vier Teil-WB
n1	1	24	36	48
n50	50	1004	1506	2008
n100	100	2004	3006	4008
n150	150	3004	4506	6008
n200	200	4004	6006	8008
n250	250	5004	7506	10008

Tabelle 5.6.: Name und Größe der Test-Wissensbasen (WB) für RailBM in Anzahl der Tripel.

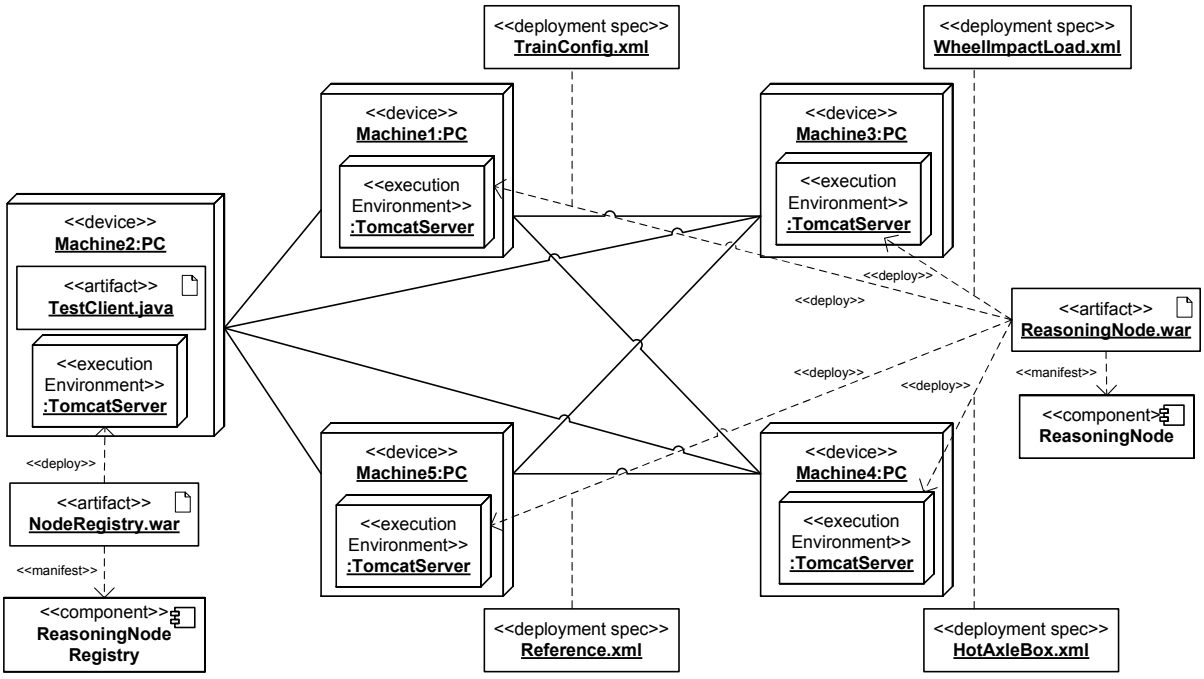

Abbildung 5.9.: Testaufbau für die experimentelle Evaluierung.

- Der Benchmark beruht auf einer zentralen Wissensbasis, die erst künstlich verteilt werden muss.
- Auch nach dieser Aufteilung stellt die Wissensbasis keine typische Wissensbasis dar, da die ABox-Zusicherungen in jeder Teil-Wissensbasis die gesamte TBox nutzen. Beim Reasoning über verteilte Kontextinformationen wird hingegen davon ausgegangen, dass die TBox mehrere Domänen beschreibt und sich jede Teil-Wissensbasis jeweils nur auf eine dieser Domänen bezieht. Dies wird bei der Beantwortung konjunktiver Anfragen ausgenutzt (vergleiche Abschnitt 5.3.4.2). Da diese Annahme bei LUBM nicht erfüllt ist, ist das vorgestellte Verfahren bei der Beantwortung von Anfragen mit vielen Anfrageatomen unnötig ineffizient. Dies betrifft vor allem die Anfragen 2, 8, 9, 12 und 13. Diese werden im Folgenden deshalb nicht näher betrachtet, die vollständigen Evaluierungsergebnisse befinden sich in Anhang A.
- Schließlich ist zu erwarten, dass für LUBM auch beim beschreibungslogischen Reasoning (also dem Beantworten der einzelnen Anfrageatome) das Ergebnis des Vergleichs von verteiltem und zentralisiertem Reasoning nur bedingt repräsentativ ist. Dies liegt daran, dass die generierten Wissensbasen des LUBM bereits künstlich partitioniert sind (vergleiche Abschnitt 5.5.2.1). Das wird von modernen Reasoner-Implementierungen ausgenutzt und führt deshalb bereits beim zentralisierten Reasoning zu hoher Effizienz, weshalb der Effizienzgewinn im verteilten Fall geringer ausfällt.

Dennoch stellt LUBM zur Zeit die etablierteste Methode zur Evaluierung von beschreibungslogischem Reasoning dar, weshalb im Folgenden die entsprechenden Evaluierungsergebnisse vorgestellt werden.

Test 1: Antwortzeit in Abhängigkeit der Größe der Wissensbasis. Für jede LUBM-Test-Wissensbasis mit zwei Teil-Wissensbasen (siehe Tabelle 5.5) wird folgendes Experiment durchgeführt: Die Reasoning-Knoten auf Rechner 3 und 4 werden mit den Teil-Wissensbasen initialisiert. Dann stellt die Test-Anwendung auf Knoten 2 die 14 LUBM-Anfragen (siehe Anhang A) nacheinander an den Reasoning-Knoten auf Rechner 1 und den Reasoning-Knoten auf Rechner 5. Rechner 1 wendet das Verfahren für Reasoning auf einer partitioniert verteilten Wissensbasis an, während Rechner 5 zentralisiertes Reasoning durchführt, d. h. die Wissensbasis aggregiert und auf der aggregierten Wissensbasis die Anfrage beantwortet. Für jede Anfrage wird die Antwortzeit gemessen, also der Zeitraum vom Absenden der Anfrage bis zum vollständigen Erhalt der Antworten auf Rechner 2. Um Schwankungen auszugleichen, die in einem verteilten System entstehen, wird dieser Ablauf dreimal wiederholt und anschließend das arithmetische Mittel gebildet.

Wie bereits theoretisch nachgewiesen, werden in beiden Fällen vollständige und korrekte Antworten geliefert. Abbildung 5.10 stellt die Antwortzeiten der betrachteten LUBM-Anfragen dar. Sie sind nach Anfragen gruppiert, um die Entwicklung der Antwortzeit mit zunehmender Größe der Wissensbasis verfolgen zu können. Generell zeigt sich, dass die Antwortzeiten für den verteilten Ansatz immer geringer sind als für den zentralisierten Ansatz. Dies ist noch klarer zu sehen in Abbildung 5.11

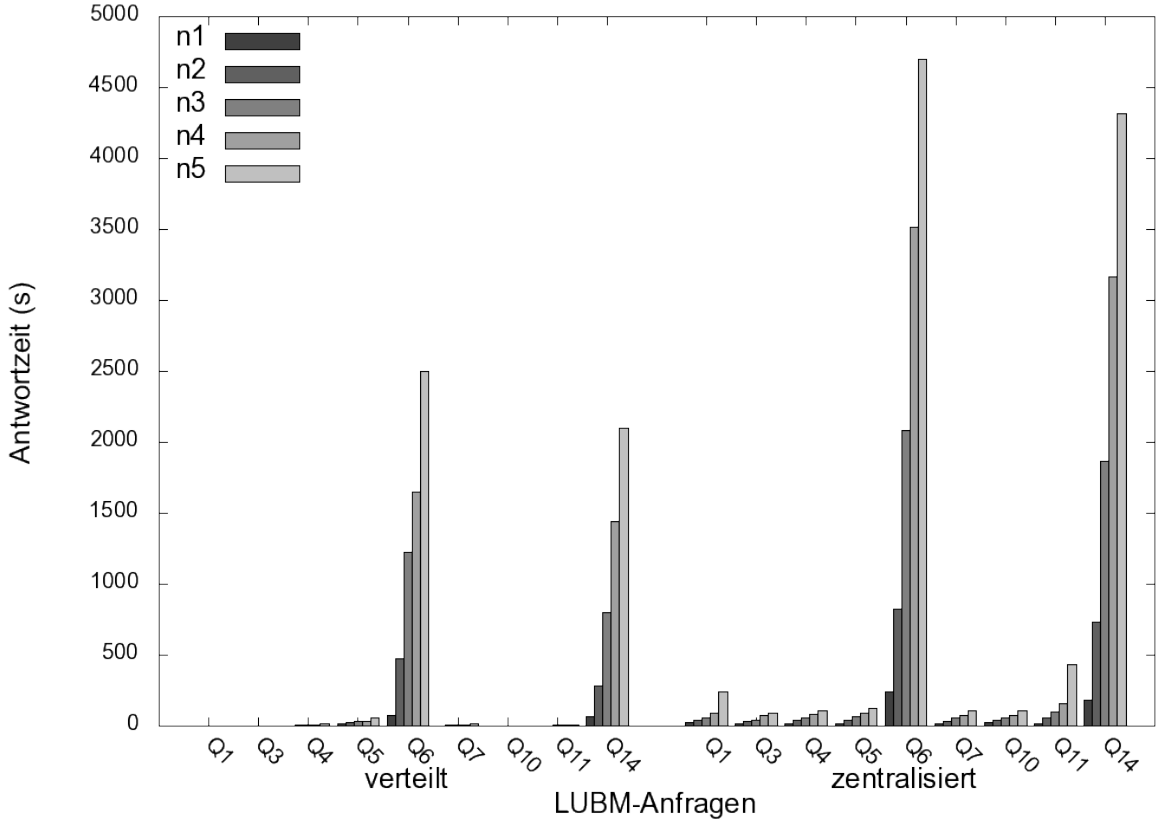

Abbildung 5.10.: Gegenüberstellung der Antwortzeiten für die LUBM-Anfragen mit steigender Größe der Wissensbasen im verteilten (links) und zentralisierten Fall (rechts).

(logarithmische Skala), wo die Antwortzeiten nach der Größe der Wissensbasis gruppiert und der verteilte dem zentralisierten Ansatz direkt gegenübergestellt ist.

Dabei lassen sich grob zwei Klassen von Anfragen unterscheiden: Zur Klasse eins gehören die Anfragen 1, 3, 4, 7, 10 und 11. Hier ist der Effizienzgewinn besonders groß, und die Antwortzeit im verteilten Fall beträgt durchschnittlich lediglich 7% der Antwortzeit des zentralisierten Ansatzes. Dies lässt sich dadurch erklären, dass die Anfrageatome dieser Anfragen eine hohe Selektivität besitzen, d. h. für ihre Lösungen gibt es jeweils nur wenige Kandidaten, die zudem nur auf einem kleinen Teil der Gesamt-Wissensbasis ausgewertet werden müssen. Dadurch können die Ergebnisse im verteilten Fall schnell gefunden werden.

Zur Klasse zwei gehören Anfragen 5, 6 und 14. Hier beträgt die Antwortzeit im verteilten Fall durchschnittlich 55% der Zeit im zentralisierten Ansatzes. Dies lässt sich dadurch begründen, dass diese Anfragen zwar wenige Anfrageatome, aber auch eine sehr geringe Selektivität besitzen. Dadurch werden sowohl im verteilten als auch im zentralisierten Fall große Lösungsmengen ermittelt. Der Effizienzgewinn ergibt sich dadurch, dass im verteilten Fall jeweils nur ein Teil der Gesamt-Wissensbasis berücksichtigt werden muss.

In Abbildung 5.11 zeigt sich außerdem, dass die Ersparnis mit steigender Größe der Wissensbasis vor allem für Anfragen der Klasse eins zunimmt. Für Anfragen der Klasse

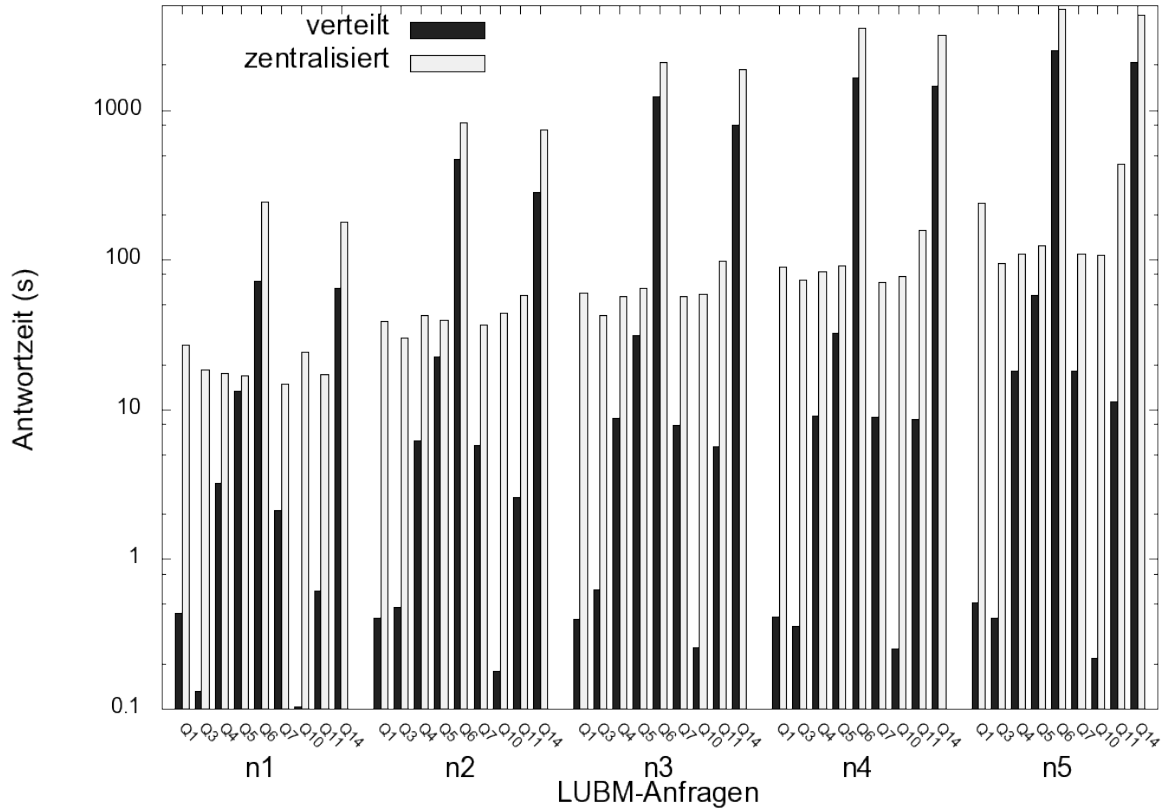

Abbildung 5.11.: Direkte Gegenüberstellung der Antwortzeiten für die LUBM-Anfragen mit steigender Größe der Wissensbasen.

zwei bleibt sie etwa konstant.

Aufgeschlüsselt nach Reasoning-Knoten sind die Antwortzeiten in Abbildung 5.12 dargestellt. Hier zeigt sich, dass Rechner 3 und 4 in etwa zu gleichen Teilen an der Beantwortung beteiligt sind, was sich durch die gleichmäßige Partitionierung der LUBM-Wissensbasen erklären lässt.

Test 2: Speicherbedarf in Abhängigkeit der Größe der Wissensbasis. Bei der Durchführung des oben beschriebenen Experiments wurde außerdem nach der Beantwortung jeder Anfrage der Speicherbedarf gemessen. Die Ergebnisse sind in Abbildung 5.13 dargestellt. Es zeigt sich, dass der Speicherbedarf bei Anfragen der Klasse eins sowohl für den verteilten als auch den zentralisierten Fall 200 MB selbst für die größte Test-Wissensbasis nicht übersteigt. Bei Anfragen der Klasse zwei steigt der Speicherbedarf jedoch sowohl im verteilten als auch im zentralisierten Ansatz mit der Größe der Wissensbasis stark an. Dies lässt sich mit der entsprechend stark steigenden Zahl von Lösungen für diese Anfragen und der Notwendigkeit zur Inferenz erklären.

Vergleicht man den verteilten mit dem zentralisierten Ansatz, liegt der Speicherbedarf auf den Rechnern 3 und 4 jeweils nur unwesentlich unter dem Speicherbedarf auf Rechner 5, in der Summe sogar darüber. Dies gilt auch für die Veränderung mit zunehmender Größe der Wissensbasis. Hier zahlt sich also bereits im zentralisierten Fall aus, dass die generierten LUBM-Test-Wissensbasen künstlich partitioniert sind.

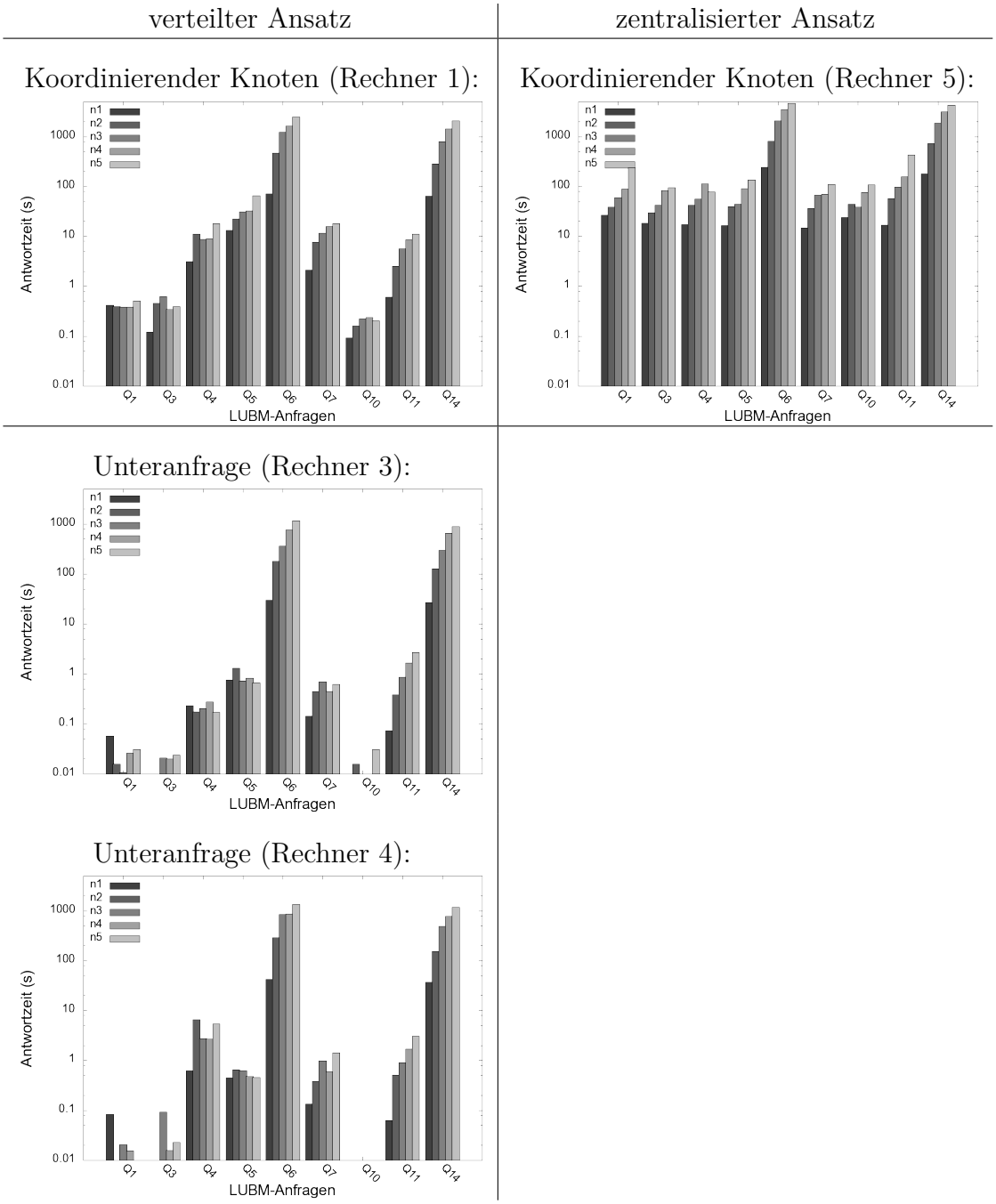

Abbildung 5.12.: Gegenüberstellung der Antwortzeiten für die LUBM-Anfragen im verteilten (links) und zentralisierten Fall (rechts) mit steigender Größe der Wissensbasen. Für den verteilten Fall sind außerdem die Antwortzeiten für die beteiligten Knoten auf den Rechnern 3 und 4 dargestellt.

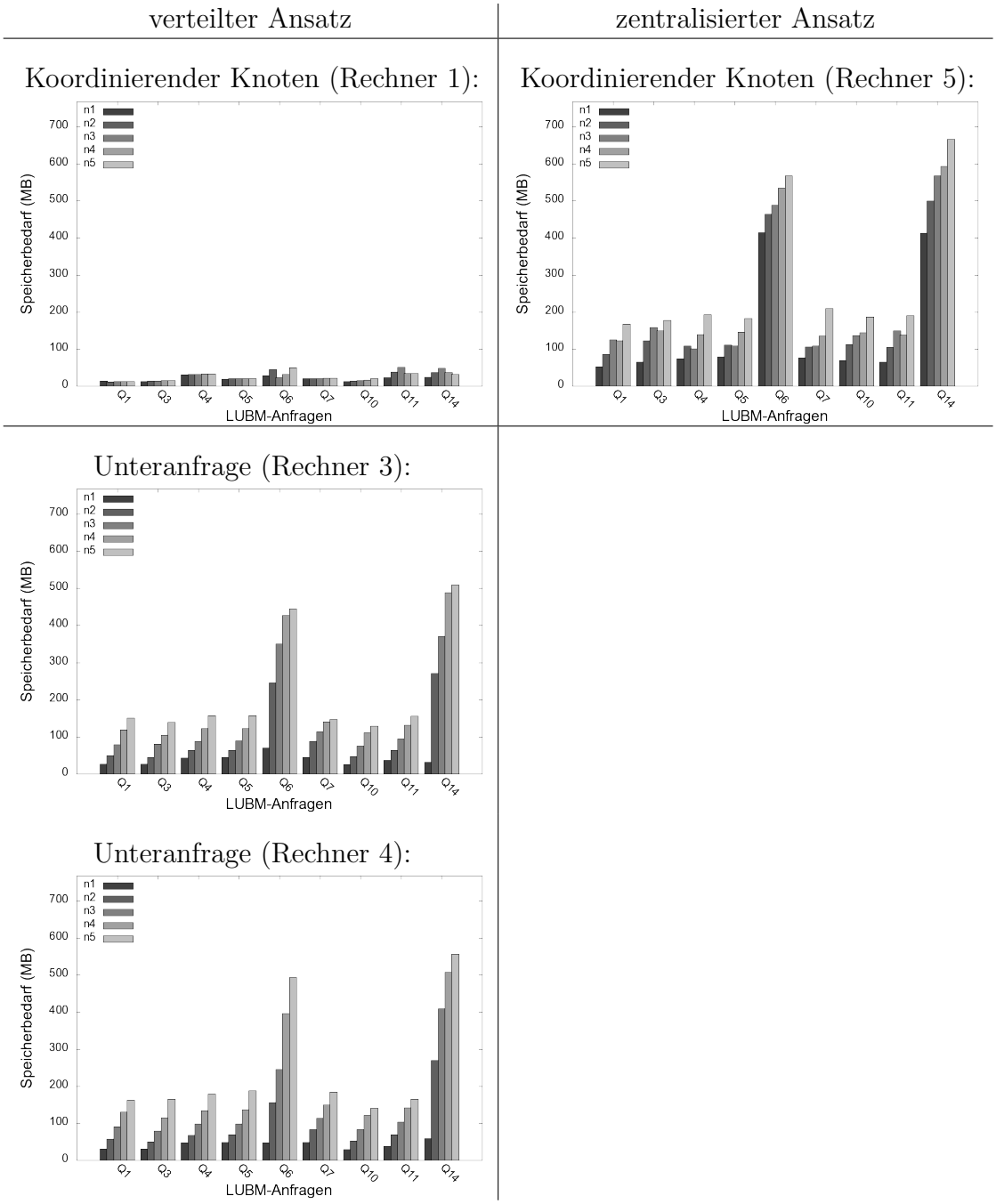

Abbildung 5.13.: Gegenüberstellung des Speicherbedarfs für die LUBM-Anfragen im verteilten (links) und zentralisierten Fall (rechts) mit steigender Größe der Wissensbasen. Für den verteilten Fall ist außerdem jeweils der Speicherbedarf der beteiligten Knoten auf den Rechnern 3 und 4 dargestellt.

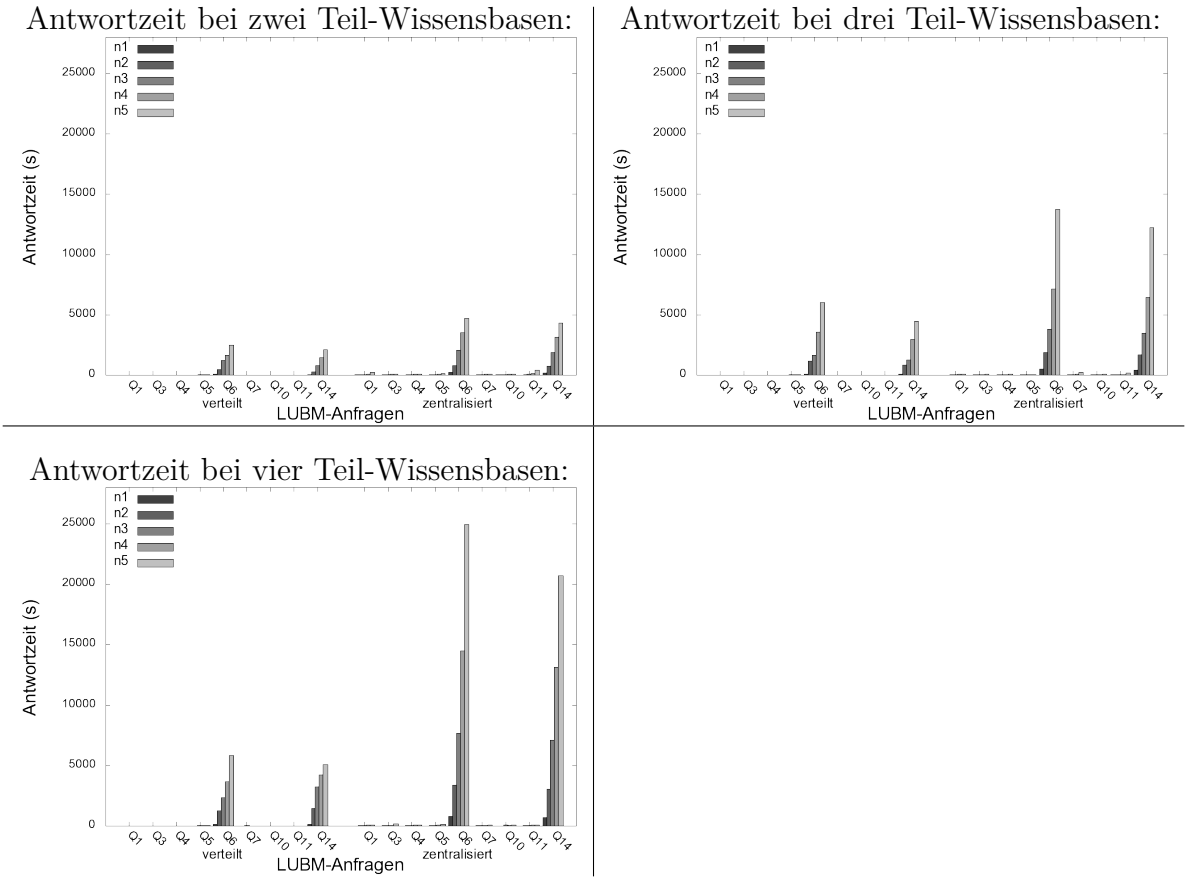

Abbildung 5.14.: Gegenüberstellung der Antwortzeiten im verteilten Fall für die LUBM-Anfragen bei zwei bis vier Teil-Wissensbasen.

Gleichzeitig ist interessant, dass der Speicherbedarf auf Rechner 1, der die Ausführung des verteilten Ansatzes koordiniert, nur sehr gering ist und auch mit zunehmender Größe der Wissensbasis im Wesentlichen konstant bleibt, da der Koordinierungsaufwand dadurch nicht steigt.

Test 3: Antwortzeit in Abhängigkeit der Zahl der Teil-Wissensbasen. Um zu untersuchen, wie sich die Zahl der Teil-Wissensbasen auf die Reasoning-Effizienz auswirkt, wurde das oben geschilderte Experiment mit den Test-Wissensbasen wiederholt, die drei bzw. vier Teil-Wissensbasen umfassen. Die Antwortzeiten für sowohl den verteilten als auch den zentralisierten Fall sind in Abbildung 5.14 dargestellt.

In beiden Fällen steigen die Antwortzeiten mit zunehmender Zahl der Teil-Wissensbasen an. Im verteilten Fall fällt dieser Anstieg jedoch wesentlich geringer aus als im zentralisierten Fall. Dies lässt sich damit begründen, dass im verteilten Fall jeweils nur Teile der Gesamt-Wissensbasis berücksichtigt werden müssen.

Test 4: Speicherbedarf in Abhängigkeit der Zahl der Teil-Wissensbasen. Zusätzlich ist interessant, wie sich die Zahl der Teil-Wissensbasen auf den Speicherbedarf des koordinierenden Reasoning-Knoten auswirkt. Die Messergebnisse für zwei, drei und vier Teil-Wissensbasen sind in Abbildung 5.15 dargestellt. Es zeigt sich, dass der

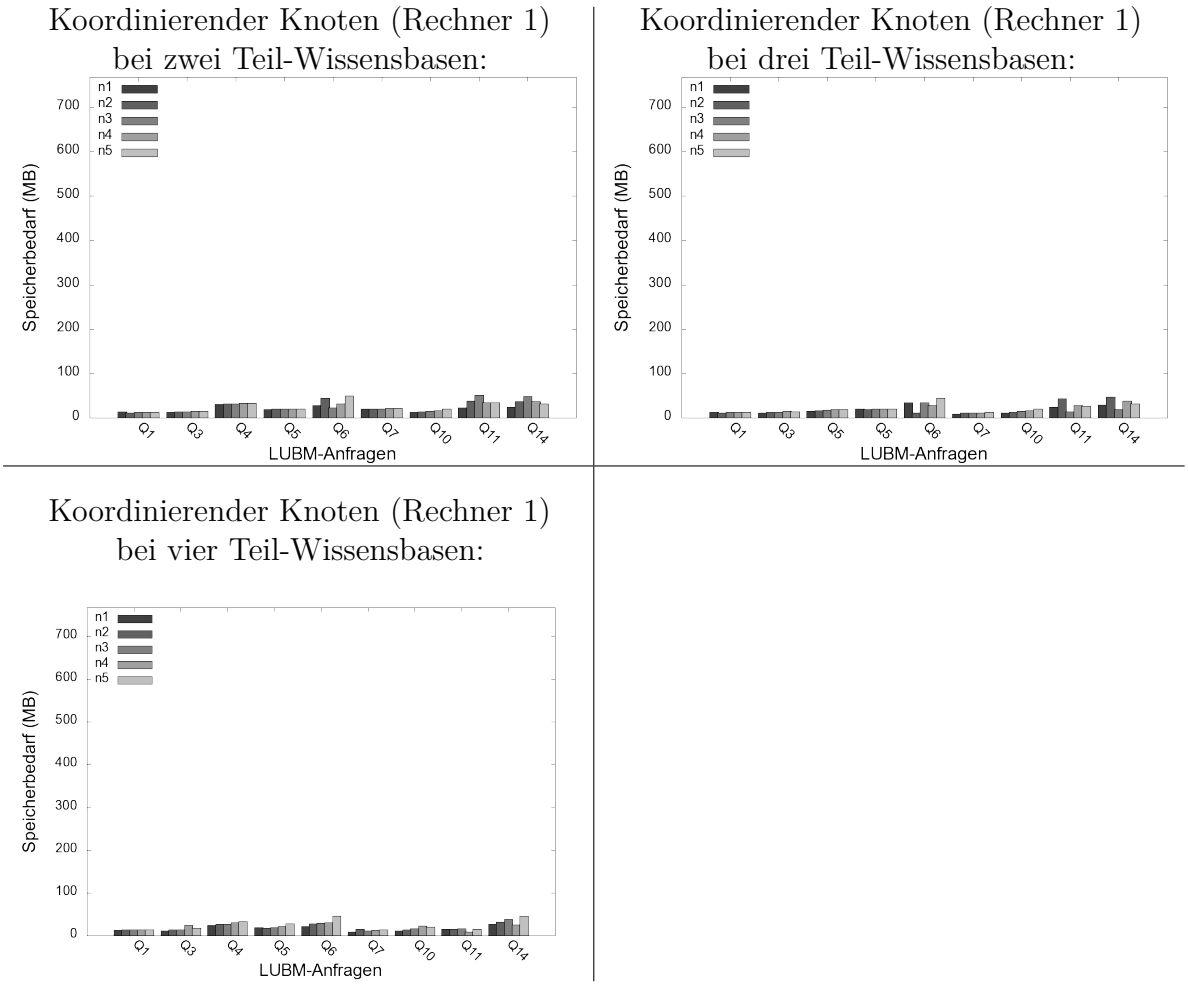

Abbildung 5.15.: Gegenüberstellung des Speicherbedarfs im verteilten Fall beim koordinierenden Knoten für die LUBM-Anfragen bei zwei bis vier Teil-Wissensbasen.

Speicherbedarf des koordinierenden Knotens bei Hinzunahme einer Teil-Wissensbasis nur unwesentlich steigt. Darüber hinaus ist bemerkenswert, dass sich auch mit zunehmender Größe der Wissensbasis keine wesentliche Veränderung ergibt.

5.5.3.2. RailBM Schienennetz-Benchmark

Aufgrund der eingeschränkten Eignung von LUBM zur Bewertung der vorgeschlagenen Verfahren wurde RailBM als zusätzlicher Benchmark konzipiert. RailBM repräsentiert die Domäne der Infrastrukturüberwachung realistischer, weil die Teil-Wissensbasen hier jeweils einen Teil der TBox verwenden, aber dennoch nicht partitioniert sind. Die Evaluierungsergebnisse werden im Folgenden im Detail vorgestellt.

Test 1: Antwortzeit in Abhängigkeit der Größe der Wissensbasis. Für jede RailBM-Test-Wissensbasis (siehe Tabelle 5.6) wurde folgender Ablauf durchgeführt: Die Reasoning-Knoten auf Rechner 3 und 4 wurden mit den Teil-Wissensbasen der verteilten Wissensbasis initialisiert. Dann wurden die acht RailBM-Test-Anfragen (siehe Abschnitt 5.5.2.2) zuerst an den Reasoning-Knoten auf Rechner 1 und dann an den Reasoning-Knoten auf Rechner 5 gestellt. Der Reasoning-Knoten auf Rechner 1 beantwortet die Anfragen mit Hilfe der Verfahren für eingeschränkt verteilte Wissensbasen (vergleiche Abschnitt 5.3.3), während der Reasoning-Knoten auf Rechner 5 zentralisiertes Reasoning durchführt. Dazu werden zunächst die Teil-Wissensbasen von den Rechnern 3 und 4 aggregiert und dann auf dieser aggregierten Wissensbasis die Anfragen beantwortet. Als *Antwortzeit* wird jeweils die Zeitspanne zwischen dem Absenden der Anfrage von Rechner 2 und dem Empfangen aller Lösungen auf Rechner 2 gemessen. Wie bei LUBM wird dieser Ablauf für jede Test-Wissensbasis dreimal wiederholt und dann der Durchschnitt der Antwortzeiten gebildet. Wie bereits theoretisch gezeigt, sind die Lösungen bei beiden Ansätzen identisch.

Die gemessenen Antwortzeiten sind in Abbildung 5.16 dargestellt. Sie sind für die unterschiedlich großen Test-Wissensbasen jeweils nach Anfragen gruppiert. Dabei zeigt sich, dass der verteilte Ansatz wesentliche Effizienzgewinne bringt. Dies lässt sich besonders deutlich in Abbildung 5.17 (logarithmische Skala) erkennen, wo die Antwortzeiten nach der Größe der Wissensbasis gruppiert und der verteilte dem zentralisierten Fall direkt gegenüber gestellt ist.

Dabei kann grob zwischen zwei Klassen von Anfragen unterschieden werden: Klasse eins umfasst die Anfragen 1, 2, 3 und 4, welche alle Rollen-Anfrageatome sind. Hier beträgt die Antwortzeit im verteilten Fall weniger als 1% der Antwortzeit im zentralisierten Fall. Dabei benötigt der zentralisierte Fall zur Aggregation der Wissensbasis im Durchschnitt etwa 30% der Gesamtzeit. Klasse zwei umfasst die Anfragen 5, 6, 7 und 8, welche alle Konzept-Anfrageatome sind. Auch hier kann eine erhebliche Effizienzsteigerung beobachtet werden, jedoch nur auf durchschnittlich etwa 50% der zentralisierten Antwortzeit. Dabei macht die Aggregation der Wissensbasis im zentralisierten Fall im Durchschnitt nur etwa 10% der Gesamtzeit aus. Die erhebliche Effizienzsteigerung lässt sich mit der beschreibungslogisch aufwändigeren Bearbeitung von Konzept-Anfrageatomen erklären. Dabei muss nach der Art des Konzept-Anfrageatoms unterschieden werden: Bei Anfrage 5 (entfernte Instanz-Prüfung) ist die Ersparnis sehr hoch, da im verteilten Fall zur Beantwortung nur eine der Teil-Wissensbasen berücksichtigt werden muss. Anfrage 6 (verteilte Instanz-Prüfung) führt jedoch zur Generierung eines Wissensbasis-Kerns über alle Teil-Wissensbasen; dadurch ist die Antwortzeit sogar deutlich höher als im zentralisierten Fall. (Die später vorgestellten Experimente mit *steigender* Zahl an Teil-Wissensbasen zeigen jedoch, dass die Ant-

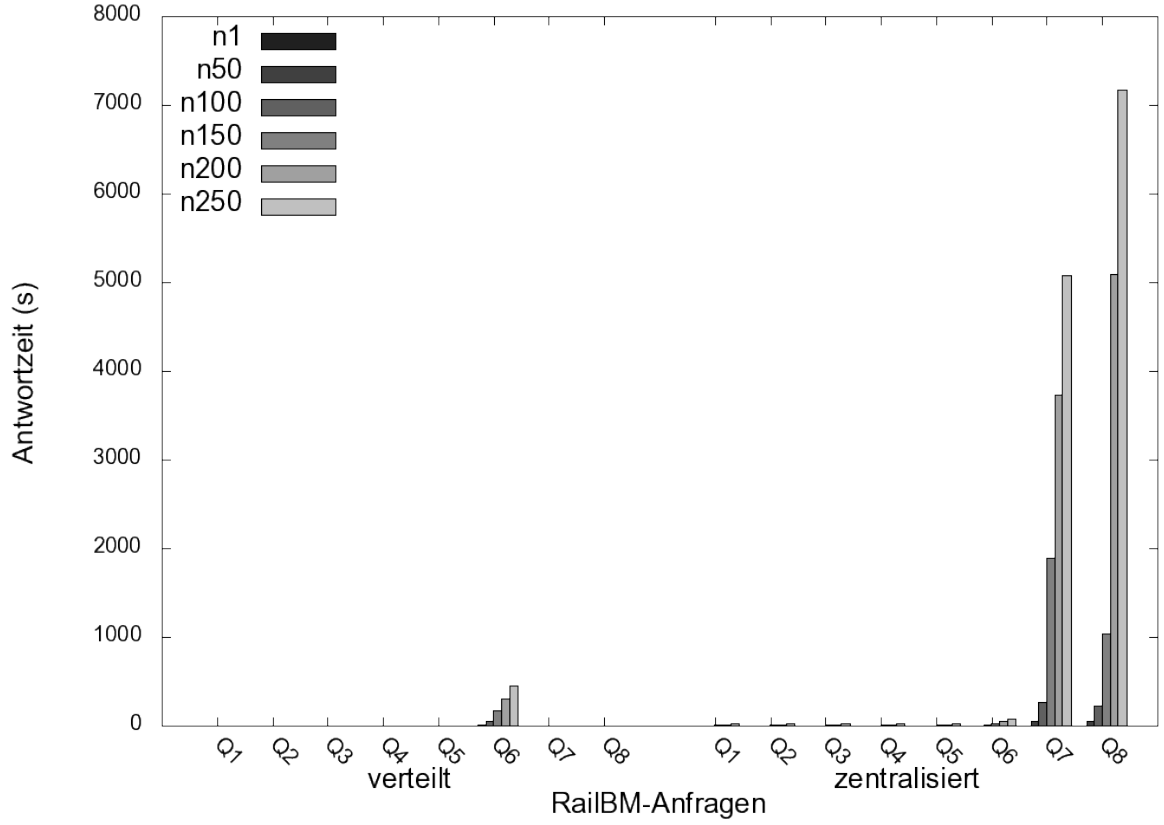

Abbildung 5.16.: Gegenüberstellung der Antwortzeiten für die RailBM-Anfragen mit steigender Größe der Wissensbasen im verteilten (links) und zentralisierten Fall (rechts).

wortzeit im zentralisierten Fall wesentlich stärker ansteigt als im verteilten und diesen bei vier Teil-Wissensbasen bereits übertrifft.) Bei den Anfragen 7 (entfernte Konzept-Realisierung) und 8 (verteilte Konzept-Realisierung) ist zunächst nicht nachvollziehbar, warum sich im verteilten Fall eine so starke Ersparnis ergibt.

Zur Beantwortung dieser Frage wurde das Experiment leicht modifiziert: Anstatt die Wissensbasis für alle acht Anfragen einmal zu Beginn zu initialisieren, wurde die Wissensbasis für *jede* Anfrage erneut initialisiert. Die resultierenden Antwortzeiten sind in Abbildung 5.18 (logarithmische Skala) dargestellt, wo die Ergebnisse ohne Reinitialisierung links und die Ergebnisse mit Reinitialisierung rechts aufgetragen sind. Hier zeigt sich auch für Anfragen 7 und 8 der erwartete Anstieg mit zunehmender Größe der Wissensbasis. Dass dieser bei der ersten Messung ausblieb, lässt sich mit Optimierungen der Reasoner-Implementierungen erklären: Die verteilte Instanz-Prüfung von Anfrage 6 erfordert die Generierung eines Wissensbasis-Kerns. Dabei werden Indexstrukturen und sonstige Zwischenergebnisse für die Teil-Wissensbasen erzeugt, die anschließend bei der Beantwortung der Anfragen 7 und 8 wiederverwendet werden können. Wird die Wissensbasis neu initialisiert, sind diese Zwischenergebnisse jedoch nicht mehr verfügbar und der erwartete Anstieg der Antwortzeiten zeigt sich auch bei Anfragen 7 und 8. Vergleicht man auch hier den verteilten mit dem zentralisierten Fall (Abbildung 5.19), zeigt sich bei Anfragen 7 und 8 jedoch weiterhin eine deutliche

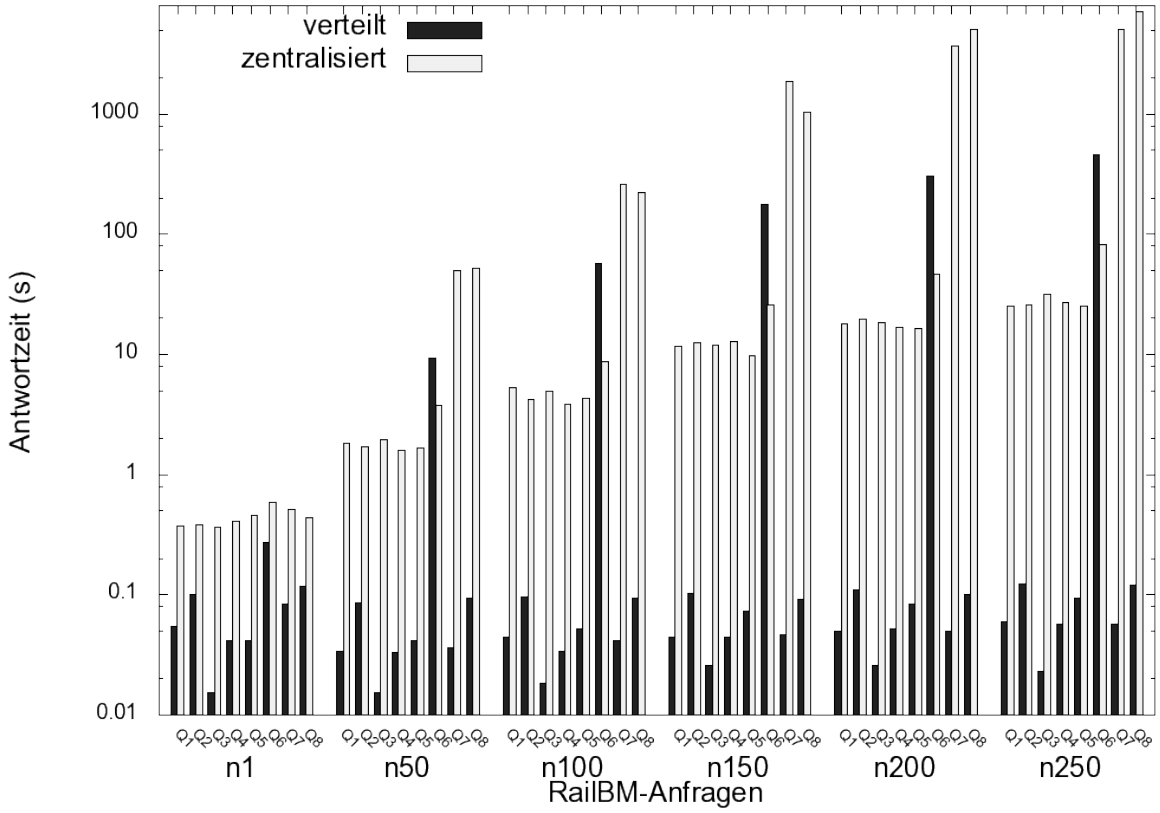

Abbildung 5.17.: Direkte Gegenüberstellung der Antwortzeiten für die RailBM-Anfragen bei steigender Größe der Wissensbasen.

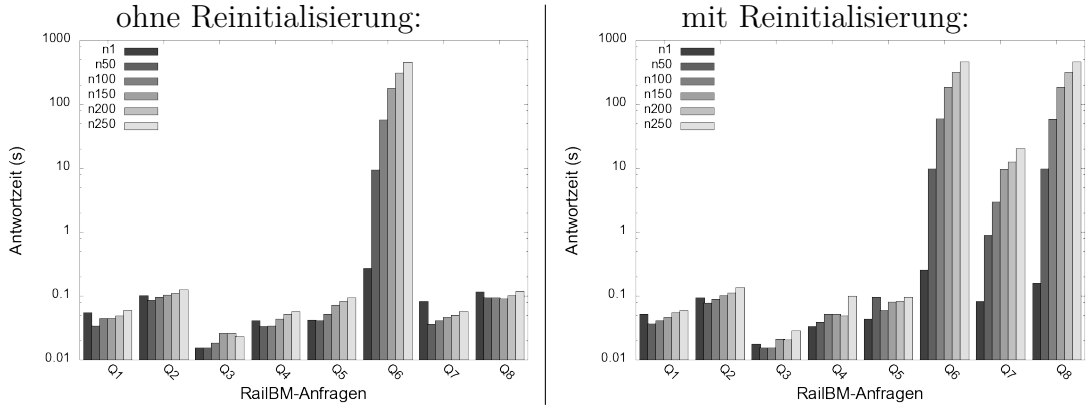

Abbildung 5.18.: Auswirkung der Reinitialisierung auf die Antwortzeiten für die RailBM-Anfragen beim verteilten Ansatz.

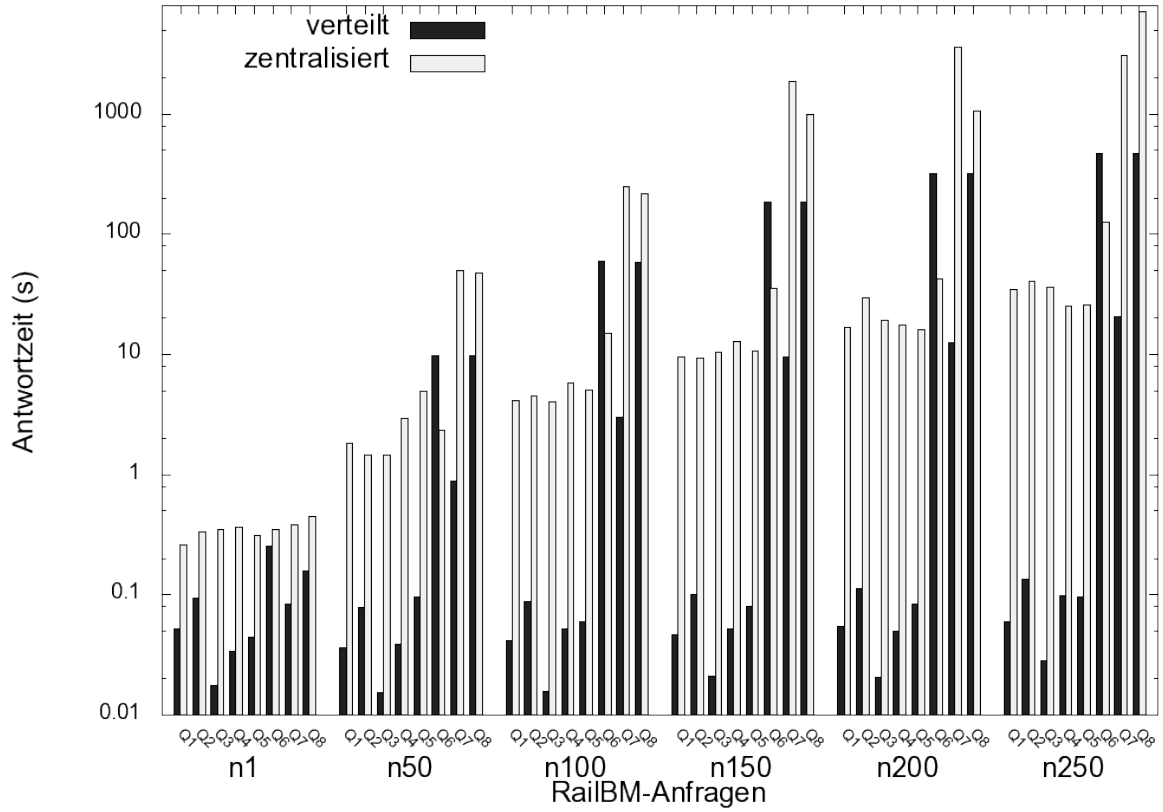

Abbildung 5.19.: Direkte Gegenüberstellung der Antwortzeiten für die RailBM-Anfragen mit Reinitialisierung bei steigender Größe der Wissensbasen.

Ersparnis beim verteilten im Vergleich zum zentralisierten Fall. Vor allem bleibt im verteilten Fall die Antwortzeit für die verteilte Instanz-Prüfung und die Konzept-Realisierung konstant, während sie im zentralisierten Fall stark ansteigt.

Welches Verhalten lässt sich nun unter realen Bedingungen erwarten? Geht man davon aus, dass in einem Zustandsüberwachungssystem fortlaufend neue Überwachungsmerkmale produziert und den Teil-Wissensbasen dadurch permanent neue Individuen hinzugefügt werden, muss ohne weitere Optimierungen von einem Verhalten wie auf der rechten Seite von Abbildung 5.18 ausgegangen werden – Zwischenergebnisse können nicht wiederverwendet werden, weil sich die Datenbasis häufig ändert. Es ist jedoch zu erwarten, dass zukünftige Reasoner-Implementierungen in der Lage sein werden, auch dann Zwischenergebnisse wiederzuverwenden, wenn sich die Wissensbasis teilweise verändert hat, z. B. weil neue Individuen hinzugekommen sind. Entsprechende Arbeiten zu *inkrementellem Reasoning* wurden bereits veröffentlicht und auch schon umgesetzt [HWPS06b, HWPS06a, HWH07]. Da die vorgestellten Verfahren nur Annahmen über die Schnittstellen der einzelnen Reasoning-Knoten in einem verteilten Reasoning-System (und nicht die Implementierung) treffen, ist es problemlos möglich, entsprechend optimierte Reasoner-Implementierungen einzubinden. Dann ist zu erwarten, dass sich das tatsächliche Antwortverhalten auch bei kontinuierlicher Erweiterung der Teil-Wissensbasen dem auf der linken Seite von Abbildung 5.18 dargestellten Verhalten annähert.

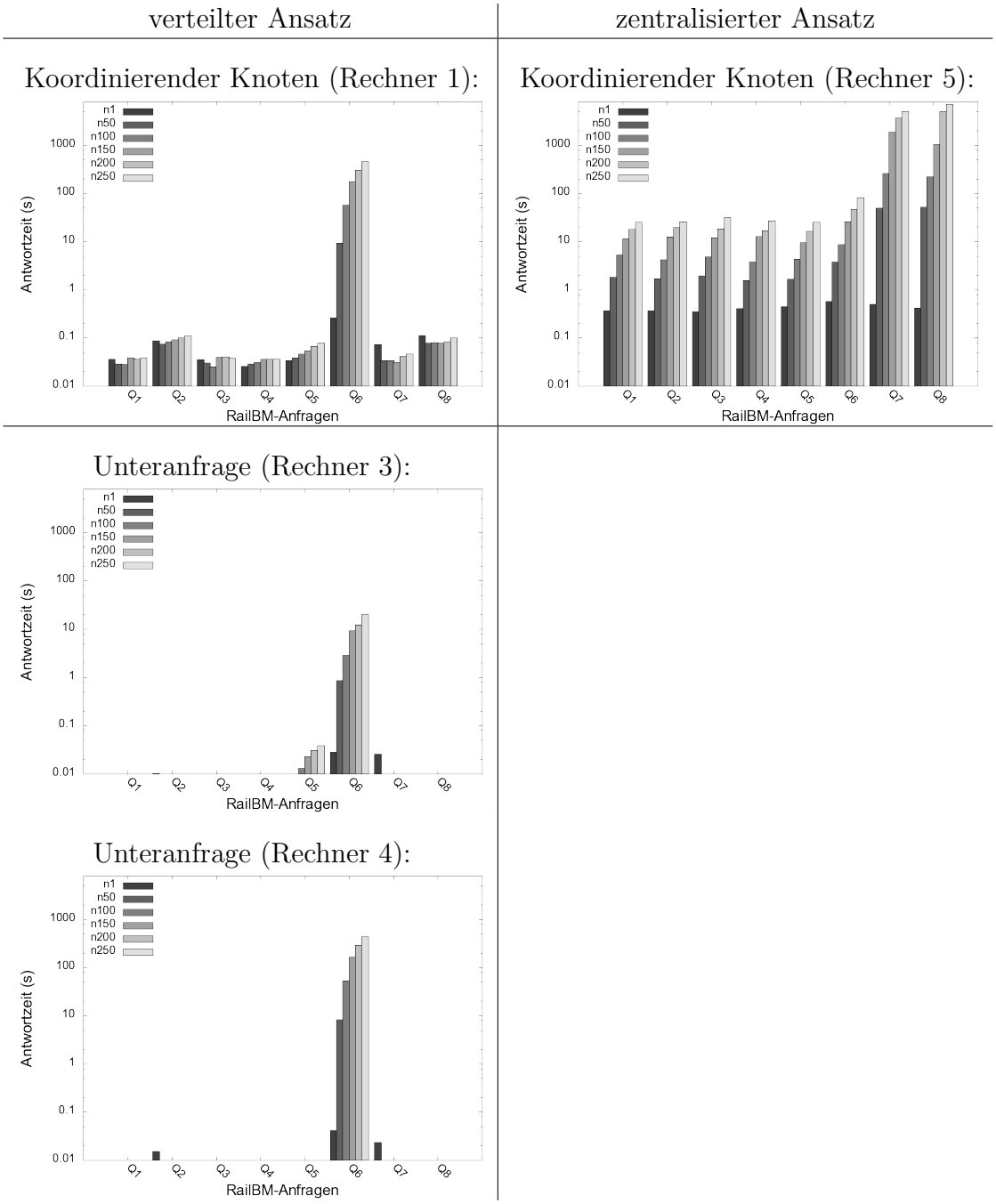

Abbildung 5.20.: Gegenüberstellung der Antwortzeiten für die RailBM-Anfragen im verteilten (links) und zentralisierten Fall (rechts) mit steigender Größe der Wissensbasen. Für den verteilten Fall sind außerdem die Antwortzeiten für die beteiligten Knoten auf den Rechnern 3 und 4 dargestellt.

Die Anteile der einzelnen Reasoning-Knoten an der Gesamt-Antwortzeit sind noch einmal in Abbildung 5.20 aufgeschlüsselt. Es zeigt sich, dass die Knoten auf den Rechnern 3 und 4 vor allem bei der Beantwortung von Anfrage 6 gefordert sind, wo ein verteiltes Realisierungs-Problem gelöst und der Wissensbasis-Kern generiert werden muss. (Dabei ist die Teil-Wissensbasis auf Knoten 4 komplexer als auf Knoten 3.) Für Anfrage 5 muss nur Rechner 3 eine lokale Realisierung durchführen. Bei Anfragen 7 und 8 kann auf die Zwischenergebnisse von Anfrage 6 zurückgegriffen und die Antwortzeit damit wesentlich verkürzt werden.

Test 2: Speicherbedarf in Abhängigkeit der Größe der Wissensbasis. Bei der Durchführung des oben beschriebenen Experiments wurde nach der Beantwortung jeder Anfrage außerdem wieder der Speicherbedarf gemessen. Die Ergebnisse sind in Abbildung 5.21 dargestellt.

Der Speicherbedarf auf Knoten 3 und 4 beträgt in der Regel jeweils weniger als die Hälfte des Speicherbedarfs im zentralisierten Fall. Dies bedeutet, dass beim verteilten Ansatz auch in der Summe über alle beteiligten Reasoning-Knoten nicht mehr Speicher benötigt wird als beim zentralisierten. Dies zeigt sich vor allem bei Anfragen der Klasse zwei, und dort besonders bei den Anfragen 7 und 8. Hier wirkt sich stark aus, dass jeder Reasoning-Knoten nur einen Teil der Wissensbasis verarbeiten muss.

Bemerkenswert ist wieder, dass der Speicherbedarf des koordinierenden Knotens sehr gering ist und sich auch bei Anfragen der Klasse zwei, die Reasoning auf einem Wissensbasis-Kern durchführen, nicht wesentlich erhöht. Es ist außerdem zu erkennen, dass der Speicherbedarf mit steigender Größe der Wissensbasis weniger stark steigt als im zentralisierten Fall, da die Größe des Wissensbasis-Kerns in der Regel unabhängig von der Größe der Wissensbasis ist.

Test 3: Antwortzeit in Abhängigkeit der Zahl der Teil-Wissensbasen. Das oben beschriebene Experiment wurde zudem mit verteilten Wissensbasen wiederholt, die aus drei bzw. vier Teil-Wissensbasen bestehen. Dabei ließen sich die in Abbildung 5.22 dargestellten Auswirkungen auf die Antwortzeit beobachten. Abbildung 5.23 stellt die Details für den verteilten Fall dar.

Sowohl im verteilten als auch zentralisierten Fall erhöht sich durch Hinzunahme einer Teil-Wissensbasis erwartungsgemäß die Antwortzeit (siehe Abbildung 5.22). Während es beim zentralisierten Fall zu einer Vervielfachung kommt und vor allem für Anfragen der Klasse zwei die absoluten Zeiten explodieren, ist die Erhöhung im verteilten Fall wesentlich geringer (siehe Abbildung 5.23). Sie beträgt bei Hinzunahme einer Teil-Wissensbasis sowohl für Anfragen der Klasse eins als auch zwei durchschnittlich nur etwa 30%. Hier führt die Verteilung von Kontextinformationen über mehrere Teil-Wissensbasen also zu einer wesentlich besseren Skalierbarkeit der Reasoning-Verfahren.

Test 4: Speicherbedarf in Abhängigkeit der Zahl der Teil-Wissensbasen. Schließlich wurde die Veränderung des Speicherbedarfs für den koordinierenden Knoten mit steigender Zahl von Teil-Wissensbasen beobachtet. Die Ergebnisse sind in Abbildung 5.24 dargestellt.

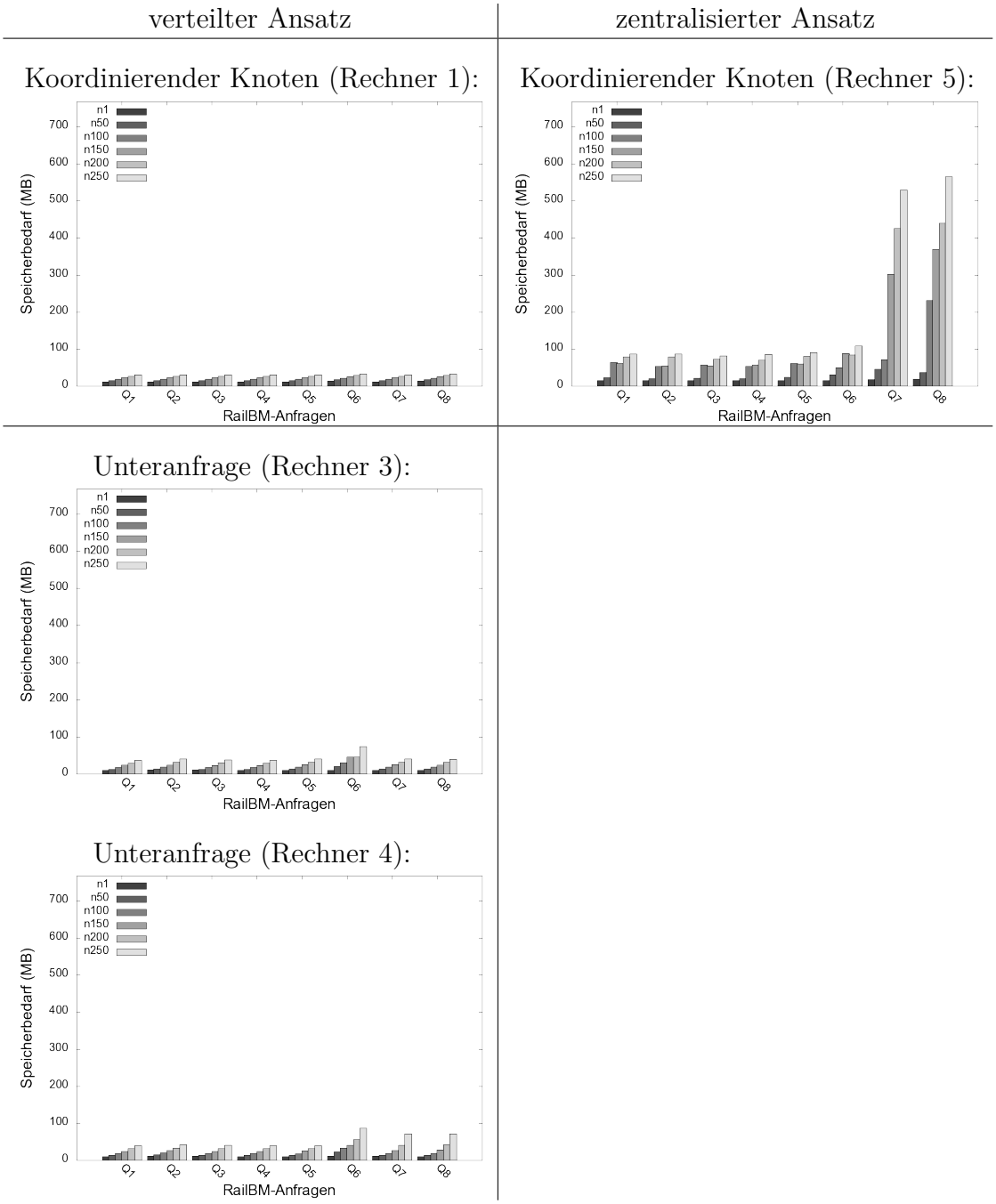

Abbildung 5.21.: Gegenüberstellung des Speicherbedarfs für die RailBM-Anfragen im verteilten (links) und zentralisierten Fall (rechts) mit steigender Größe der Wissensbasen. Für den verteilten Fall ist außerdem jeweils der Speicherbedarf der beteiligten Knoten auf den Rechnern 3 und 4 dargestellt.

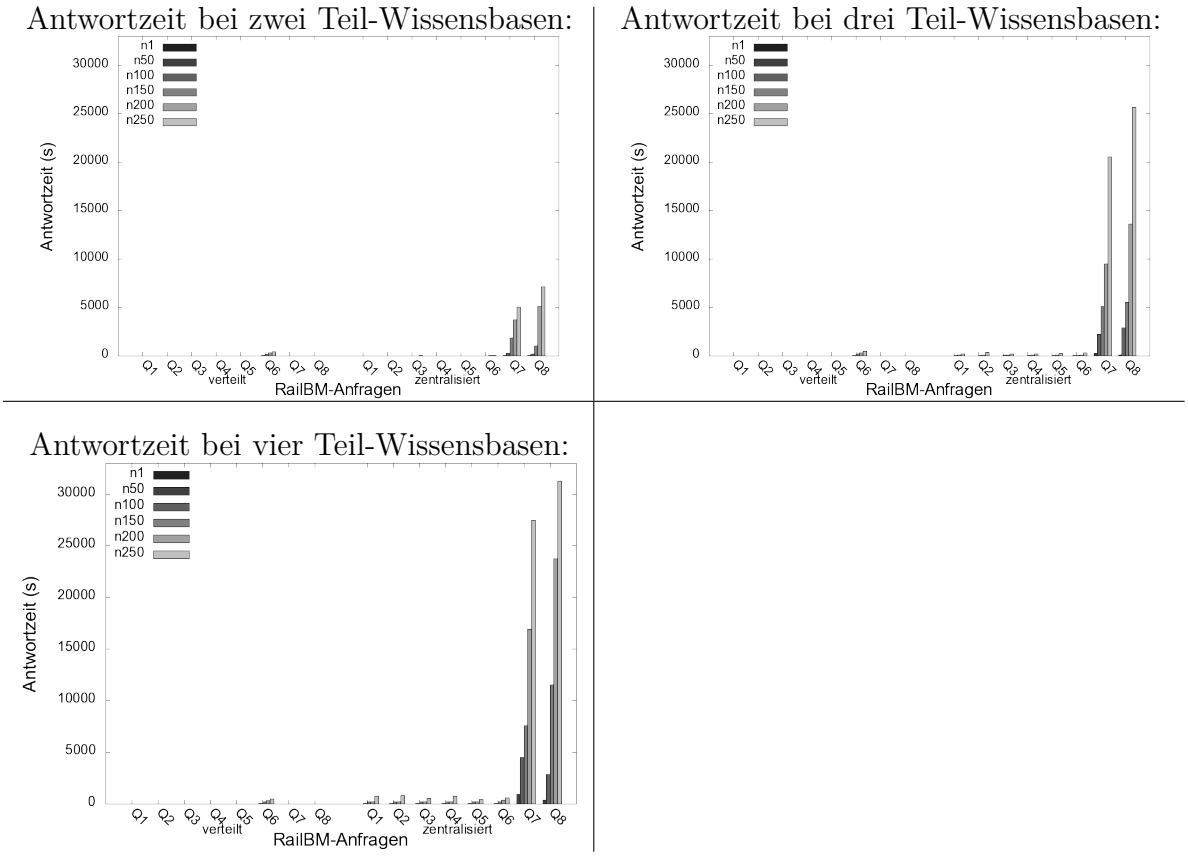

Abbildung 5.22.: Gegenüberstellung der Antwortzeiten im verteilten Fall für die RailBM-Anfragen bei zwei bis vier Teil-Wissensbasen.

Es zeigt sich, dass sich der Speicherbedarf nur unwesentlich erhöht. Dies gilt selbst für Anfragen der Klasse zwei, wo Reasoning auf einem Wissensbasis-Kern durchgeführt werden muss. Dies lässt sich wieder damit erklären, dass die Größe des Wissensbasis-Kerns von der Größe der Wissensbasis entkoppelt ist.

5.5.3.3. Zusammenfassung der experimentellen Evaluierung

Dieser Abschnitt vergleicht die vorgeschlagene Implementierung der Verfahren zum verteilten Reasoning mit dem alternativen Ansatz des zentralisierten Reasonings. Für die Test-Wissensbasen und -Anfragen der beiden Benchmarks LUBM und RailBM wurden dabei Antwortzeit und Speicherbedarf gemessen.

Obwohl die Kontextinformationen verteilt waren, hat sich der verteilte Ansatz in beiden Fällen als effizienter als der zentralisierte Ansatz erwiesen. Vollständigkeit und Korrektheit der Antworten wurden bereits theoretisch nachgewiesen. Bei RailBM fielen die Effizienzgewinne besonders hoch aus, weil die RailBM-Test-Wissensbasen repräsentativ für verteilte Wissensbasen bei der Infrastrukturüberwachung sind. LUBM generiert hingegen Test-Wissensbasen, die von vornherein partitioniert sind, weshalb sie von entsprechend optimierten Reasoner-Implementierungen auch im zentralisierten Fall effizient bearbeitet werden.

Die Skalierbarkeit mit steigender Größe der Wissensbasen ist beim verteilten Ansatz

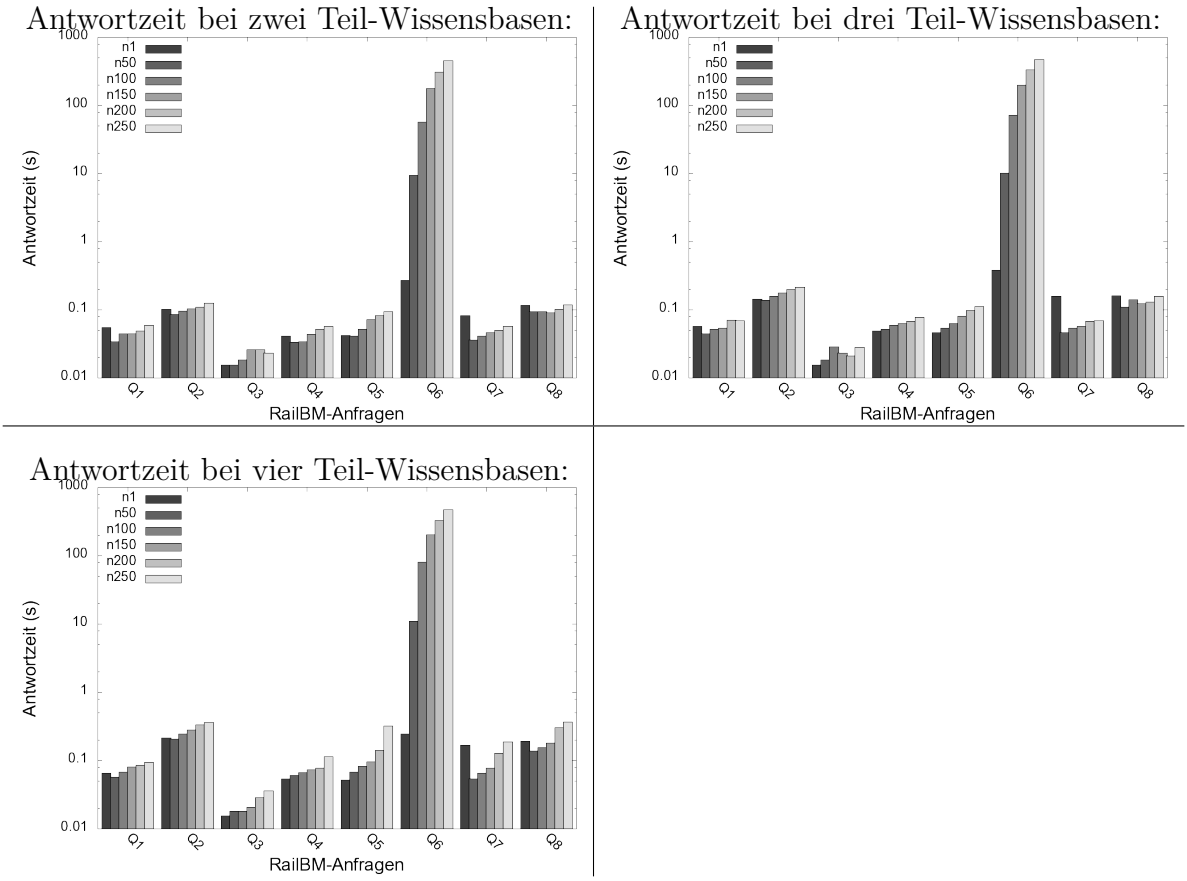

Abbildung 5.23.: Gegenüberstellung der Antwortzeiten im verteilten Fall für die RailBM-Anfragen bei zwei bis vier Teil-Wissensbasen.

wesentlich besser als beim zentralisierten. Dies gilt vor allem für Rollen-Anfrageatome. Aber auch Konzept-Anfrageatome, für die teilweise ein Wissensbasis-Kern generiert werden muss, werden im verteilten Fall effizient beantwortet. Dies zeigt sich wieder besonders deutlich bei RailBM. Dieses Verhalten ist darauf zurückzuführen, dass im verteilten Fall jeweils nur Teile der Gesamt-Wissensbasis berücksichtigt werden müssen, was sich aufgrund der exponentiellen Worst-Case-Komplexität der $\mathcal{SHIN}$-Beschreibungslogik überproportional auswirkt.

Bei der Skalierbarkeit bezüglich der Zahl von Teil-Wissensbasen ergibt sich durch die Verteiltheit der Kontextinformationen im verteilten Fall ebenfalls kein Mehraufwand: Für LUBM ist die Ersparnis zwar gering, weil die Partitionierung der LUBM-Wissensbasen auch im zentralisierten Fall ausgenutzt werden kann. Für RailBM zeigt sich im Vergleich jedoch ein wesentlich geringerer Anstieg von Antwortzeit und Speicherbedarf. Bemerkenswert ist auch, dass der Ressourcenbedarf des koordinierenden Knotens von der Zahl der Teil-Wissensbasen nur unwesentlich beeinflusst wird. Bei Konzept-Anfrageatomen liegt dies vor allem auch daran, dass die Größe des Wissensbasis-Kerns in der Regel unabhängig von der Größe der Gesamt-Wissensbasis ist.

Abgesehen von den *relativen* Effizienzvorteilen des verteilten zum zentralisierten Ansatz, hat die experimentelle Evaluierung auch gezeigt, dass sich die Komplexität des $\mathcal{SHIN}$-Reasonings stark im *absoluten* Ressourcenbedarf niederschlägt. Die absolut

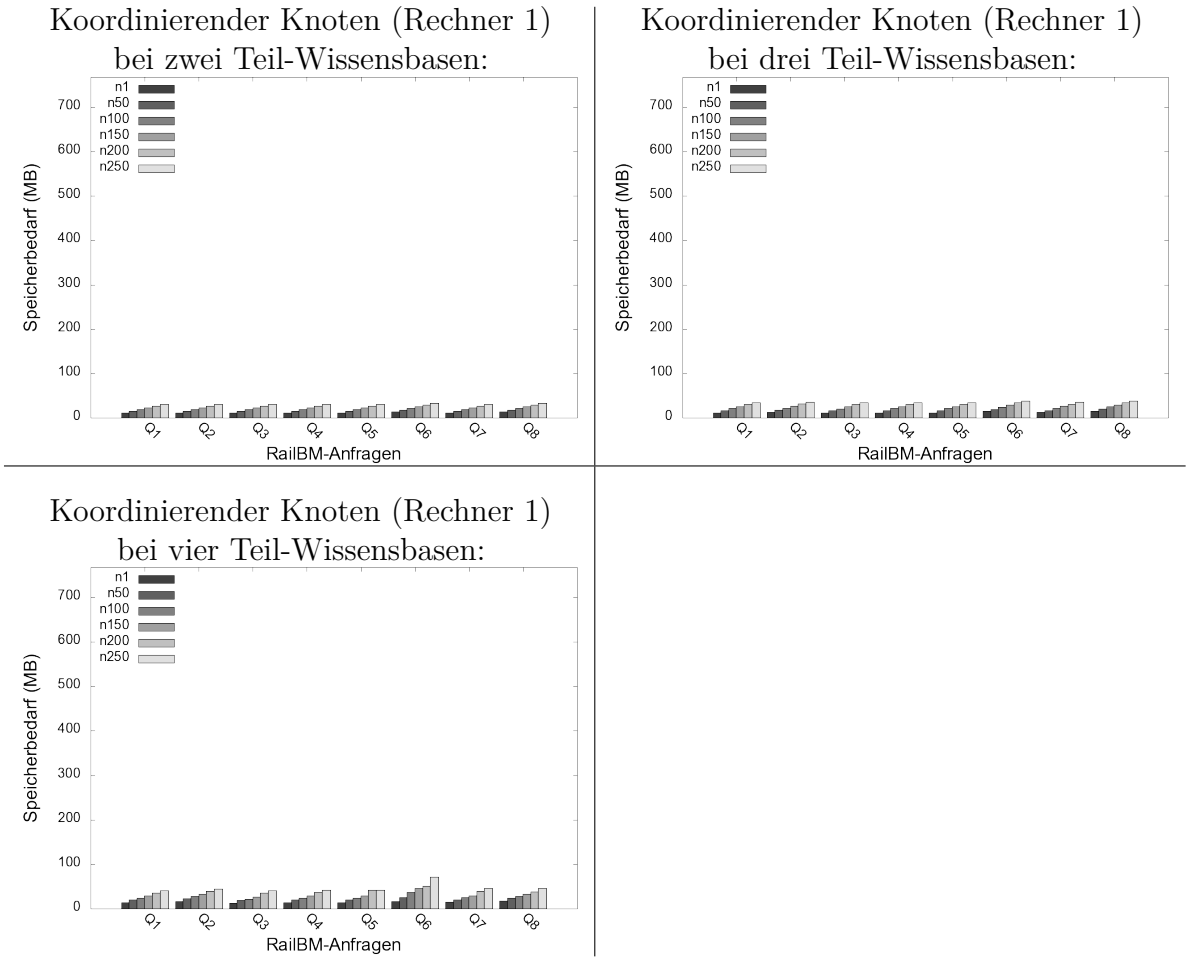

Abbildung 5.24.: Gegenüberstellung des Speicherbedarfs im verteilten Fall beim koordinierenden Knoten für die RailBM-Anfragen bei zwei bis vier Teil-Wissensbasen.

gemessenen Werte müssen jedoch vor folgendem Hintergrund gesehen werden:

- Die der Evaluierung zugrunde liegende prototypische Implementierung bietet noch viel Optimierungspotential: So erfolgt die Generierung des Wissensbasis-Kerns über mehrere Teil-Wissensbasen noch sequentiell, sie kann jedoch problemlos parallelisiert werden. Auch bei der Bearbeitung konjunktiver Anfragen können die Heuristiken zur Anfrageoptimierung noch verbessert werden.

- Auch die Effizienz der Reasoner-Implementierungen wird durch neue Optimierungen stetig erhöht [HLMS08]. Natürlich stellt die ExpTime-Worst-Case-Komplexität eine theoretische Grenze dar. Die letzten Jahre haben jedoch gezeigt, dass sie für praktische Reasoning-Probleme nur selten erreicht wird. Da die Verfahren zur Anfragebearbeitung die Reasoner-Komponenten als austauschbare Komponenten betrachtet, lassen sich effizientere Reasoner-Implementierungen zukünftig problemlos übernehmen.

- Im realen Einsatz dürften mehr Ressourcen verfügbar sein als in der Testum-

gebung: Für die Experimente wurden fünf einfache Arbeitsplatzrechner für den Büroeinsatz verwendet, die nicht für den Server-Betrieb vorgesehen sind. Auch wurde während der Messungen umfangreiches Logging durchgeführt, was ebenfalls die Leistung beeinträchtigt. Für den Produktionsbetrieb ist deshalb eine Leistungssteigerung zu erwarten.

Prinzipiell hat die experimentelle Evaluierung gezeigt, dass die stärkste Leistungssteigerung erreicht werden kann, wenn die Größe der zu berücksichtigenden Wissensbasis verringert wird. Diese Erkenntnis motiviert das Prinzip, immer nur die für eine gegebene Anfrage *relevante* Untermenge der Wissensbasis in den Reasoning-Knoten zu laden. Ein entsprechendes Ziel wird auch im folgenden Kapitel verfolgt, wo die Menge der zu berücksichtigenden Kontextinformationen anhand der Topologie des Infrastrukturnetzes beschränkt wird. Es wird außerdem bei der Konzeption der Systemarchitektur in Kapitel 7 aufgegriffen, die zwischen Komponenten differenziert, die große Mengen an Kontextinformationen archivieren und verwalten, und solchen, die Reasoning über eine kleine Untermenge dieser Kontextinformationen durchführen.

5.6. Zusammenfassung

Dieses Kapitel stellte Verfahren zur Anfragebeantwortung in einem verteilten Reasoning-System vor. Damit konnte Reasoning über verteilte Kontextinformationen umgesetzt werden. Die Notwendigkeit der Verteilung der Wissensbasen ergab sich aus den Anforderungen der Infrastrukturüberwachung mit ihren großen Datenmengen und den vielen beteiligten Organisationen. Da es zur Modellierung der übergreifenden Zusammenhänge nötig ist, eine einheitliche Systemontologie einzusetzen, waren bisherige Ansätze nicht anwendbar. Deshalb wurde ein eigenes Konzept auf Basis eines verteilten Reasoning-Systems vorgeschlagen. Da es kein effizientes Verfahren für eine *beliebig* verteilte Wissensbasis geben kann, wurden unterschiedliche Verteilungsklassen untersucht. Dabei wurde insbesondere die eingeschränkte Verteilung detailliert vorgestellt, da sie einen attraktiven Kompromiss zwischen der für Infrastrukturüberwachung erforderlichen Verteilung und effizienter Anfragebearbeitung darstellt. Korrektheit und Vollständigkeit der vorgestellten Verfahren wurden nachgewiesen. Auf Basis einer prototypischen Implementierung als Framework wurden zudem Experimente durchgeführt, um die Effizienz und Praktikabilität des vorgestellten Verfahrens bewerten zu können. Dabei hat sich herausgestellt, dass trotz Verteiltheit der Wissensbasis im Vergleich zu der zentralisierten Alternative – abhängig von der Komplexität der Anfrage – erhebliche Effizienzgewinne bei Antwortzeit und Speicherbedarf erzielt werden. Das zeigt, dass die vorgeschlagene Verteilung des Reasonings sowohl den Erfordernissen der Infrastrukturüberwachung entspricht, als auch in der Lage ist, die Vorteile der formalen Wissensrepräsentation von Kontextinformationen und Infrastrukturzuständen umzusetzen.

6. Reasoning mit Topologiebezug

Im vorherigen Kapitel wurden Verfahren zum Reasoning über verteilte Kontextinformationen entwickelt. Damit können Zustände von Überwachungsobjekten abgeleitet werden, die explizit angegeben werden, wie z. B. der Zustand eines bestimmten Gleisabschnitts. Zudem können Zustände von Überwachungsobjekten ermittelt werden, deren Bezug zu einem explizit angegebenen Überwachungsobjekt mittels konjunktiver Anfragen ausgedrückt werden kann: Ist in der Systemontologie beispielsweise modelliert, dass ein Gleisabschnitt über die Rolle hasSleepers mit seinen Eisenbahnschwellen assoziiert ist, dann muss nur ein bestimmter Gleisabschnitt angegeben werden, um mittels hasSleepers auch die (nicht bekannten) Eisenbahnschwellen abzufragen.

Für die Zustandsüberwachung von Infrastrukturnetzen ist zudem häufig der Bezug zur *Topologie* des Netzes wichtig. Beispiele hierfür sind folgende Anfragen:

> *Welche Oberleitungen in Fahrentfernung von 20 Kilometern vom Hauptbahnhof München müssen bevorzugt gewartet werden?*
>
> *Welche drei Umspannwerke sind Umspannwerk 143 nächstgelegen und voll einsatzbereit?*
>
> *Welches ist die für einen Gefahrguttransport am wenigsten risikoreiche Route von einer gegebenen Abfahrts- zu einer gegebenen Zielhaltestelle?*

Die in diesen Anfragen gesuchten Überwachungsobjekte sind mit Bezug zur Topologie der Infrastruktur spezifiert: Es werden beispielsweise die topologische Distanz (Fahrentfernung 20 Kilometer, die nächstgelegenen Umspannwerke) und die topologische Erreichbarkeit (auf dem Pfad von Abfahrts- zu Zielhaltestelle) verwendet. Mit den bisherigen konjunktiven Anfragen können derartige topologische Beziehungen jedoch nicht ausgedrückt werden. Dazu müsste es möglich sein, Bedingungen an Pfade im Infrastrukturnetz zu stellen, wie beispielsweise eine Obergrenze für die topologische Distanz (Fahrentfernung 20 Kilometer) oder eine Optimalitätsbedingung („kürzester“ Pfad bezüglich des Risikos). Eine konjunktive Anfrage kann jedoch nur Pfade mit fester, vorgegebener Länge spezifizieren, bei Länge 4 beispielsweise $\{r(a,b), r(b,c), r(c,d), r(d,e)\}$.

Für die genannten Anfragetypen kann die Länge der Pfade im Vorhinein jedoch nicht angegeben werden – sie ergibt sich implizit aus der Topologie des Infrastrukturnetzes (z. B. alle Pfade zwischen Abfahrts- und Zielhaltestelle). Um diese Anfragetypen dennoch formulieren und beantworten zu können, entwickelt dieses Kapitel darum Konzepte zum Reasoning mit Topologiebezug. Damit demonstriert es außerdem, wie die einmal semantisch modellierten Kontextinformationen und Infrastrukturzustände (siehe Kapitel 4) von spezialisierten Auswertungsverfahren genutzt werden können, um unterschiedliche Probleme auf effiziente Weise zu lösen.

Zunächst werden Anforderungen an das Reasoning mit Topologiebezug in Abschnitt 6.1 formuliert. Abschnitt 6.2 sichtet und bewertet vor diesem Hintergrund

verwandte Arbeiten. Abschnitt 6.3 führt die Modellierung des Infrastrukturnetzes als Hierarchie von räumlichen Teil-Netzwerken ein. Auf dieser Basis kann in Abschnitt 6.4 ein allgemeingültiges Konzept beschrieben werden, um einen Bezug zwischen ortsbezogenen Kontextinformationen und der Netztopologie herzustellen. Abschnitt 6.5 stellt schließlich das im Rahmen dieser Arbeit entwickelte Konzept des topologiebezogenen Reasoning-Systems vor. Auf dessen Basis können Graphenalgorithmen mit der Auswertung semantisch modellierter Kontextinformationen kombiniert werden, um topologiebezogene Anfragen zu beantworten. Die vorgestellten Ergebnisse bauen auf [MFSW08, Mie07] auf. In Abschnitt 6.6 wird eine Softwarearchitektur zur Umsetzung dieser Verfahren als Framework vorgeschlagen. Abschnitt 6.7 nimmt eine abschließende Bewertung anhand der zuvor aufgestellten Anforderungen vor.

6.1. Anforderungen an Reasoning mit Topologiebezug

Bevor Anforderungen an topologiebezogenes Reasoning formuliert werden können, müssen zunächst einige Begriffe eingeführt werden.

6.1.1. Begriffe

Ein *Geoobjekt* (engl. *spatial object*) ist ein räumliches Element, das zusätzlich zu Sachinformationen geometrische und topologische Eigenschaften besitzt und zeitlichen Veränderungen unterliegen kann. Kennzeichnend für Geoobjekte sind somit *Geometrie*, *Topologie*, *Thematik* und *Dynamik* [Lan02, Küp05].

Die *Geometrie* umfasst alle Informationen zur absoluten Lage, Form und Ausdehnung eines Geoobjekts. Diese Angaben werden auf der Basis eines eindeutigen räumlichen Bezugssystems gemacht. Dabei ist der Punkt der Träger der geometrischen Information, auf dem die anderen Grundformen aufbauen. Weitere geometrische Grundformen sind Linie, Fläche und Körper.

Die *Topologie* beinhaltet alle Informationen über die relative Lage und andere Beziehungen zwischen Geoobjekten. Sie stellt eine Abstraktion der Geometrie dar. Topologische Beziehungen sind gegenüber topologischen Transformationen wie z. B. Dehnung, Drehung, Streckung und Stauchung invariant. Beispiele für topologische Beziehungen sind *Umgebungs-*, *Nachbarschafts-*, *Teilmengen-* oder *Inklusions-* sowie *Überdeckungs-* oder *Überschneidungsbeziehungen*. Topologische Grundformen sind Knoten, Kanten, Polygone und Polyeder.

Mit *Thematik* werden Sachinformationen bzw. thematische Informationen über Geoobjekte, mit *Dynamik* wird die zeitliche Variabilität der Geoobjekte bezeichnet.

Übertragen auf die Begriffe dieser Arbeit stellen damit sowohl Überwachungsobjekte als auch Kontextinformationen *Geoobjekte* dar:

- Ein *Überwachungsobjekt* ist ein räumliches Element mit einer Lagekoordinate im Raum, das in topologischen Beziehungen zu anderen Überwachungsobjekten stehen kann – zwei Gleisabschnitte können beispielsweise über eine Weiche verbunden sein. Die Thematik und die Dynamik von Überwachungsobjekten umfassen im Wesentlichen Merkmale, Symptome, Fehler und Zustände sowie ihre Veränderungen im Laufe der Zeit.

- Eine *Kontextinformation* stellt ebenfalls ein Geoobjekt dar, weil sie einen Bezugspunkt im Raum besitzt. Bei einer *ortsbezogenen* Kontextinformation, die eine Beobachtung für einen bestimmten Ort repräsentiert, besteht dieser direkt. Bei einer *bezeichnerbezogenen* Kontextinformation, die sich auf ein Überwachungsobjekt bezieht, besteht er indirekt über den Ortsbezug des Überwachungsobjekts. Die Thematik und die Dynamik einer Kontextinformation werden durch den Kontexttyp und den Zeitbezug dargestellt.

Die bisher entwickelten Konzepte ermöglichen es, die Thematik und die Dynamik von Überwachungsobjekten und Kontextinformationen beschreibungslogisch zu formalisieren (vergleiche Kapitel 4). Für das Reasoning mit Topologiebezug müssen in diesem Kapitel zusätzlich Konzepte zur Repräsentation und Verarbeitung von Geometrie und Topologie dieser Geoobjekte entwickelt werden.

6.1.2. Anforderungen

Anhand der Anfragebeispiele zu Beginn dieses Kapitels und der in Abschnitt 2.3 auf Seite 20 identifizierten Anforderungen an die Zustandsüberwachung für Infrastrukturnetze können folgende Anforderungen an Reasoning mit Topologiebezug abgeleitet werden:

A6.1 *Anwendbarkeit auf verschiedenste Infrastrukturnetze*: Diese Arbeit sucht generische Konzepte für die Zustandsüberwachung bei Infrastrukturnetzen. Verfahren für topologiebezogenes Reasoning dürfen also keine Annahmen über die Eigenschaften einer zugrunde liegenden Infrastruktur treffen, sondern müssen auf unterschiedliche Domänen anwendbar sein.

A6.2 *Föderierte Administration der Infrastrukturtopologie*: Zur Umsetzung des Topologiebezugs ist es nötig, die Topologie des Infrastrukturnetzes zu modellieren. Dabei muss auch berücksichtigt werden, wer diese Modellierung vornimmt und wo das Wissen über die zu modellierende Netztopologie vorhanden ist. Wie Kapitel 2 gezeigt hat, werden große Infrastrukturen wie Schienen- und Stromnetze national und vor allem international von verschiedenen Organisationen betrieben. Dementsprechend ist auch das Wissen über die Topologie der verschiedenen Infrastrukturteile über mehrere Organisationen verteilt (Anforderungen A2.5 und A2.6). Aufgrund von Baumaßnahmen, Stilllegungen und Erweiterungen kommt es zudem permanent zu Aktualisierungen (A2.7). Dies muss bei der Modellierung der Topologie geeignet berücksichtigt werden.

A6.3 *Integration ortsbezogener Kontextinformationen*: Zum Zweck der Zustandsüberwachung werden Elemente des Infrastrukturnetzes als Überwachungsobjekte angesehen und die zu ihnen verfügbaren Kontextinformationen integriert betrachtet. Beispiele für derartige Überwachungsobjekte sind Gleisabschnitte, Weichen und Oberleitungen bzw. Stromleitungen, Transformatoren und Umspannwerke. Bei den zu integrierenden Kontextinformationen kann wiederum zwischen bezeichner- und ortsbezogenen unterschieden werden: *Bezeichnerbezogene* Kontextinformationen besitzen einen Bezug zu einem Überwachungsobjekt. Dazu gehören beispielsweise Merkmale, die von einem stationären Weichen-Überwachungssystem zu

einer bestimmten Weiche produziert werden. *Ortsbezogene* Kontextinformationen besitzen hingegen lediglich einen Ortsbezug, d. h. Lagekoordinaten in einem räumlichen Bezugssystem. Dazu gehören Umgebungsbedingungen wie Temperatur, Niederschlagsmenge usw. oder auch Messungen von mobilen Überwachungssystemen, wie sie beispielsweise auf Zügen montiert sind. Damit ortsbezogene Kontextinformationen trotz des fehlenden Bezugs zu einem Überwachungsobjekt bei der integrierten Zustandsüberwachung berücksichtigt werden können (A2.1), müssen entsprechende Konzepte entwickelt werden.

A6.4 *Beantwortung unterschiedlicher topologiebezogener Anfragen*: In Infrastrukturnetzen treten topologiebezogene Anfragen in unterschiedlicher Form auf. Einige Beispiele wurden am Anfang dieses Kapitels gegeben, die vom Finden kürzester Pfade und nächster Nachbarn bis zu Bereichsanfragen reichen. Diese Aufzählung muss jedoch nicht vollständig sein. Außerdem können verschiedene Anfragetypen in unterschiedlichen Domänen einen unterschiedlichen Stellenwert besitzen. Um möglichst flexibel und erweiterbar zu sein, wird deshalb ein generischer Ansatz für Reasoning mit Topologiebezug gesucht (A2.2). Ziel ist es, Komponenten und Schnittstellen zu entwickeln, mit denen unterschiedlichste Graphenalgorithmen integriert werden können, um topologiebezogenes Reasoning auf Basis der bereits semantisch modellierten Kontextinformationen zu realisieren.

6.2. Verwandte Arbeiten

Vor diesem Hintergrund werden zunächst verwandte Arbeiten untersucht. Dazu werden Sensornetze und Sensor Webs, Netzinformationssysteme sowie Verkehrstelematik-Systeme betrachtet.

6.2.1. Sensornetze und Sensor Webs

Sensornetze und Sensor Webs wurden bereits in Abschnitt 4.2.1 eingeführt. Im Gegensatz zu der dort vorgestellten Modellierung der Sensormessungen steht hier die Verarbeitung der erfassten Daten im Vordergrund.

Sensornetze (engl. *wireless sensor networks*) werden eingesetzt, um ein bestimmtes Gebiet oder eine Umgebung zu überwachen. Sie bestehen aus einer Menge von Sensorknoten, die mit Sensorik, Energieversorgung, Rechen- und Speicherfähigkeit sowie einem Funkmodul ausgestattet sind, über das sie ad-hoc mit anderen Sensorknoten kommunizieren. Ziel ist es, die Sensorknoten so stark wie möglich zu verkleinern (*smart dust*) und das Netzwerk so lange wie möglich autonom und selbstorganisierend zu betreiben. Forschungsschwerpunkt auf diesem Gebiet ist deshalb, die Energieeffizienz dieser Netze zu maximieren, wozu z. B. optimierte Routingverfahren entwickelt werden.

Sensor Webs dienen im Wesentlichen ebenfalls der Überwachung von Umgebungen. Sie gehen darüber hinaus, indem sie die verschiedenen Sensorknoten (hier genannt *pods*) logisch gesehen als *ein* Messinstrument betrachten [Del04]: Sensorknoten verbreiten gemessene und verarbeitete Daten im gesamten Netz, wodurch jeder Sensorknoten immer über die Gesamtsituation im Sensor Web informiert ist. Damit kann sich ein

Sensor Web besser an neue Situationen anpassen und darauf reagieren als dies bei Sensornetzen der Fall ist [DJ01].

Im *SensorMap*-Projekt von *Microsoft* sollen Sensordaten räumlich integriert werden [NLZ07]. Die *SensorMap*-Webseite[1] betreibt jedoch keine eigene Sensorinfrastruktur. Stattdessen stellt sie ein Portal zur Verfügung, das Sensordaten *beliebiger* Sensorbetreiber in Echtzeit aggregiert und darstellt. Indem jeder in die Lage versetzt wird, seine Sensordaten zu veröffentlichen, soll eine flächendeckende Verfügbarkeit aktueller und verschiedenartigster Sensordaten erreicht werden. Beispiele für aktuell verfügbare Datenquellen sind Wetterstationen, Videokameras und Verkehrsdichtesensoren. Die Möglichkeiten zur Abfrage und Auswertung dieser Sensordaten sind jedoch sehr beschränkt: Sie können teilweise visualisiert werden. Zur Abfrage des aktuellen Messwerts eines Sensors, muss dieser jedoch explizit ausgewählt werden. Hier fehlt also die explizite Modellierung einer Topologie, mit der topologiebezogene Auswertungen möglich wären.

Es gibt einige Übersichtsarbeiten, die weitere Anwendungen für Sensornetze und Sensor Webs auflisten [ALM05, ASSC02, Del04]. Es zeigt sich, dass in der Regel ein räumliches Bezugssystem ausreicht, weil keine topologiebezogenen Anfragen beantwortet werden müssen. Vereinzelt werden zwar explizit modellierte Topologiemodelle verwendet, wie z. B. bei Anwendungen im Gebäude, die sichere Fluchtwege im Feuerfall ermitteln [BLA07]. Die Integration von Sensormessungen, wie z. B. der Rauchmelder, sind hier jedoch auf den Anwendungsfall zugeschnitten und können nicht ohne weiteres verallgemeinert werden.

6.2.2. Netzinformationssysteme

Netzinformationssysteme (NIS) werden eingesetzt, um Ver- und Entsorgungsnetze wie Strom und Wasser zu erfassen, zu dokumentieren, zu betreiben und zu analysieren. Sie beruhen im Wesentlichen auf *räumlichen Netzwerk- und Topologiedatenbanken* (engl. *spatial network databases,* SNDB). Diese sind ein wichtiger Spezialfall von Geodatenbanken, weil sie die Repräsentation der Netztopologie mit Lageinformationen kombinieren [Bri07].

Ein Datenmodell, das eine Netztopologie mit Lageinformationen kombiniert, wird auch *räumliches Netzwerk* (engl. *spatial network*) genannt. Während sich Geoobjekte in typischen geographischen Informationssystemen (GIS) an beliebigen Punkten in der euklidischen Ebene befinden, können sie hier nur auf dem Netz liegen. Während bei geographischen Informationssystemen die euklidische Distanz das wichtigste Maß darstellt, ist es bei räumlichen Netzwerkdatenbanken die *topologische* oder *Netzdistanz* [PZMT03]. Trotz dieser Besonderheiten beschränken sich viele geographische Informationssysteme und räumliche Datenbanken auf die rein geometrische Repräsentation und Anfragebearbeitung [PZMT03, Bri07]. Mit internationalen Standards wie *Spatial Schema* (ISO 19107:2003), *Simple Feature Access* (ISO 19125:2004) und SQL/MM *Spatial* (ISO/IEC 13249-3:2006) gibt es zunehmend die Möglichkeit, räumliche Netzwerke standardisiert zu repräsentieren. Die Unterstützung durch kommerzielle und Open-Source-Datenbanksysteme steht jedoch erst am Anfang. Während für viele Datenbanken Geodatenbank-Erweiterungen verfügbar sind, bieten bisher lediglich *Oracle*

[1] Microsoft Research SensorMap: http://atom.research.microsoft.com/sensewebv3/sensormap/

Spatial 10 und *PostGIS* (Geodatenerweiterung von PostgreSQL) lineare Bezugssysteme an [Bri07]. Die Einbindung dynamischer Daten, wie sie von Sensoren geliefert werden, ist dabei in der Regel nicht vorgesehen.

Zur Anfragebearbeitung in räumlichen Netzwerk- und Topologiedatenbanken haben beispielweise Papadis et al. ein umfassendes Rahmenwerk entwickelt [PZMT03]. Darin behandeln sie die Beantwortung von *Nächste-Nachbarn-*, *Bereichs-*, *Nächste-Paare-* und von *Entfernungsbezogenen-Join-*Anfragen. Diese sind analog zu den entsprechenden geometrischen Problemen definiert, beziehen sich jedoch auf die topologische Distanz in räumlichen Netzwerken. Arbeiten wie z. B. [KS04] bauen darauf auf und präsentieren alternative Algorithmen zur Beantwortung dieser Anfragen. Spezialisierte Arbeiten beschäftigen sich darüber hinaus mit der effizienten Repräsentation und Speicherung räumlicher Netzwerke [SSI+07, SAS05].

Anwendungen dieser Algorithmen finden sich beispielsweise in Straßennetzen, wo Fahrzeuge die mobilen Geoobjekte darstellen und nächste Nachbarn für sie gesucht werden [JKPT03]. Neuere Ansätze beschäftigen sich mit einer reichhaltigeren Annotation der zugrunde liegenden Netze mit Zusatzinformationen, z. B. zur besseren Navigation in Gebäuden [TA06]. Dazu werden häufig formale Ontologien eingesetzt. Deshalb wird auch zunehmend die generische Modellierung des Ortsbezugs untersucht [KS07a], beispielsweise die Modellierung der Geometrie mit *GeoRSS*[2] und der Topologie mit dem *Region Connection Calculus (*RCC*)* [KG05].

6.2.3. Verkehrstelematik

Verkehrstelematik (engl. *intelligent transportation systems*) bezeichnet Telematik-Anwendungen im Verkehr. Der Begriff Telematik ist wiederum zusammengesetzt aus „Telekommunikation“ und „Informatik“ und benennt damit Systeme, die Daten übertragen und verarbeiten. Mittels Verkehrstelematik sollen Verkehrsströme gezielt genutzt, organisiert und gelenkt werden, um effiziente, umweltfreundliche und zeitsparende Verkehrsinfrastrukturen zu realisieren.

Ein Beispiel ist das vom texanischen Verkehrsministerium entwickelte *Railtrac* System, das mit Hilfe mehrerer Radar-Sensoren die Positionen von Zügen überwacht [RB01]. Es ist als zentrales System konzipiert, das Sensordaten von den verteilten Radar-Stationen aggregiert. Da es genau auf diese Sensoren zugeschnitten ist, können keine weiteren Datenquellen eingebunden werden. Außerdem wird kein explizites Modell des zugrunde liegenden Schienennetzes verwendet, weil im realisierten Anwendungsfall nur ein (eindimensionaler) Korridor überwacht wird. Dadurch existieren jedoch auch keine allgemeingültigen Konzepte, mit denen dieser Ansatz auf netzartige Transportinfrastrukturen übertragbar ist.

Einen verteilten Ansatz für allgemeine Verkehrstelematik-Systemen stellen Dailey et al. vor [DHM96]. Hier werden vier Typen von Systemkomponenten unterschieden, die entweder Sensordaten produzieren, verteilen, verarbeiten oder konsumieren. Ihre Instantiierung und Verschaltung ist anwendungsspezifisch. Jedoch fehlt auch hier ein explizites Netzmodell, mit dem die generische Beantwortung netzbezogener Anfragen möglich wäre.

[2] GeoRSS – Geographically Encoded Objects for RSS feeds: http://georss.org/

Anforderung	Sensornetze und Webs	Netzinformationssysteme	Verkehrstelematik
A6.1 Domänen-Unabhängigkeit	−	+/−	−
A6.2 Föderierte Administration	−	−	−
A6.3 Ortsbezogene Kontextinformationen	+	−	+/−
A6.4 Unterschiedliche Anfragetypen	−	+	−

Tabelle 6.1.: Bewertung verwandter Arbeiten zum Reasoning mit Topologiebezug.

Das *Traffic Data Acquisition and Distribution* (*TDAD*) System integriert Sensordaten in Straßennetzen [DMPG02]: Im Großraum Seattle werden seit 1998 alle 20 Sekunden die Messungen von etwa 5000 Induktionsschleifen gesammelt und in einer Datenbank abgelegt. Dadurch wird eine Historie der Daten aufgebaut, die mittels *Data Mining*-Verfahren analysiert werden kann[3]. Auch hier wird jedoch auf eine explizite Modellierung des Straßennetzes verzichtet; topologiebezogene Anfragen können also nicht automatisiert beantwortet werden.

6.2.4. Zusammenfassende Bewertung

Tabelle 6.1 fasst noch einmal die Eigenschaften der vorgestellten Ansätze bezüglich der Anforderungen zusammen: Während sowohl Sensornetze als auch Verkehrstelematik-Systeme ihren Schwerpunkt auf die Erfassung und Bereitstellung von Sensordaten legen, bieten sie in der Regel keine Unterstützung für weitergehende Anfragen, die z. B. eine Topologie berücksichtigen. Für Netzwerkinformationssysteme und die zugrunde liegenden räumlichen Netzwerkdatenbanken stellen derartige Anfragen zwar die Kernfunktionalität dar – hier ist jedoch nicht vorgesehen, ortsbezogene und kontinuierlich anfallende Sensordaten mit der Topologie zu integrieren. Die föderierte Verwaltung und Administration eines Netzmodells wird darüber hinaus in keinem der genannten Bereiche betrachtet.

Zur Erfüllung aller Anforderungen, die für diese Arbeit relevant sind, müssen somit Konzepte aus den unterschiedlichen Gebieten kombiniert werden: Einerseits soll es möglich sein, ortsbezogene Kontextinformationen bezüglich eines Infrastrukturnetzes zu integrieren. Andererseits sollen topologiebezogene Anfragen bezüglich dieses Infrastrukturnetzes beantwortet und dabei die vorhandenen Kontextinformationen berücksichtigt werden. Als gemeinsame Grundlage für beide Aspekte wird im Folgenden zunächst eine geeignete Modellierung des Infrastrukturnetzes entwickelt.

[3]Das *Washington State Department of Transportation* stellt den vollständigen und aktuellen Datensatz unter http://www.its.washington.edu/tdad/ der Öffentlichkeit für eigene Analysen zur Verfügung.

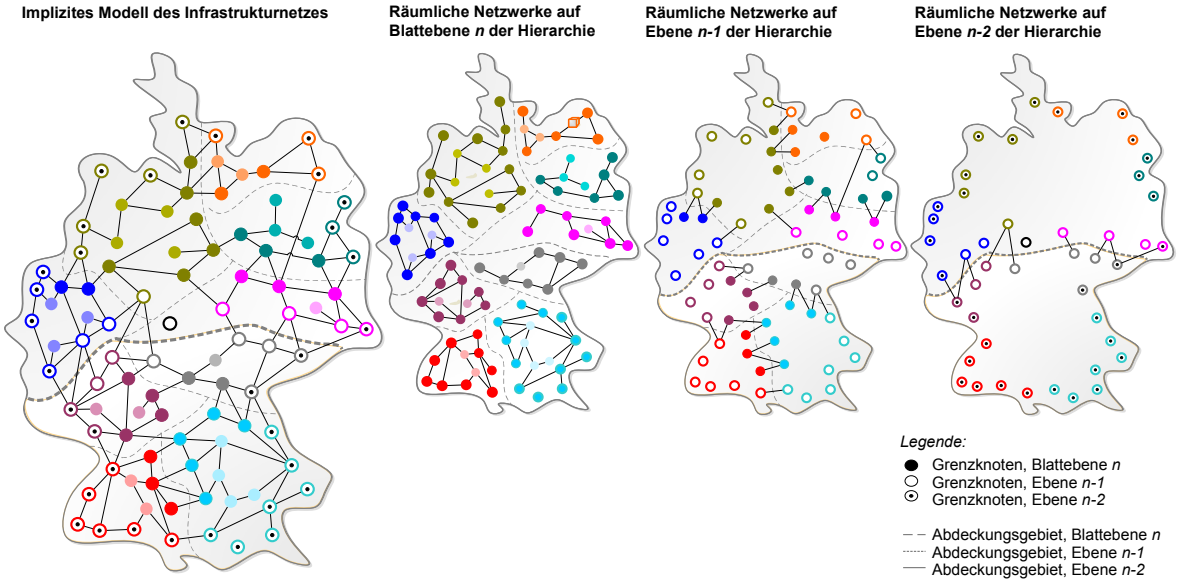

Abbildung 6.1.: Illustration der Modellierung eines Infrastrukturnetzes (nach [MFSW08]).

6.3. Modellierung des Infrastrukturnetzes

Um topologiebezogenes Reasoning betreiben zu können, wird zunächst ein geeignetes Modell des Infrastrukturnetzes benötigt. Dieser Abschnitt stellt einen entsprechenden Ansatz vor.

Ein Infrastrukturnetz wird als räumliches Netzwerk modelliert. Ein *räumliches Netzwerk* (engl. *spatial network*) ist eine Datenstruktur, die sowohl die Topologie als auch die Geometrie von Geoobjekten repräsentiert. Angewendet auf ein Infrastrukturnetz repräsentiert es sowohl die Topologie als auch die Geometrie der Elemente des Infrastrukturnetzes. Die Topologie wird durch einen gerichteten Graphen modelliert, dessen Knoten für logische Knoten im Infrastrukturnetz und dessen Kanten für logische Kanten im Infrastrukturnetz stehen. Die Geometrie wird wie folgt modelliert: Jeder Knoten ist mit Lagekoordinaten versehen und besitzt damit eine Referenz in ein räumliches Bezugssystem. Jedes räumliche Netzwerk besitzt ein Abdeckungsgebiet, das die Lagekoordinaten aller seiner Knoten enthält.

Das Modell des Infrastrukturnetzes soll föderiert administrierbar sein. Dies ist eine der Kernanforderungen an die Modellierung (Anforderung A6.1). Um dies zu erreichen, wird das Infrastrukturnetz als *Hierarchie von räumlichen Teil-Netzwerken* modelliert. Für die Hierarchie gilt: Jedes Teil-Netzwerk, das in der Hierarchie kein Blatt ist, wird von den Teil-Netzwerken auf der darunter liegenden Ebene *vollständig zerlegt* (engl. *jointly exhaustive, pairwise disjoint,* JEPD). Diese Eigenschaft ist in Abbildung 6.1 illustriert. Sie wird im Folgenden formalisiert.

Formalisierung der Hierarchie von räumlichen Netzwerken. Ein räumliches Teil-Netzwerk $G = (N, N^{G'}, E, \mu, A)$ wird definiert durch eine Knotenmenge $N \neq \emptyset$, eine gegenüber einem anderen Teil-Netzwerk G' veröffentlichte *Grenzknotenmenge*

$N^{G'} \subseteq N$, eine Kantenmenge $E \subseteq N \times N$, eine Abbildung $\mu : N \to \mathbb{R}^3$, die Knoten auf Lagekoordinaten abbildet, und ein Abdeckungsgebiet A mit $\mu(N) \subseteq A$.

Sei $\mathfrak{G}$ die Menge aller räumlichen Teil-Netzwerke. Diese sind wie folgt hierarchisch modelliert: Ein Knoten eines beliebigen räumlichen Teil-Netzwerkes in $\mathfrak{G}$ ist immer auch Knoten eines Teil-Netzwerkes auf Blattebene der Hierarchie. Teil-Netzwerke auf derselben Ebene können über sogenannte *Grenzknoten* miteinander verbunden sein. Solche Verbindungen sind jedoch nur auf höheren Ebenen bekannt. Die Grenzknoten aller Teil-Netzwerke einer Ebene sind deshalb die Knoten der Teil-Netzwerke der darüberliegenden Ebene. Auf diese Weise wird eine Hierarchie von Teil-Netzwerken über die jeweiligen *Grenzknoten* gebildet. Die Menge der Grenzknoten ist genau für das räumliche Teil-Netzwerk an der Wurzel der Hierarchie leer.

Eine derartige Modellierung hat folgende Vorteile: Da jedes räumliche Teil-Netzwerk nach außen hin nur seine Grenzknoten veröffentlicht, ist der Rest der Modellierung – d. h. der interne Aufbau des Netzwerks – im Teil-Netzwerk gekapselt. Eine Änderung des internen Aufbaus – verursacht beispielsweise durch Baumaßnahmen, die die Topologie verändern – ist nach außen hin nicht sichtbar. Sie hat deshalb auch keine Auswirkungen auf übergeordnete oder benachbarte Teil-Netzwerke in der Hierarchie. Dies gilt natürlich nur, sofern diese Änderung keine Grenzknoten betrifft. Dadurch wird besonders bei sehr großen Infrastrukturnetzen mit häufigen und gleichzeitigen Veränderungen die Konsistenzhaltung der Modellierung erleichert. In der Regel werden Teile des Infrastrukturnetzes von unterschiedlichen Organisationen betrieben. Falls jeder Teil als räumliches Teil-Netzwerk modelliert ist, kann eine Organisation also eigenständig Änderungen am internen Aufbau ihres Teils des Infrastrukturnetzes vornehmen, ohne sich mit den anderen Organisationen koordinieren zu müssen.

Für die Hierarchie von räumlichen Teil-Netzwerken wird eine zusätzliche Annahme eingeführt: Jedes Teil-Netzwerk wird von den Teil-Netzwerken auf der darunter liegenden Ebene *vollständig zerlegt* (engl. *jointly exhaustive, pairwise disjoint*).

Unter dieser Annahme können die Topologie des Infrastrukturnetzes effizienter verarbeitet und topologiebezogene Anfragen effizienter beantwortet werden. Dies wird in Abschnitt 6.4.2 zur Integration ortsbezogener Kontextinformationen und in Abschnitt 6.5.3 zur Beantwortung von Kontext-Filter-Anfragen ausgenutzt. Formal lässt sich eine derartige Hierarchie wie folgt definieren: Sei G_0 das räumliche Teil-Netzwerk an der Wurzel.

1. Für die Beziehung zwischen räumlichen Teil-Netzwerken auf *unterschiedlichen* Hierarchieebenen gilt: Sei $G_j = (N_j, N_j^{G_{j-1}}, E_j, \mu_j, A_j)$ ein räumliches Teil-Netzwerk auf Ebene $0 < j < n$. Dann gibt es auf Ebene $j+1$ räumliche Teil-Netzwerke $G_{j+1,i} = (N_{j+1,i},\ N_{j+1,i}^{G_j},\ E_{j+1,i},\ \mu_{j+1,i},\ A_{j+1,i}), 1 \leq i \leq k$ für ein $k \in \mathbb{N}$, für die gilt:

$$N_j = \bigcup_{i=1}^{k} N_{j+1,i}^{G_j} \qquad \text{(jointly exhaustive)}$$
$$A_j = \bigcup_{i=1}^{k} A_{j+1,i}.$$

2. Für die Beziehung zwischen räumlichen Teil-Netzwerken auf *derselben* Hierarchie-

ebene gilt: Seien $G_{j,i} = (N_{j,i}, N_{j,i}^{G_{j-1}}, E_{j,i}, \mu_{j,i}, A_{j,i})$ für $1 \leq i \leq m_j$ alle räumlichen Teil-Netzwerke auf Ebene j mit $0 < j \leq n$. Dann gilt:

$$N_{j,i} \cap N_{j,i'} = \emptyset \quad \text{für alle } 1 \leq i \neq i' \leq m_j \qquad \text{(pairwise disjoint)}$$
$$A_{j,i} \cap A_{j,i'} = \emptyset \quad \text{für alle } 1 \leq i \neq i' \leq m_j.$$

Beispiele für die Modellierung von Schienen- und Stromnetzen. Mit dieser Modellierung lassen sich Infrastrukturnetze wie folgt repräsentieren: Beim Schienennetz werden Gleise als gerichtete Kanten und Haltepunkte und Weichen als Knoten in räumlichen Teil-Netzwerken modelliert. Die Hierarchiebildung kann sich aus der Unterteilung in Regionen ergeben. Im deutschen Schienennetz (siehe Abbildung 6.2) gibt es z. B. folgende Regionen, für die unterschiedliche Organisationen zuständig sind: *Nord*, *Nordost*, *Nordrhein-Westfalen*, *Südost*, *Hessen*, *Südwest*, *RheinNeckar*, *Baden-Württemberg* und *Bayern*. Jede dieser Regionen wird durch ein räumliches Teil-Netzwerk auf Blattebene der Hierarchie repräsentiert. Seien z. B. G_{Bayern} und $G_{\text{Baden-W.}}$ die räumlichen Teil-Netzwerke für Bayern und Baden-Württemberg. Diese und die räumlichen Teil-Netzwerke für die anderen Regionen teilen ihre Grenzknoten, d. h. Knoten, die eine direkte Verbindung zu einem Knoten in einer *anderen* Region besitzen, der bundesweit zuständigen Stelle mit. Dort entsteht ein räumliches Teil-Netzwerk $G_{\text{Deutschland}}$, dessen Knotenmenge genau den Grenzknoten der verschiedenen Regionen entspricht und dessen Kantenmenge zusätzlich die möglichen Übergänge zwischen diesen Regionen repräsentiert. Diese Strukturierung lässt sich auf europäischer Ebene weiterführen: So könnte ein räumliches Teil-Netzwerk für Zentraleuropa die grenzüberschreitenden Verknüpfungen der Teil-Netzwerke $G_{\text{Deutschland}}$ und $G_{\text{Frankreich}}$ modellieren und selbst über ein gesamteuropäisches Teil-Netzwerk mit den anderen Teilen Europas verbunden sein.

Auch für Stromnetze kann diese Modellierung angewendet werden: Hier entsprechen die Kanten den Leitungen und die Knoten den Masten, Umspannstationen, Erzeugern und Verbrauchern. Dabei kann die technisch bedingte und auch organisatorisch widergespiegelte Hierarchiebildung in Übertragungs- und Verteilnetze (vergleiche Abschnitt 2.2) durch entsprechende Hierarchiebildung in der Modellierung repräsentiert werden. Als Beispiel ist das deutsche Verbundnetz in Abbildung 6.3 dargestellt.

6.4. Netzbasierte Integration ortsbezogener Kontextinformationen

Das Infrastrukturnetz und seine Elemente stellen Überwachungsobjekte im Sinne von Kapitel 4 dar. Dieser Abschnitt stellt ein generisches Konzept zur Integration ortsbezogener Kontextinformationen bezüglich dieser Überwachungsobjekte vor. Wie oben beschrieben, können sowohl Kontextinformationen als auch Überwachungsobjekte als Geoobjekte angesehen werden. Bei beiden ist die geometrische Komponente bekannt, d. h. die Lage in einem räumlichen Bezugssystem ist gegeben. Bei Überwachungsobjekten ist durch das im vorigen Abschnitt beschriebene Modell des Infrastrukturnetzes zusätzlich die topologische Komponente, also die relative Lage zu anderen Überwa-

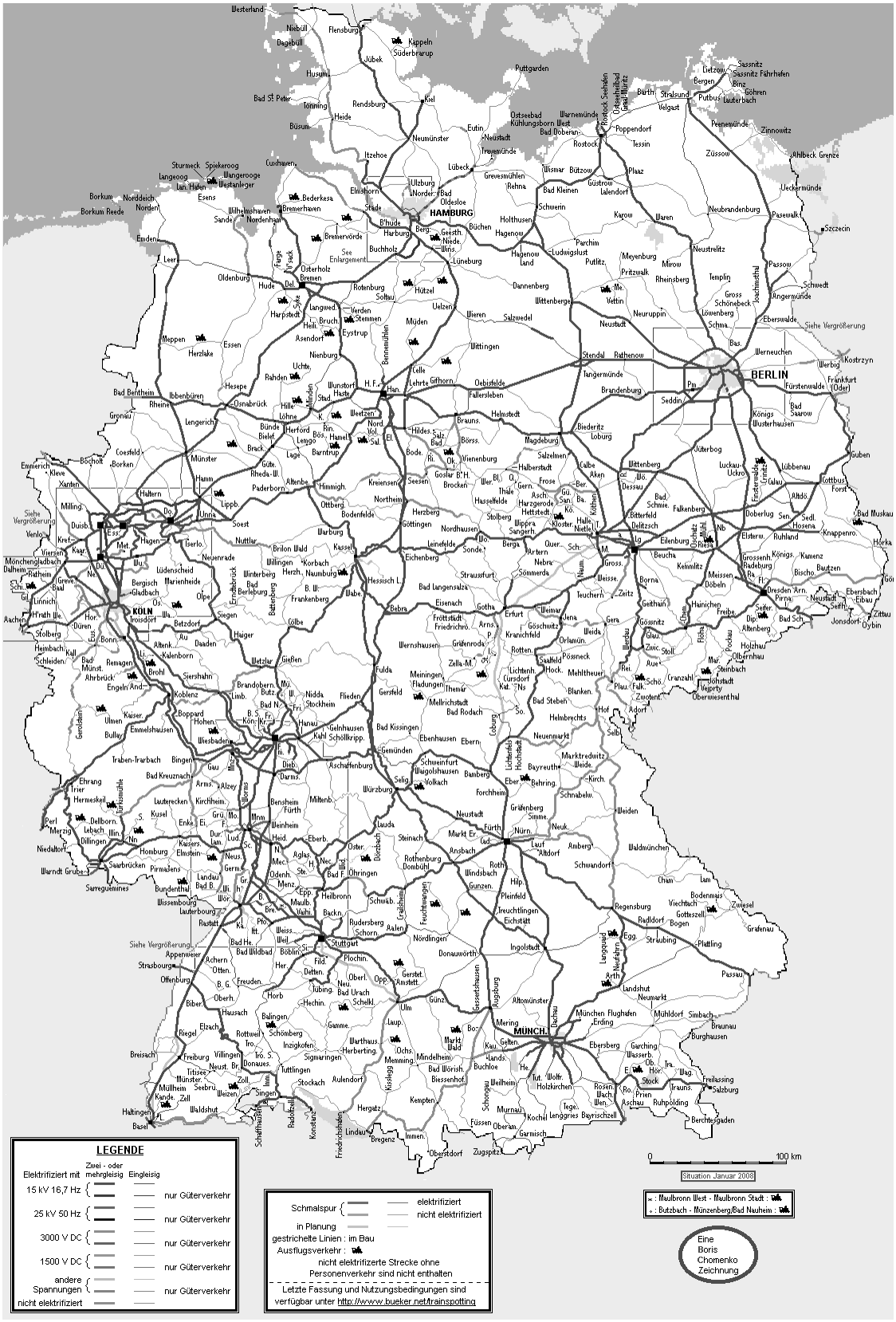

Abbildung 6.2.: Deutsches Schienennetz (Stand: 1. Januar 2008, nach [Bük08]).

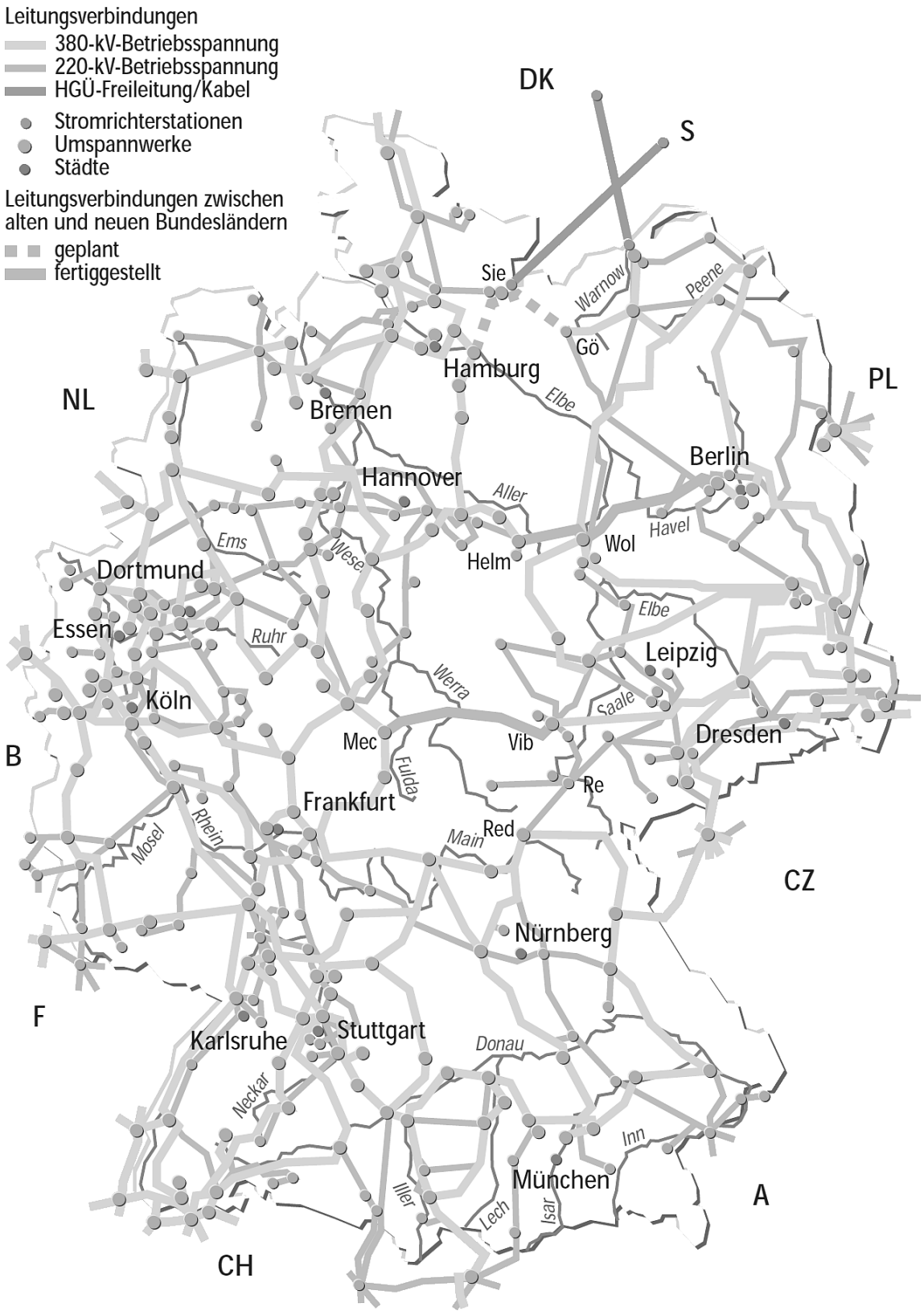

Abbildung 6.3.: Deutsches Stromverbundnetz (Stand: 1. Januar 1996, nach [Sch05]).

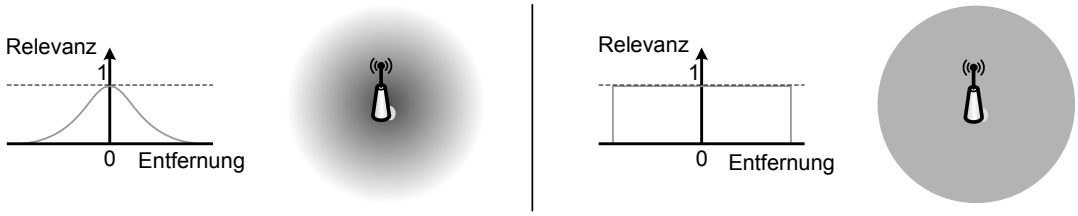

Abbildung 6.4.: Beispiele für kontexttypspezifische Wirkfunktionen: Kontexttyp Temperatur (links), Kontexttyp Niederschlag (rechts) (nach [MFSW08]).

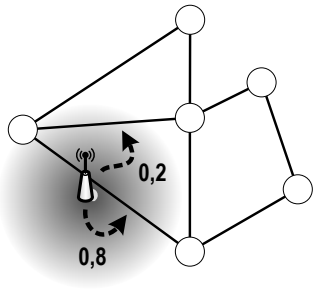

Abbildung 6.5.: Zuordnung einer Kontextinformation zu Infrastrukturelementen [Mie07].

chungsobjekten, gegeben. Dazu gehört beispielsweise, welche Gleise sich in welcher Weiche treffen und welche Stromleitungen welches Umspannwerk verlassen.

Nicht explizit bekannt ist jedoch die Topologie von *ortsbezogenen Kontextinformationen* und *Überwachungsobjekten*, d. h. die relative Lage einer ortsbezogenen Kontextinformation zu einem Überwachungsobjekt. Um die Relevanz einer Kontextinformation für den Zustand eines Infrastrukturelements beurteilen zu können, muss diese Beziehung ermittelt werden. Dazu werden zunächst Wirkfunktionen zur Relevanzbestimmung eingeführt und anschließend ihre Anwendung zur Zuordnung ortsbezogener Kontextinformationen zu Überwachungsobjekten beschrieben.

6.4.1. Wirkfunktionen zur räumlichen Relevanzbestimmung von Kontextinformationen

Die Elemente des Infrastrukturnetzes sind wichtige Überwachungsobjekte. Wie vorher beschrieben, ist das Infrastrukturnetz als Hierarchie von räumlichen Teil-Netzwerken modelliert. Die Relevanz einer bestimmten Kontextinformation für ein Element des Infrastrukturnetzes hängt dabei (i) von der Entfernung d der Kontextinformation zum Infrastrukturelement und (ii) vom *Kontexttyp* κ ab. So sind von einer Kontextinformation des Kontexttyps „Niederschlag“ typischerweise Infrastrukturelemente in einem größeren Umkreis betroffen, während eine Kontextinformation vom Kontexttyp „Blitzschlag“ in der Regel nur ein eng begrenztes Gebiet betrifft. Zur generischen Modellierung dieser beiden Aspekte schlägt die Arbeit *Wirkfunktionen* vor, die spezifisch für einen bestimmten Kontexttyp sind.

Definition 6.4.1 (Wirkfunktion ι^κ). *Sei κ ein Kontexttyp und d eine euklidische Entfernung. Dann ist die Wirkfunktion ι^κ für κ definiert als:*

$$\iota^\kappa : \mathbb{R} \rightarrow [0, 1].$$

$\iota^\kappa(d)$ gibt auf einer Skala von 0 bis 1 an, welche Relevanz eine Kontextinformation vom Typ κ in einer Entfernung d von ihren Lagekoordinaten hat.

Beispiele für Wirkfunktionen für zwei unterschiedliche Kontexttypen sind in Abbildung 6.4 dargestellt: Die x-Achse gibt die euklidische Distanz an, während die y-Achse den Grad der Relevanz in dem Intervall von 0 bis 1 darstellt. Eine Wirkfunktion kodiert also für einen bestimmten Kontexttyp seine Relevanz für die Zustandsüberwachung in Abhängigkeit von der Entfernung.

Sind eine Kontextinformation mit typspezifischer Wirkfunktion und die Geometrie eines Infrastrukturelements gegeben, kann somit ermittelt werden, zu welchem Grad die Kontextinformation für das Infrastrukturelement relevant ist (siehe Illustration in Abbildung 6.5).

6.4.2. Zuordnung einer Kontextinformation zu Infrastrukturelementen

Um den Zustand eines Infrastrukturelements überwachen zu können, müssen alle verfügbaren Kontextinformationen zu diesem Infrastrukturelement integriert werden. Eine *ortsbezogene* Kontextinformation muss dazu den Infrastrukturelementen zugeordnet werden, für die sie relevant ist. Eine ortsbezogene Kontextinformation ist genau dann für ein Infrastrukturelement relevant, wenn ihre Wirkfunktion für die Entfernung zwischen Lagekoordinate der Kontextinformation und Infrastrukturelement eine gewisse Mindestrelevanz ergibt. Es muss also ein Verfahren entwickelt werden, das für eine gegebene ortsbezogene Kontextinformation *alle* Infrastrukturelemente identifiziert, für die sie relevant ist.

Ein naiver Ansatz zur Bestimmung dieser Infrastrukturelemente besteht darin, jeweils die Entfernung der Kontextinformation zu jedem Infrastrukturelement zu bestimmen, die Wirkfunktion anzuwenden und die entsprechend gewichtete Kontextinformation dem Element zuzuordnen. Für reale Infrastrukturnetze ist das aufgrund ihrer Größe jedoch nicht praktikabel.

Ein effizienterer Ansatz ist es, *räumliche* bzw. *mehrdimensionale Indexstrukturen* einzusetzen. Diese wurden zu dem Zweck entwickelt, aus einer großen Zahl räumlicher Objekte diejenigen mit einer bestimmten räumlichen Eigenschaft effizient zu ermitteln [Sam06b].

Für die Zuordnung einer ortsbezogenen Kontextinformation zu allen Infrastrukturelementen, für die sie relevant ist, sind folgende Überlegungen nötig: Das Infrastrukturnetz ist als Hierarchie von räumlichen Teil-Netzwerken modelliert. Die Struktur seiner Hierarchie muss in der Regel als gegeben angesehen werden, weil sie die Zuständigkeiten von Organisationen abbildet. Räumliche Indexstrukturen nutzen üblicherweise zwar auch Hierarchien. Deren Struktur kann in der Regel jedoch nicht vorgegeben werden, sondern wird durch die zu indizierenden Daten impliziert. Deshalb können räumliche

Indexstrukturen nicht direkt für die Suche nach den Infrastrukturelementen, für die die Kontextinformation relevant ist, eingesetzt werden. Stattdessen schlägt diese Arbeit einen zweistufigen Ansatz vor:

1. Zunächst werden die räumlichen Teil-Netzwerke in der Hierarchie bestimmt, für deren Elemente die Kontextinformation potentiell relevant ist. Dazu wird ausgenutzt, dass jedes räumliche Teil-Netzwerk G_j auf Ebene j durch räumliche Teil-Netzwerke $G_{j+1,i}$ auf Ebene $j+1$ vollständig zerlegt wird (vergleiche Abschnitt 6.3):

   Sei die Kontextinformation vom Typ κ. Dann lässt sich mit der Wirkfunktion dieses Kontexttyps ι^κ ein Relevanzgebiet A_{ι^κ} ableiten. Dieses entspricht einem Kreis mit der Lagekoordinate der Kontextinformation als Mittelpunkt und dem Radius $\max\{d \,|\, \iota^\kappa(d) > r\}$ für eine Mindestrelevanz $r \in [0,1]$. Dann kann geprüft werden: Überlappen sich dieses Relevanzgebiet und das Abdeckungsgebiet A_j von G_j ($A_{\iota^\kappa} \cap A_j \neq \emptyset$), so wird für alle $G_{j+1,i}$ (und rekursiv die Teil-Netzwerke auf der darunter liegenden Ebene bis hin zur Blattebene) geprüft, ob $A_{\iota^\kappa} \cap A_{j+1,i} \neq \emptyset$. Überlappen sich Relevanzgebiet und Abdeckungsgebiet A_j nicht, wird diese Prüfung für das räumliche Teil-Netzwerk auf Ebene $j-1$ vorgenommen.

2. Wurde in Schritt 1 ein $G_{j+1,i} = (N_{j+1,i},\ N_{j+1,i}^{G_j},\ E_{j+1,i},\ \mu_{j+1,i},\ A_{j+1,i})$ mit $A_{\iota^\kappa} \cap\ A_{j+1,i} \neq \emptyset$ identifiziert, werden nun die Elemente in $G_{j+1,i}$ bestimmt, für die die Kontextinformation relevant ist. Dazu muss für alle Knoten $N_{j+1,i}$ und Kanten $E_{j+1,i}$ geprüft werden, ob die Kontextinformation für sie relevant ist. Da diese Prüfung innerhalb eines räumlichen Teil-Netzwerkes vorgenommen wird, können nun klassische räumliche Indexstrukturen eingesetzt werden. Die Auswahl einer bestimmten Indexstruktur, wie zum Beispiel eines *R-Baums*, ist eine Implementierungsentscheidung. Die Eigenschaften der unterschiedlichen Indexstrukturen werden beispielsweise ausführlich in [Sam06b] behandelt. Die Entfernung zwischen der Kontextinformation und einem Knoten ist dabei definiert als die euklidische Distanz; die Entfernung zwischen der Kontextinformation und einer Kante als die lotrechte (d. h. kürzeste) euklidische Distanz.

Für eine gegebene Kontextinformation ist das Ergebnis dieses Verfahrens die Menge der Knoten und Kanten, für die sie relevant ist, sowie jeweils der Grad der Relevanz, der auf der Skala von 0 bis 1 quantifiziert wird. Die Knoten und Kanten sind Infrastrukturelemente und stellen Überwachungsobjekte im Sinne der Zustandsüberwachung dar. Auf Basis dieser Informationen können nun passende Merkmals-Zusicherungen bezüglich der Überwachungsobjekte abgeleitet werden und einer Wissensbasis entsprechend der in Kapitel 4 entwickelten Modellierung hinzugefügt werden.

6.5. Beantwortung topologiebezogener Anfragen

Die in Kapitel 5 vorgestellten Verfahren ermöglichen es, für ein oder mehrere gegebene Überwachungsobjekte, deren Beziehung mittels Rollen ausgedrückt werden kann, den aktuellen technischen Zustand abzuleiten. Für viele topologiebezogene Anfragen reicht dies jedoch nicht aus:

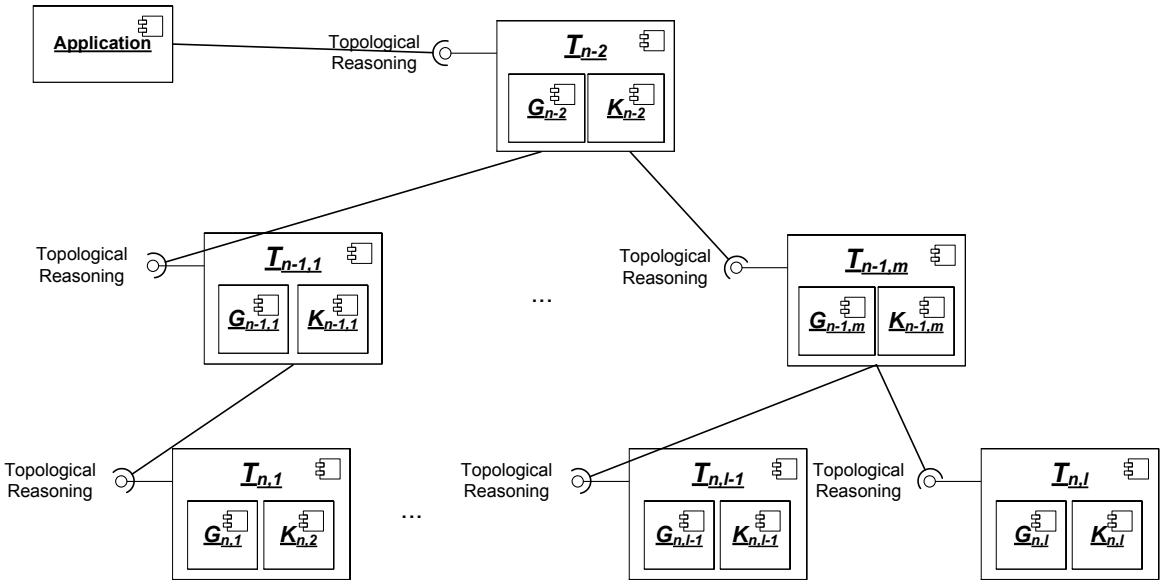

Abbildung 6.6.: Konzeptionelles topologiebezogenes Reasoning-System mit topologiebezogenen Reasoning-Knoten T_j, denen jeweils ein räumliches Teil-Netzwerk G_j und eine Wissensbasis K_j mit Kontextinformationen zu den Kanten und Knoten in G_j zugeordnet ist.

- Viele topologiebezogene Anfragen können mit konjunktiven Anfragen nicht ausgedrückt werden, weil die Beziehungen zwischen den gesuchten Überwachungsobjekten im Vorhinein nicht bekannt sind. Wie zu Beginn dieses Kapitels erläutert, zählt dazu beispielsweise die Suche nach einem optimalen Pfad, da dessen Länge im Vorhinein nicht bekannt ist.

- Besitzt die Anfrage einen Topologiebezug, kann das Wissen über die Topologie ausgenutzt werden, um eine effiziente Anfragebearbeitung zu erreichen. Sind beispielsweise Start- und Endknoten gegeben, müssen bei der Suche nach einem optimalen Pfad nur Kanten berücksichtigt werden, die Teil eines Pfades zwischen diesen Knoten sind. Unter Ausnutzung von Erreichbarkeits- und Lageinformationen kann der Suchraum also effektiv verkleinert werden. Sind die Kontextinformationen verteilt, kann damit auch die Zahl und Größe der zu berücksichtigenden Teil-Wissensbasen verringert werden. Dies wurde bereits in Kapitel 5 als effektiver Weg identifiziert, um die Effizienz der Anfragebeantwortung zu erhöhen.

Als konzeptionelle Grundlage für die Beantwortung topologiebezogener Anfragen wird ein *topologiebezogenes Reasoning-System* eingeführt. Es besteht aus einer Menge von *topologiebezogenen Reasoning-Knoten* T_j. Jedem topologiebezogenen Reasoning-Knoten T_j ist ein räumliches Teil-Netzwerk G_j zugeordnet sowie die Wissensbasis K_j, die Kontextinformationen zu diesem Teil-Netzwerk enthält (vergleiche Illustration in Anlehnung an UML-Komponentendiagramme in Abbildung 6.6). Damit abstrahiert ein topologiebezogener Reasoning-Knoten eine Organisation, die einen Teil des Infrastrukturnetzes verwaltet und außerdem Zugriff auf die zugeordneten Kontextinformationen

hat. Die topologiebezogenen Reasoning-Knoten T_j sind entsprechend der Hierarchie der räumlichen Teil-Netzwerke angeordnet. Ein topologiebezogener Reasoning-Knoten nimmt topologiebezogene Anfragen an und kann diese für das zugeordnete räumliche Teil-Netzwerk und die entsprechenden Kontextinformationen beantworten. Müssen darüber hinaus andere räumliche Teil-Netzwerke in der Hierarchie berücksichtigt werden, delegiert er die Anfrage an seinen direkten Vorgänger oder einen seiner direkten Nachfolger in der Hierarchie.

Sei ein topologiebezogenes Reasoning-System gegeben. Auf dessen Basis sollen im Folgenden Verfahren zur Beantwortung unterschiedlicher topologiebezogener Anfragen entwickelt werden. Ziel ist es, *atomare* Anfragetypen zu identifizieren und im topologiebezogenen Reasoning-System umzusetzen, auf denen Verfahren zur Beantwortung *komplexer* Anfragetypen aufbauen können. Dabei sollen auch bestehende Graphenalgorithmen übertragbar sein.

Dieser Abschnitt ist wie folgt aufgebaut: Zunächst werden die atomaren Anfragetypen *Teil-Netzwerk-Identifikation* und *Kontext-Abfrage* in den Abschnitten 6.5.1 und 6.5.2 beschrieben. Jeweils darauf aufbauend werden dann folgende komplexe Anfragetypen im topologiebezogenen Reasoning-System umgesetzt: *Kontext-Filter-* (Abschnitt 6.5.3), *Optimale-Pfade-* (Abschnitt 6.5.4), *Nächste-Nachbarn-* (Abschnitt 6.5.5) und *Bereichs-*Anfragen (Abschnitt 6.5.6).

Damit wird demonstriert, wie in einem topologiebezogenen Reasoning-System sowohl bestehende als auch neu entwickelte Verfahren für topologiebezogene Anfragetypen umgesetzt werden. Dadurch können auf Basis des einmal modellierten Infrastrukturnetzes und der einmal semantisch repräsentierten Kontextinformationen unterschiedlichste Reasoning-Verfahren realisiert werden.

6.5.1. Atomare Teil-Netzwerk-Identifikation für ein Infrastrukturelement

Ein atomarer Anfragetyp des Systems für topologiebezogenes Reasoning ist die Teil-Netzwerk-Identifikation: Ist ein Infrastrukturelement gegeben, so soll der topologiebezogene Reasoning-Knoten zurückgeliefert werden, der das (eindeutige) räumliche Teil-Netzwerk verwaltet, welches dieses Element enthält und auf Blattebene der Hierarchie ist. Dazu muss die Hierarchie der räumlichen Teil-Netzwerke durchlaufen werden.

Zur Traversierung der Hierarchie können unterschiedliche Strategien eingesetzt werden. Bei der Zustandsüberwachung von Infrastrukturnetzen ist es sinnvoll, das sogenannte *Lokalitätsprinzip* anzuwenden, wie es auch bei anderen großflächigen Lokalisierungssystemen genutzt wird [SHHT98]. Es beruht auf der Annahme, dass der Absender einer Lokalisierungsanfrage sich häufig räumlich nahe dem gesuchten Objekt befindet und die Lokalisierung räumlich naher Objekte deshalb besonders effizient möglich sein soll.

Angewendet auf die hier relevante Hierarchie von räumlichen Teil-Netzwerken bedeutet dies: Sei T_j der topologiebezogene Reasoning-Knoten auf Ebene j, der die Anfrage erhält. Zunächst wird für das räumliche Teil-Netzwerk G_j geprüft, ob es das Infrastrukturelement enthält (d. h. ob das Element ein Knoten in N_j oder eine Kante in E_j ist). Falls nicht, wird die Anfrage zuerst an die topologiebezogenen Reasoning-Knoten auf

den *darunter liegenden* Ebenen $i > j$ propagiert. Erst wenn das Infrastrukturelement auch dort nicht lokalisiert werden kann, wird die Anfrage an den topologiebezogenen Reasoning-Knoten auf Ebene $j-1$ und damit an den Rest der Hierarchie weitergeleitet.

Da anzunehmen ist, dass die Hierarchie von räumlichen Teil-Netzwerken (und damit topologiebezogenen Reasoning-Knoten) relativ statisch ist, kann die Effizienz der Anfragebearbeitung außerdem durch *Caching* erhöht werden: Wurde ein Infrastrukturelement lokalisiert, werden bei der Rückgabe der Antwort zum Ausgangs-Reasoning-Knoten T_j in jedem topologiebezogenen Reasoning-Knoten auf dem Pfad durch die Hierarchie der Bezeichner des Elements sowie der Pfad-Nachfolger gespeichert. Erhält ein beliebiger topologiebezogener Reasoning-Knoten auf diesem Pfad anschließend erneut eine Anfrage nach diesem Infrastrukturelement, kann die Anfrage gezielt an den zuständigen Reasoning-Knoten weitergeleitet werden.

6.5.2. Atomare Kontext-Abfrage für ein Infrastrukturelement

Ein weiterer atomarer Anfragetyp des Systems für topologiebezogenes Reasoning ist die Kontext-Abfrage: Seien ein Infrastrukturelement und ein Kontexttyp gegeben, so soll die aktuelle Kontextinformation dieses Typs zurückgeliefert werden.

Zur Beantwortung einer atomaren Kontext-Abfrage muss also zunächst der topologiebezogene Reasoning-Knoten identifiziert werden, dessen räumliches Teil-Netzwerk für das gegebene Überwachungsobjekt zuständig ist. Dazu wird eine *atomare Teil-Netzwerk-Identifikation* durchgeführt. Ist der topologiebezogene Reasoning-Knoten mit dem zuständigen Teil-Netzwerk gefunden, nutzt er die zugeordnete Wissensbasis, um die aktuellen Kontextinformationen zu dem gegebenen Element und dem gegebenen Kontexttyp abzurufen und zurückzugegeben.

6.5.3. Kontext-Filter-Anfragen bezüglich aller Infrastrukturelemente

Ein erster komplexer Anfragetyp, der mit Hilfe der atomaren Anfragetypen in einem topologiebezogenen Reasoning-System umgesetzt werden kann, ist die Kontext-Filter-Anfrage: Sind ein geometrisches Gebiet und eine kontextbezogene Filterbedingung gegeben, so sollen alle Infrastrukturelemente geliefert werden, die in dem geometrischen Gebiet liegen und die Filterbedingung erfüllen. Ein Beispiel ist die Anfrage nach allen Überlandleitungen im Landkreis Fürstenfeldbruck, deren Abnutzung einen kritischen Grad erreicht hat.

Definition 6.5.1 (Kontext-Filter-Anfrage). *Sei $\mathfrak{G}$ die Menge der räumlichen Teil-Netzwerke in der Hierarchie, A ein geometrisches Anfragegebiet und C eine Kontextbedingung. Dann findet eine* Kontext-Filter-Anfrage *alle Knoten n und Kanten e in den räumlichen Teil-Netzwerken in $\mathfrak{G}$, die in A liegen und für die $C(n)$ bzw. $C(e)$ gilt.*

Um die Effizienz der Suche zu erhöhen, werden zunächst die räumlichen Teil-Netzwerke ermittelt, die die geometrische Bedingung erfüllen. Dazu wird analog zu Abschnitt 6.4.2 die Eigenschaft der vollständigen Zerlegung ausgenutzt:

Sei T_j der topologiebezogene Reasoning-Knoten auf Ebene j, der die Anfrage erhält, und A das Anfragegebiet. Ist die Schnittmenge des Abdeckungsgebiets A_j des zugeordneten räumlichen Teil-Netzwerks G_j mit dem Anfragegebiet leer oder das Anfragegebiet nur teilweise im Abdeckungsgebiet enthalten ($A \not\subset A_j$), wird die Anfrage rekursiv solange an den topologiebezogenen Reasoning-Knoten auf Ebene $i < j$ weitergeleitet, bis A vollständig in dem Abdeckungsgebiet des zugeordneten räumlichen Teil-Netzwerks enthalten ist. Dann wird die Anfrage an die Reasoning-Knoten auf der darunter liegenden Ebene gestellt. Falls deren Abdeckungsgebiet mit dem Anfragegebiet überlappt, wird für jeden Knoten und jede Kante des zugeordneten Teil-Netzwerks geprüft, ob sowohl die geometrische Bedingung als auch die kontextbezogene Filterbedingung erfüllt sind. Dabei wird die Prüfung der Filterbedingung auf eine *atomare Kontext-Abfrage* zurückgeführt. Dies wird rekursiv solange wiederholt, bis die Blattebene erreicht ist. Dann wurden alle Knoten und Kanten der Lösung gefunden.

6.5.4. Optimale-Pfade-Anfragen zwischen zwei Infrastrukturelementen

Ein weiterer Anfragetyp, der auf Basis der atomaren Anfragetypen in einem topologiebezogenen Reasoning-System umgesetzt werden kann, ist die Suche nach optimalen Pfaden im gesamten Infrastrukturnetz. Der zu optimierende Kontexttyp kann sich dabei sowohl auf *statische* Kontexttypen, wie z. B. Länge oder Elektrifizierung eines Gleisabschnitts, als auch auf *dynamische* Kontexttypen beziehen. Ein Beispiel für dynamischen Kontext ist das Verspätungsrisiko, das beispielsweise bei der Anfrage nach einer Verbindung für einen wichtigen Gütertransport von Hamburg nach Rotterdam minimiert werden soll.

Definition 6.5.2 (Optimale-Pfade-Anfrage). *Sei s ein Startknoten, t ein Zielknoten und $O^\kappa = \{min^\kappa, max^\kappa\}$ ein Optimalitätskriterium bezüglich des Kontexttyps κ. Sei $\delta^\kappa(p)$ eine Funktion, die für einen Pfad $p = (n_1, \ldots, n_k)$ mit Knoten $n_1, \ldots, n_k$ den Kontextwert bezüglich κ angibt. Dann findet eine* Optimale-Pfade-Anfrage *einen Pfad $p^{O^\kappa}_{s \to t} = (s, n_1, \ldots, n_k, t)$ mit $\delta^\kappa(p^{O^\kappa}_{s \to t}) = O^\kappa(\delta^\kappa(P))$ für die Menge P aller Pfade von s nach t.*

Das topologiebezogene Reasoning-System soll es ermöglichen, bestehende Graphenalgorithmen auf Grundlage der atomaren Anfragetypen umzusetzen: Für die Suche nach optimalen und fast-optimalen Pfaden in hierarchischen Graphen wurden mehrere Algorithmen vorgeschlagen [HMZM96, HDP+94, YJYQ03, HJR95]. Die Flexibilität des vorgeschlagenen Systems wird demonstriert, indem im Folgenden exemplarisch sowohl ein Verfahren zum Finden garantiert optimaler Pfade als auch ein Verfahren zum Finden fast-optimaler Pfade umgesetzt werden.

6.5.4.1. Garantiert optimale Pfade

Jing et al. stellen einen Ansatz zum Finden garantiert optimaler Pfade in zweistufigen hierarchischen Graphen vor [JHR98]. Sie verfolgen das Ziel, für einen gegebenen *nicht hierarchischen* Graphen durch Partitionierung und (zweistufige) Hierarchie-Bildung die Effizienz bei der Suche nach optimalen Pfaden zu erhöhen. Damit unterscheidet sich

ihr Ansatz wesentlich von den hier identifizierten Anforderungen, weil in dieser Arbeit davon ausgegangen werden muss, dass Partitionierung und Hierarchie des Graphen gegeben sind. Dennoch lässt sich das grundsätzliche Konzept von Jing et al. übernehmen. Es muss jedoch zusätzlich verallgemeinert werden, um für Hierarchien beliebiger Höhe eingesetzt werden zu können.

Das verallgemeinerte Verfahren beruht auf einer Vorverarbeitung des Graphen und garantiert die Optimalität des gefundenen Pfades. Es umfasst zwei grundlegende Schritte:

1. Berechnung einer *hierarchisch kodierten Pfad-Sicht* (engl. *hierarchically encoded path view,* HEPV(G)) für jedes räumliche Teil-Netzwerk G in der Hierarchie. Aus HEPV(G) kann für jedes Knotenpaar in $N \times N$ direkt der optimale Pfad sowie sein Gewicht abgelesen werden.

2. Suche nach einem optimalen Pfad durch die räumlichen Teil-Netzwerke in der Hierarchie mit Hilfe der hierarchisch kodierten Pfad-Sichten.

Diese Schritte werden im Folgenden im Detail beschrieben.

Berechnung einer hierarchisch kodierten Pfad-Sicht. Sei die hierarchisch kodierte Pfad-Sicht für das räumliche Teil-Netzwerk $G = (N, N^{G'}, E, \mu, A)$ definiert als HEPV$(G) = (N, E^+)$ mit $E^+ \subseteq N \times N \times L$. Dabei ist L eine Menge von Kennzeichnungen $L(n_1, n_2) = (G_{n_1,n_2}, w_{n_1,n_2}, \vec{n}_{n_1,n_2}) \in \mathfrak{G} \times \mathbb{R}_{\geq 0} \times N$. Hier bezeichnet G_{n_1,n_2} das – ausgehend von der Blattebene der Hierarchie – erste räumliche Teil-Netzwerk, welches mit seinen Unter-Netzwerken alle Knoten und Kanten des optimalen Pfades von n_1 nach n_2 enthält. w_{n_1,n_2} ist das Gewicht dieses Pfades. $\vec{n}_{n_1,n_2}$ gibt den nächsten (d. h. zweiten) Knoten auf diesem Pfad an. Ein Beispiel für ein HEPV ist in Tabelle 6.2 dargestellt. Es kann auch als Graph visualisiert werden (siehe die rechte Seite von Abbildung 6.7).

Für die Generierung von HEPV(G_j) für ein Teil-Netzwerk G_j muss nach der Hierarchieebene, auf der sich G_j befindet, differenziert werden:

- Sei G_j auf der Blattebene der Hierarchie, d. h. $j = n$. Dann entspricht HEPV(G_j) der transitiven Hülle von G_j mit L definiert durch den optimalen Pfad von n_1 nach n_2 in G_j für alle Knotenpaare $(n_1, n_2) \in N_j \times N_j$ und dem Gewicht w_{n_1,n_2} gleich dem Wert des zu optimierenden Kontexttyps. Dieser wird wieder mittels *atomarer Kontext-Abfragen* ermittelt. Zur Berechnung der optimalen Pfade zwischen allen Paaren von Knoten kann ein beliebiger *all pairs shortest path*-Algorithmus (APSP-Algorithmus, analog für längste Pfade) eingesetzt werden (siehe Abbildung 6.7 unten rechts).

- Sei G_j nicht auf der Blattebene, d. h. $j < n$. Dann wird HEPV(G_j) in zwei Schritten erzeugt.

  Zunächst wird G_j auf Basis seiner Unter-Netzwerke $G_{j+1,i}$ auf Ebene $j+1$ wie folgt zu G'_j erweitert: Falls das Knotenpaar (n_1, n_2) mit $n_1, n_2 \in N_{j+1,i}^{G_j}$ in der transitiven Hülle von $G_{j+1,i}$ ist, füge (n_1, n_2) zu G'_j hinzu. Falls in G'_j dadurch ein neuer optimaler Pfad von n_1 nach n_2 entstanden ist, dann aktualisiere

das Gewicht von (n_1, n_2) in G'_j mit dem Gewicht des neuen optimalen Pfades (vergleiche Abbildung 6.7 oben Mitte).

Nun kann HEPV(G_j) berechnet werden, indem ein APSP-Algorithmus auf G'_j angewendet wird (vergleiche Tabelle 6.2 und Abbildung 6.7 oben rechts).

Suche nach einem optimalen Pfad $p^{O^\kappa}_{s\to t}$ von s nach t. Seien G_s mit Knotenmenge N_s und G_t mit Knotenmenge N_t diejenigen Teil-Netzwerke auf Blattebene der Hierarchie, für die $s \in N_s$ bzw. $t \in N_t$ gilt. Ihre topologiebezogenen Reasoning-Knoten werden mit Hilfe der *atomaren Teil-Netzwerk-Identifikation* bestimmt. Zur einfacheren Darstellung wird angenommen, dass die Suche nach dem optimalen Pfad von s nach t im topologiebezogenen Reasoning-Knoten T_k startet, dessen räumliches Teil-Netzwerk G_k mit Knotenmenge N_k und Kantenmenge E_k der – bezüglich der Hierarchie – kleinste gemeinsame Vorgänger von G_s und G_t ist. Dann können vier Fälle unterschieden werden:

1. Es ist $s, t \in N_k$. Dann kann T_k den Pfad $p^{O^\kappa}_{s\to t}$ direkt aus HEPV(G_k) ablesen.

   Am Beispiel von Tabelle 6.2 führt der kürzeste Pfad von v nach r von v über x (Zelle (v, r)) und y (Zelle (x, r)) nach r (Zelle (y, r)).

2. Es ist $s \in N_k \wedge t \notin N_k$ (und analog dazu $s \notin N_k \wedge t \in N_k$). Dann bestimmt T_k mit Hilfe der *atomaren Teil-Netzwerk-Identifikation* den topologiebezogenen Reasoning-Knoten T_t mit Teil-Netzwerk G_t und $t \in N_t$. T_k ermittelt außerdem anhand von E_k die Knoten $m \in N_k$, über die G_t erreicht werden kann. Aus HEPV(G_k) kann zu jedem m ein optimaler Pfad von s zu m direkt abgelesen werden. Jedes m ist Grenzknoten eines räumlichen Teil-Netzwerkes auf Ebene $k+1$. Das verbleibende Pfadstück von m zu t startet nun auf dieser Ebene. T_k ermittelt den optimalen Pfad von m nach t für jedes m durch rekursive Aufrufe an die Reasoning-Knoten der weiteren Unter-Netzwerke. Anschließend kombiniert T_k diese Teil-Pfade zu einem optimalen Pfad von s nach t.

   Am Beispiel von Abbildung 6.7 ergibt sich für den kürzesten Pfad von w nach s Folgendes: Es ist $w \in N_{n-1}$ und $s \in N_{n,2}$. $G_{n,2}$ ist in G_{n-1} über die Grenzknoten t, x, y, z erreichbar. In $G_{n,2}$ gilt anhand von HEPV($G_{n,2}$) für den kürzesten Pfad von t nach s die Länge 8, für x nach s die Länge 4, für y nach s die Länge 2 und für z nach s die Länge 3. Kombiniert mit den kürzesten Pfaden in HEPV(G_{n-1}) von w nach t, x, y, z mit den Längen $5, 4, 6, 10$ ergibt sich der kürzeste Pfad von w nach s also über x mit (w, x, y, s) und der Länge 8.

3. Es ist $s, t \notin N_k$: Hier kombiniert T_k das Verfahren aus Punkt 2 für beide Knoten s, t, indem er sowohl die optimalen Pfade von s zu den entsprechenden Grenzknoten als auch von den entsprechenden Grenzknoten zu t bestimmt und zusätzlich die optimalen Pfade zwischen diesen Grenzknoten aus HEPV(G_k) abliest. Daraus kann wieder ein optimaler Pfad von s nach t konstruiert werden.

Die detaillierte Ausformulierung der beschriebenen Algorithmen kann bei Mieth nachgelesen werden [Mie07]. Dort wird auch ausgearbeitet, wie eine hierarchisch kodierte Pfad-Sicht effizient aktualisiert werden kann, wenn sich das zugrunde liegende räumliche Teil-Netzwerk G ändert.

$L(n_1, n_2)$		n_2							
		q_2	r	z	y	t	x	v	w
n_1	q_2	–	$G_{n,1}$,2,r	G_{n-1},4,z	G_{n-1},7,r	G_{n-1}, 12,r	G_{n-1},9,r	G_{n-1}, 10,r	G_{n-1}, 13,r
	r	$G_{n,1}$,2, q_2	–	G_{n-1},3,z	G_{n-1},5,y	G_{n-1}, 10,y	G_{n-1},7,y	G_{n-1},8,y	G_{n-1}, 11,y
	z	G_{n-1},4, q_2	G_{n-1},3,r	–	$G_{n,2}$,4,y	G_{n-1},9,y	$G_{n,2}$,6,y	G_{n-1},7,x	G_{n-1}, 10,x
	y	G_{n-1},7,r	G_{n-1},5,r	$G_{n,2}$,4,z	–	G_{n-1},5,x	$G_{n,2}$,2,x	G_{n-1},3,x	G_{n-1},6,x
	t	G_{n-1}, 12,x	G_{n-1}, 10,x	G_{n-1},9,x	G_{n-1},5,x	–	G_{n-1},3,v	G_{n-1},2,v	G_{n-1},5,v
	x	G_{n-1},9,y	G_{n-1},7,y	$G_{n,2}$,6,y	$G_{n,2}$,2,y	G_{n-1},3,t	–	G_{n-1},1,v	G_{n-1},4,v
	v	G_{n-1}, 10,x	G_{n-1},8,x	G_{n-1},7,x	G_{n-1},3,x	G_{n-1},2,t	G_{n-1},1,x	–	$G_{n,3}$, 3,w
	w	G_{n-1}, 13, v	G_{n-1}, 11, v	G_{n-1}, 10, v	G_{n-1},6,v	G_{n-1}, 5, v	G_{n-1}, 4, v	$G_{n,3}$, 3, v	–

Tabelle 6.2.: Darstellung von E^+_{n-1} aus $\text{HEPV}(G_{n-1}) = (N_{n-1}, E^+_{n-1})$ mit N_{n-1} als Spalten und Zeilen und den Kennzeichnungen L als Einträgen (vergleiche Abbildung 6.7 oben rechts).

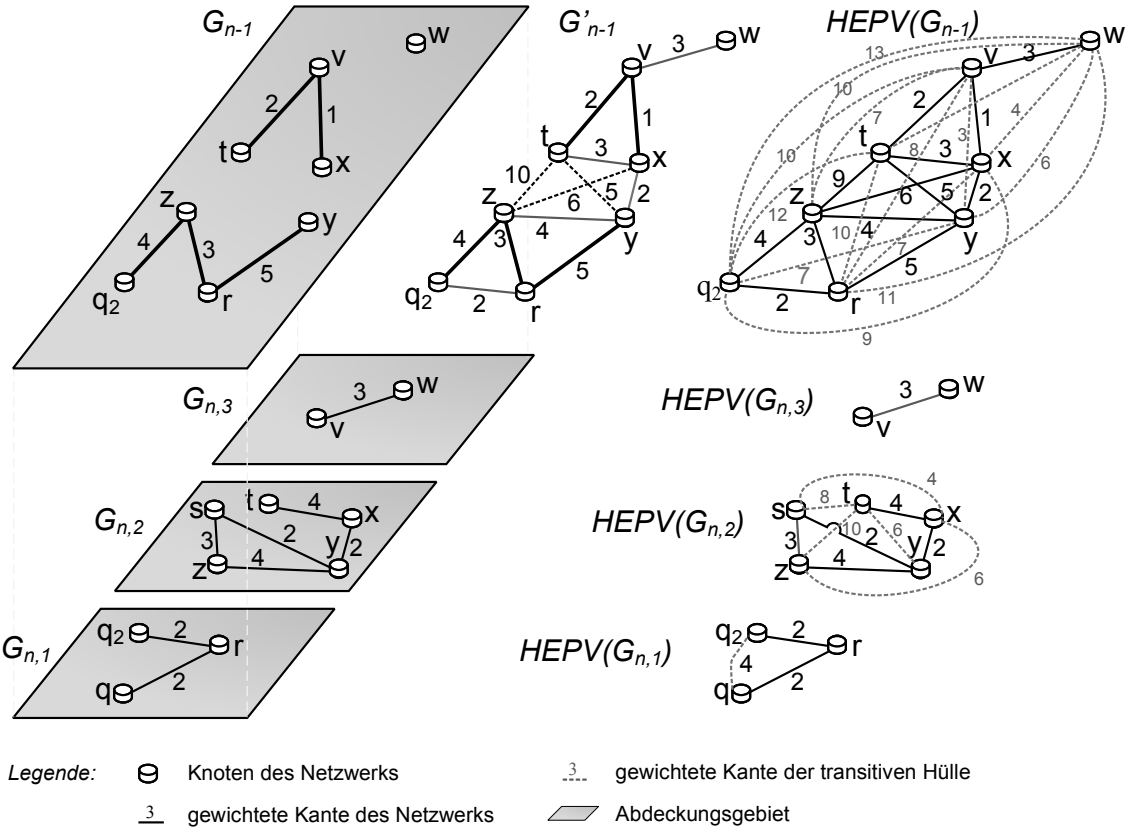

Abbildung 6.7.: Beispiele für die Erzeugung von hierarchisch kodierten Pfad-Sichten (nach [Mie07]).

6.5.4.2. Fast-optimale Pfade

Das zuvor vorgestellte Verfahren zum Finden garantiert optimaler Pfade beruht darauf, dass umfangreiche Datenstrukturen vorberechnet werden. Dies ist nur für statische Kontexttypen wie beispielsweise Länge oder Elektrifizierung eines Gleisabschnitts bzw. Spannungsebene einer Stromleitung praktikabel. Bei dynamischen Kontexttypen wie Abnutzung, Niederschlag und Ähnlichem kann sich eine derartige Vorverarbeitung jedoch nicht auszahlen.

Das vorgeschlagene topologiebezogene Reasoning-System ermöglicht es, unterschiedliche Verfahren zur Beantwortung topologiebezogener Anfragen einzusetzen. Darum kann für dynamische Kontexttypen problemlos ein anderes, besser geeignetes Verfahren verwendet werden. Ein Ansatz dafür sind zum Beispiel Verfahren, die auf Verfeinerung (engl. *refinement search*) basieren und dabei jeweils die aktuellsten Kontextinformationen berücksichtigen können. Diese können zwar nicht garantieren, dass der gefundene Pfad optimal ist. Es werden jedoch bestimmte Heuristiken genutzt, mit denen in der Regel ein fast-optimaler Pfad gefunden wird. Zudem erhöhen diese Verfahren die Effizienz, weil sie den Suchraum effektiv einschränken.

Die Suche nach (fast-)optimalen Pfaden mit Verfeinerung beruht auf einer Abstraktion der Netzwerktopologie: Für jedes räumliche Teil-Netzwerk G_j auf Ebene j wird zunächst der *Nachbarschaftsgraph* der Teil-Netzwerke $G_{j+1,i}$ auf Ebene $j+1$ genutzt, um einen *abstrakten Pfad* von Teil-Netzwerken zu ermitteln. Der abstrakte Pfad wird anschließend zu einem konkreten Pfad von Kanten verfeinert. Dazu werden nur noch die $G_{j+1,i}$ auf dem abstrakten Pfad berücksichtigt; die übrigen $G_{j+1,i}$ können ignoriert werden.

Berechnung eines Nachbarschaftsgraphen. Für $k < n$ ist der *Nachbarschaftsgraph* definiert als

$$G_k^A = (\{\, G_{k+1,i} \mid G_{k+1,i} \text{ ist Unter-Netzwerk von } G_k \text{ auf Ebene } k+1\}, E_k^A)$$

mit $E_k^A \subseteq \mathfrak{G} \times \mathfrak{G} \times \mathbb{R}_{\geq 0}$. Dabei gelte $(G_{k+1,i}, G_{k+1,i'}, w) \in E_k^A$ genau dann, wenn $G_{k+1,i}$ und $G_{k+1,i'}$ benachbart sind, d. h. wenn in E_k eine Kante von einem Knoten in $N_{k+1,i}^{G_k}$ zu einem Knoten in $N_{k+1,i'}^{G_k}$ existiert. w ist abhängig von der verwendeten Heuristik. Darauf wird später eingegangen. Beispiele für Nachbarschaftsgraphen sind in Abbildung 6.8 neben den räumlichen Teil-Netzwerken dargestellt.

Suche nach einem fast-optimalen Pfad $p_{s\to t}^{O^\kappa}$ von s nach t. Sei also ein (fast-) optimaler Pfad zwischen den Knoten s und t gesucht. Sei T_k wieder der topologiebezogene Reasoning-Knoten, dessen räumliches Teil-Netzwerk G_k der kleinste gemeinsame Vorgänger der Teil-Netzwerke auf Blattebene ist, die s bzw. t enthalten.

Der *abstrakte Pfad* $p_k^{A,s\to t}$ von s nach t im Nachbarschaftgraphen G_k^A ist eine Liste von Teil-Netzwerken $(G_{k+1,1}, \ldots, G_{k+1,m})$. Er entspricht einem optimalen Pfad von $G_{k+1,1}$ nach $G_{k+1,m}$ im Nachbarschaftsgraphen G_k^A, wobei s in $G_{k+1,1}$ oder einem seiner Unter-Netzwerke und t in $G_{k+1,m}$ oder einem seiner Unter-Netzwerke enthalten ist.

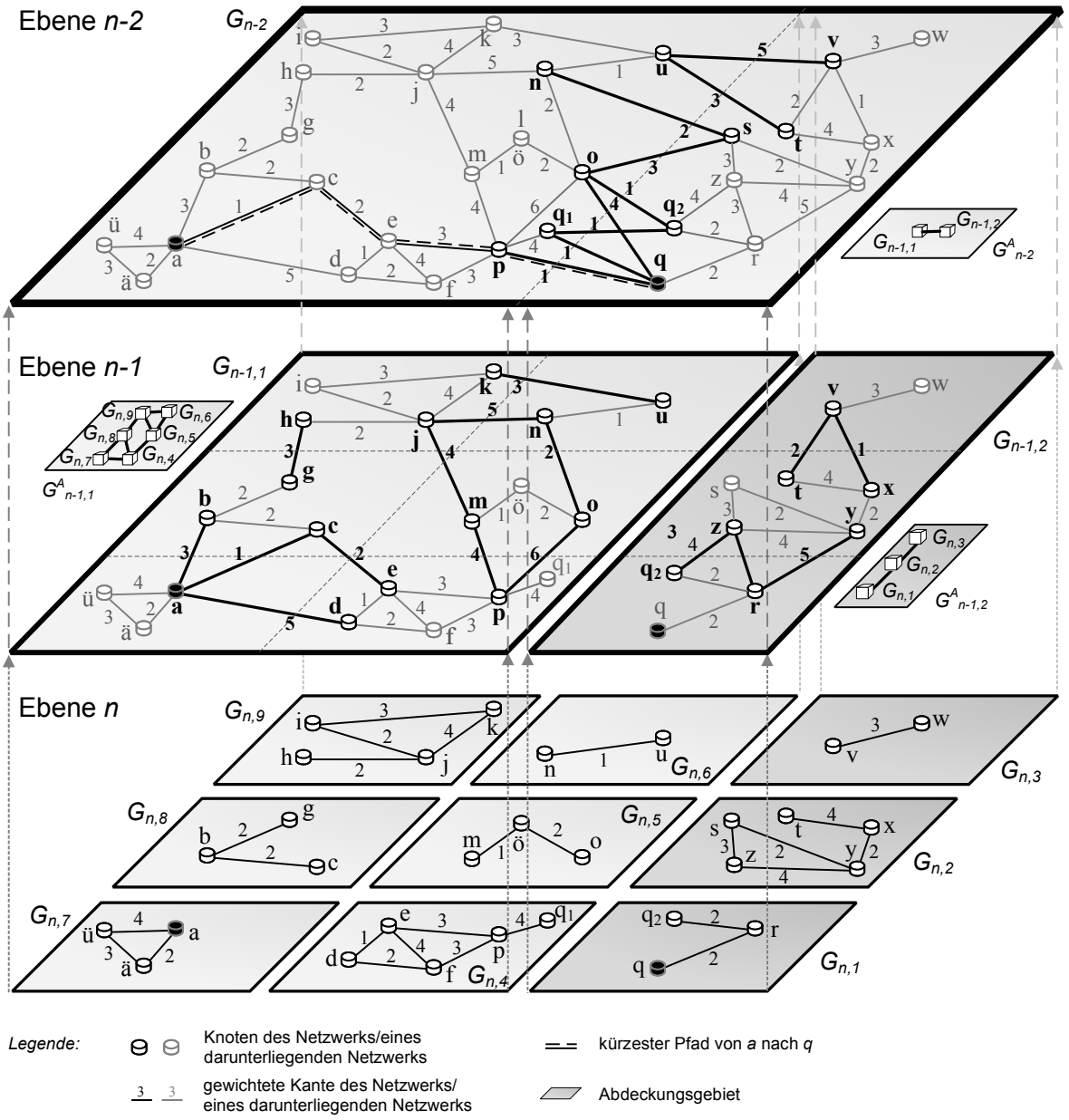

Abbildung 6.8.: Beispiel für eine Verfeinerungs-basierte Suche nach einem kürzesten Pfad von a in $G_{n,7}$ nach q in $G_{n,1}$ (nach [Mie07]).

Sei also $p_k^{A,s\to t} = (G_{k+1,1}, \ldots, G_{k+1,m})$ der abstrakte Pfad von Teil-Netzwerken für den Pfad von s nach t, den der topologiebezogene Reasoning-Knoten T_k ermittelt hat. Um ihn zu einem konkreten Pfad von Kanten zu verfeinern, betrachtet T_k alle Tripel $(G_{k+1,l-1}, G_{k+1,l}, G_{k+1,l+1})$ in $p_k^{A,s\to t}$ wie folgt:

1. Für jedes Tripel $(G_{k+1,l-1}, G_{k+1,l}, G_{k+1,l+1})$ fragt T_k den mittleren Reasoning-Knoten $T_{k+1,l}$ bezüglich seines Teil-Netzwerks $G_{k+1,l}$ nach gewichteten optimalen Pfaden: Diese starten in einem Grenzknoten, über den eine Verbindung von $G_{k+1,l-1}$ nach $G_{k+1,l}$ besteht, und enden in einem Grenzknoten, über den eine Verbindung von $G_{k+1,l}$ nach $G_{k+1,l+1}$ besteht. Diese Verbindungen sind T_k

aufgrund von E_k bekannt.

Formalisiert heißt das: T_k fragt jeden $T_{k+1,l}$ nach gewichteten optimalen Pfaden für alle Grenzknoten-Paare $(n^{(l-1)\rightarrow}, n^{\rightarrow l})$ mit $n^{(l-1)\rightarrow} \in N^{G_k}_{k+1,l-1}, n^{\rightarrow l} \in N^{G_k}_{k+1,l}$ und $(n^{(l-1)\rightarrow}, n^{\rightarrow l}) \in E_k$ sowie für alle Grenzknoten-Paare $(n^{l\rightarrow}, n^{\rightarrow(l+1)})$ mit $n^{l\rightarrow} \in N^{G_k}_{k+1,l}, n^{\rightarrow(l+1)} \in N^{G_k}_{k+1,l+1}$ und $(n^{l\rightarrow}, n^{\rightarrow(l+1)}) \in E_k$. Die Spezialfälle sind $s \in N_{k+1,l}$ mit $(s, n^{\rightarrow l})$ und $t \in N_{k+1,l}$ mit $(n^{l\rightarrow}, t)$.

Dabei ist zu beachten, dass jeder $T_{k+1,l}$ zur Beantwortung dieser Anfrage unter Umständen weitere abstrakte Pfade $p^{A,n\rightarrow n'}_{k+1,l}$ mit $n, n' \in N_{k+1,l}$ rekursiv ermitteln und verfeinern muss. Dies kann auch parallelisiert werden. Die Gewichte sind abhängig vom Optimalitätskriterium. Ihre Berechnung kann auf die vorher eingeführten *atomaren Kontext-Abfragen* reduziert werden. Auf diese Weise berücksichtigt dieses Verfahren immer die aktuellsten Kontextinformationen.

2. T_k (und rekursiv auch weitere Reasoning-Knoten auf Ebene $i > k$) kombiniert die so erhaltenen Pfade mit der Kantenmenge E_k seines Teil-Netzwerks G_k zu einem temporären Netzwerk $G^{s\rightarrow t}_k$. $G^{s\rightarrow t}_k$ enthält dann s, t, die Verbindungen zwischen den relevanten Teil-Netzwerken über ihre Grenzknoten sowie die (fast-)optimalen Pfade innerhalb der Teil-Netzwerke zwischen diesen Grenzknoten. Dann kann T_k einen (fast-)optimalen Pfad von s nach t in der Hierarchie durch einfache Suche nach einem optimalen Pfad in $G^{s\rightarrow t}_k$ ermitteln.

Das Beispiel in Abbildung 6.8 illustriert die Suche nach einem fast-kürzesten Pfad von a nach q in G_{n-2}. Als Kantengewicht im Nachbarschaftsgraphen wird das Einheitsgewicht 1 verwendet:

Für den abstrakten Pfad von a nach q in G_{n-2} gilt $p^{A,a\rightarrow q}_{n-2} = (G_{n-1,1}, G_{n-1,2})$. Deshalb werden in $G_{n-1,1}$ kürzeste Pfade von a nach p, q_1, o, n und u sowie in $G_{n-1,2}$ kürzeste Pfade von q, q_2, s, t und v nach q gesucht. Dazu werden rekursiv weitere abstrakte Pfade berechnet und deshalb die Nachbarschaftsgraphen $G^A_{n-1,1}$ und $G^A_{n-1,2}$ von $G_{n-1,1}$ und $G_{n-1,2}$ betrachtet.

In $G^A_{n-1,1}$ gilt beispielsweise für den abstrakten Pfad von a nach p $p^{A,a\rightarrow p}_{n-1,1} = (G_{n,7}, G_{n,8}, G_{n,4})$ und für den abstrakten Pfad von a nach o $p^{A,a\rightarrow o}_{n-1,1} = (G_{n,7}, G_{n,8}, G_{n,4}, G_{n,5})$. In $G^A_{n-1,2}$ ist der abstrakte Pfad von s nach q $p^{A,s-q}_{n-1,2} = (G_{n,2}, G_{n,1})$.

Zur Verfeinerung von $p^{A,a\rightarrow p}_{n-1,1}$ muss das Tripel $(G_{n,7}, G_{n,8}, G_{n,4})$ betrachtet werden. In $G_{n,7}$ werden kürzeste Pfade von a zu allen Grenzknoten – in diesem Fall also lediglich zu a selbst – gesucht; in $G_{n,8}$ werden kürzeste Pfade von b zu c und von c zu c gesucht; in $G_{n,4}$ wird der kürzeste Pfad von e zu p ermittelt. Aus diesen Pfaden ergibt sich in $G_{n-1,1}$ zusammen mit $E_{n-1,1}$ ein temporärer Graph $G^{a\rightarrow p}_{n-1,1}$. In diesem wird der kürzeste Pfad von a nach p als (a, c, e, p) mit Länge 6 bestimmt.

Das ist einer der Pfade, die in G_{n-2} zusammen mit den anderen Pfaden der Unter-Netzwerke und mit E_{n-2} zu dem temporären Graphen $G^{a\rightarrow q}_{n-2}$ vereinigt werden. Die Suche in diesem Graph ergibt als fast-kürzesten Pfad von a nach q schließlich (a, c, e, p, q) mit Länge 7.

Dieses Verfahren hat wesentliche Vorteile im Vergleich zu einem naiven Ansatz, der alle Unter-Netzwerke von G_k in T_k aggregiert: Mit Hilfe der abstrakten Pfade können

Teil-Netzwerke, die (fast sicher) für das Finden eines optimalen Pfades irrelevant sind, von Anfang an ignoriert werden. Weil die $T_{k+1,l}$ die Verfeinerungsschritte unabhängig voneinander vornehmen können, kann Rechenlast effektiv verteilt und die Bearbeitung parallelisiert werden. Schließlich wird die Datenmenge, die T_k verarbeiten muss, wesentlich verringert, weil $G_k^{s \to t}$ nur die relevante Untermenge aller $G_{k+1,l}$ enthält.

Dabei hängt der Optimalitätsgrad des errechneten Pfades von der Heuristik zur Gewichtung der Nachbarschaftsgraphen ab. Die einfache Heuristik h_1 weist allen Kanten in G_k^A das Einheitsgewicht 1 zu. Eine andere Heuristik h_2 gewichtet alle $e \in E_k^A$ mit dem Minimum der Gewichte aller Kanten, die von e abstrahiert werden. Welche Heuristik dabei einen Pfad ergibt, der näher am optimalen Pfad ist, hängt sowohl von der Anfrage als auch vom Graphen ab. Häufig wird jedoch h_1 verwendet, weil sie sehr einfach ist und in der Regel dennoch gute Ergebnisse liefert (vergleiche [HMZM96]).

Insgesamt kann das verfeinerungsbasierte Verfahren also nicht garantieren, dass ein optimaler Pfad gefunden wird. Aus den oben genannten Gründen ist es in der Regel jedoch sehr effizient. Vor allem ist es nicht auf die Vorverarbeitung von Kontextinformationen angewiesen und kann deshalb auch bei dynamischen Kontexttypen die aktuellsten Werte berücksichtigen.

6.5.5. Nächste-Nachbarn-Anfragen bezüglich eines Infrastrukturelements

Ein weiterer komplexer Anfragetyp, der für Infrastrukturüberwachung wichtig ist, sind k-nächste-Nachbarn-Anfragen. Eine Anfrage könnte beispielsweise die Suche nach den drei – bezogen auf den Standort eines defekten Zuges – nächstgelegenen Werkstätten sein, die über Qualifikation und Kapazitäten zur Reparatur verfügen. Dieser Abschnitt zeigt, wie auch dafür ein bestehendes Verfahren adaptiert und mit Bezug zu den atomaren Anfragetypen umgesetzt werden kann.

Definition 6.5.3 (Topologiebezogene *k*-nächste-Nachbarn-Anfrage). *Sei s der Startknoten, $\mathfrak{G}$ die Menge der räumlichen Teil-Netzwerke in der Hierarchie und C eine Kontextbedingung. Dann findet eine* k-nächste-Nachbarn (kNN)-Anfrage *die $k \geq 1$ Knoten in den räumlichen Teil-Netzwerken in $\mathfrak{G}$, für die $C(n)$ gilt und die s bezüglich der topologischen Distanz am nächsten sind.*

Zur Beantwortung von k-nächste-Nachbarn-Anfragen gibt es viele Verfahren, die auf Vorverarbeitung basieren und Indizes erstellen (siehe [KS04] für eine ausführliche Beschreibung). Um bei der Anfragebeantwortung die jeweils aktuellsten Kontextinformationen berücksichtigen zu können, soll hier jedoch ebenfalls wieder auf Vorverarbeitung verzichtet werden.

In räumlichen Netzwerken wird als Distanz die *topologische* Distanz betrachtet. Da als Lösungen nur Knoten in Frage kommen, die Teil des Netzwerks sind, kann bei der Anfragebeantwortung die Kenntnis der Netztopologie ausgenutzt werden. Zwei derartige Verfahren werden in [PZMT03] vorgestellt, von denen *Incremental Network Expansion* (INE) im Folgenden adaptiert wird.

Die Kernidee von INE besteht darin, vom Startknoten s aus den Graphen schrittweise solange zu expandieren, bis k Lösungen gefunden wurden, die die Kontextbedingung C

erfüllen. Dadurch handelt es sich um eine Variante des *Dijkstra-Algorithmus* [Dij59], wobei als Abbruchkriterium die Zahl der gefundenen Lösungen verwendet wird.

Beim Traversieren der Infrastrukturtopologie muss die Hierarchie berücksichtigt werden. Sei T_n der topologiebezogene Reasoning-Knoten, dem das räumliche Teil-Netzwerk G_n auf Blattebene n zugeordnet ist, welches für den Startknoten s zuständig ist. Diese wird mittels *atomarer Teil-Netzwerk-Identifikation* ermittelt. Dann prüft T_n für alle Grenzknoten $N_n^{G_{n-1}}$, die im Zuge der schrittweisen Expansion erreicht werden, ob Verbindungen zu anderen räumlichen Teil-Netzwerken bestehen. Diese Information ist in G_{n-1} als Kantenmenge E_{n-1} repräsentiert. Deshalb delegiert T_n einen Teil der Beantwortung der Nächste-Nachbarn-Anfrage an den topologiebezogenen Reasoning-Knoten T_{n-1} auf Ebene $n-1$. Müssen im Laufe des Verfahrens weitere Knoten expandiert werden, weil noch nicht k Lösungen gefunden wurden, delegiert T_{n-1} dies wiederum an weitere topologiebezogene Reasoning-Knoten. Für jeden expandierten Knoten wird die *atomare Kontext-Abfrage* genutzt, um zu prüfen, ob die Kontextbedingung C erfüllt ist und deshalb ein Element der Lösung gefunden wurde. Dabei kann sich die Anfragebeantwortung rekursiv auch über mehr als zwei Hierarchieebenen erstrecken.

6.5.6. Bereichsanfragen bezüglich eines Infrastrukturelements

Ebenfalls relevant für die Infrastrukturüberwachung sind Bereichsanfragen. Ein Beispiel hierfür ist die Suche nach allen Oberleitungen, die höchstens 50 km Fahrstrecke von München entfernt sind und deren Abnutzungszustand baldige Instandsetzung erfordert. Auch für diesen Anfragetyp wird im Folgenden ein bestehendes Verfahren adaptiert und auf Basis der atomaren Anfragetypen umgesetzt.

Definition 6.5.4 (Topologiebezogene Bereichsanfrage). *Sei s der Startknoten, $\mathfrak{G}$ die Menge der räumlichen Teil-Netzwerke in der Hierarchie, C eine Kontextbedingung und d ein Entfernungswert. Dann findet eine* Bereichsanfrage *alle Knoten n und Kanten e in den räumlichen Teil-Netzwerken in $\mathfrak{G}$, für die $C(n)$ bzw. $C(e)$ gilt und die von s eine topologische Distanz von höchstens d haben.*

Auch bei Bereichsanfragen wird als Distanz in räumlichen Netzwerken die topologische Distanz betrachtet. Bei der Anfragebearbeitung wird deshalb ebenfalls die Netztopologie ausgenutzt. [PZMT03] stellt wieder zwei Verfahren zur Beantwortung topologiebezogener Bereichsanfragen vor. Davon wird im Folgenden *Range Network Expansion* (RNE) adaptiert.

Die Kernidee von RNE besteht darin, vom Startknoten s aus den Graphen schrittweise solange zu expandieren und Knoten und Kanten, die die Kontextbedingung C erfüllen, als Lösungen zurückzuliefern, bis die gegebene Bereichs-Distanz überschritten ist. Dadurch handelt es sich um eine klassische *beschränkte Tiefen-* oder *Breitensuche* (siehe z. B. [RN03]). Dabei ist die Tiefensuche der Breitensuche vorzuziehen, weil ihr Speicherbedarf zur Laufzeit geringer ist. Im Unterschied zur reinen Tiefensuche wird durch die Begrenzung der Suchtiefe neben der Korrektheit auch die Vollständigkeit der Ergebnisse garantiert.

Beim Durchlaufen der Infrastrukturtopologie muss die Hierarchie berücksichtigt werden. Sei T_n der mittels *atomarer Teil-Netzwerk-Identifikation* ermittelte topologiebezogene Reasoning-Knoten, dem das räumliche Teil-Netzwerk G_n auf Blattebene

zugeordnet ist, das für den Startknoten s zuständig ist. Dann muss der topologiebezogene Reasoning-Knoten T_n für alle Grenzknoten $N_n^{G_{n-1}}$, die im Laufe des Verfahrens erreicht werden, prüfen, ob Verbindungen zu anderen räumlichen Teil-Netzwerken bestehen. Diese Information ist in G_{n-1} in Form der Kantenmenge E_{n-1} repräsentiert. Deshalb delegiert T_n einen Teil der Beantwortung der Bereichsanfrage an den topologiebezogenen Reasoning-Knoten T_{n-1} auf Ebene $n-1$. Müssen auch hier die Kanten weiterverfolgt werden, delegiert T_{n-1} die Anfrage wiederum an weitere topologiebezogene Reasoning-Knoten. Für jeden expandierten Knoten und jede expandierte Kante wird die *atomare Kontext-Abfrage* genutzt, um zu prüfen, ob die Kontextbedingung C erfüllt ist und deshalb ein Element der Lösung gefunden wurde. Dabei kann sich die Anfragebeantwortung rekursiv auch wieder über mehr als zwei Hierarchieebenen erstrecken.

6.5.7. Zusammenfassung

Insgesamt hat dieser Abschnitt also gezeigt, wie mit Hilfe des vorgeschlagenen Systems für topologiebezogenes Reasoning unterschiedliche Verfahren zur Beantwortung von Kontext-Filter-, Optimale-Pfade-, Nächste-Nachbarn- und Bereichsanfragen für eine Hierarchie räumlicher Teil-Netzwerke umgesetzt werden können. Teilweise wurden bestehende Verfahren adaptiert, teilweise neue Verfahren entwickelt. Als Annahmen liegen allen Umsetzungen lediglich dasselbe hierarchische Modell des Infrastrukturnetzes und dieselben Wissensbasen mit Kontextinformationen zu den Elementen des Infrastrukturnetzes zugrunde. Das demonstriert die Vielseitigkeit des Ansatzes und motiviert die Erweiterung um Verfahren für weitere Anfragetypen.

6.6. Framework zur Umsetzung der Verfahren

Dieser Abschnitt entwickelt ein Framework, mit dem die zuvor beschriebenen Verfahren für topologiebasiertes Reasoning umgesetzt werden können. Zunächst werden Sprachen zur Repräsentation des Infrastrukturnetzes und zur Formulierung der Anfragen vorgestellt. Anschließend werden die Komponenten und Schnittstellen des Frameworks beschrieben.

6.6.1. Repräsentation des Infrastrukturnetz-Modells

Die Modellierung des Infrastrukturnetzes wurde in Abschnitt 6.3 entwickelt. Sie basiert auf einer Hierarchie von räumlichen Teil-Netzwerken. Räumliche Netzwerke können auf unterschiedliche Arten repräsentiert werden. Wie in Abschnitt 6.2.2 vorgestellt wurde, existieren dafür teilweise erste internationale Standards. Bislang hat sich jedoch noch keine einheitliche Repräsentation durchgesetzt. Darum wird für die hier vorgestellte Implementierung eine eigene Darstellung entwickelt, die auf OWL DL basiert. Damit soll erreicht werden, dass die einmal erstellte Repräsentation eines Infrastrukturnetzes in Zukunft bei Bedarf in eine andere Repräsentation transformiert werden kann: Sollte sich eine ontologiebasierte Darstellung als Standard für räumliche Netzwerke durchsetzen, können bestehende Netzwerkmodelle mittels einfacher semantischer Abbildungen

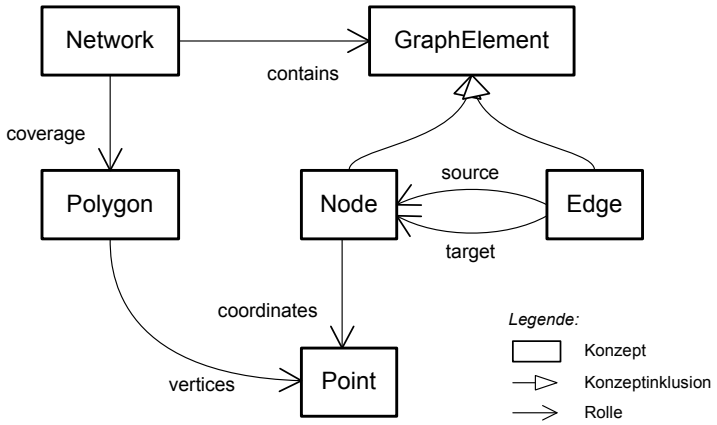

Abbildung 6.9.: Konzepte und Rollen zur beschreibungslogischen Repräsentation räumlicher Netzwerke.

weiterverwendet werden. Auch wenn sich andere Datenmodelle für die Repräsentation von räumlichen Netzwerken etablieren – bei ISO-Standards beispielsweise relationale Modelle – sind automatisierte Transformationen von der OWL DL-basierten Darstellung in die relationale Darstellung möglich.

Deshalb nutzt die vorgestellte Implementierung eine OWL DL-basierte Repräsentation als Austauschformat für räumliche Netzwerke. Dazu wurde eine Ontologie entwickelt, die die nötigen Konzepte und Rollen spezifiziert. Die wichtigsten Konzepte und Zusammenhänge sind in Abbildung 6.9 dargestellt: Ein Network enthält GraphElements und damit Nodes und Edges, weil diese die einzigen Spezialisierungen darstellen. Node repräsentiert Knoten im räumlichen Netzwerk, denen über die Rolle coordinates außerdem eine Lage im räumlichen Bezugssystem zugeordnet werden kann. Edge repräsentiert die gerichteten Kanten des Netzwerks, die über source mit ihrem Start- und über target mit ihrem Endknoten verbunden sind. Das Abdeckungsgebiet eines Netzwerks wird über die Rolle coverage spezifiziert, die auf ein Polygon zeigt. Dieses ist auf Basis von Points spezifiziert.

Die Repräsentation eines räumlichen Netzwerks besteht aus ABox-Zusicherungen auf Basis dieser Ontologie. Die Hierarchie zwischen räumlichen Teil-Netzwerken wird repräsentiert, indem die Individuennamen der Grenzknoten $N_j^{G_{j-1}}$ eines räumlichen Teil-Netzwerks G_j auf Ebene j als Individuennamen für Knoten im räumlichen Teil-Netzwerk G_{j-1} auf Ebene $j-1$ wiederverwendet werden.

6.6.2. Auswahl der Anfragesprache

Zur Formulierung topologiebezogener Anfragen hat sich ebenfalls noch kein Standard etabliert. Da für das Infrastrukturnetz-Modell eine ontologiebasierte Repräsentation gewählt wurde, bietet sich zur Formulierung topologiebezogener Anfragen eine ebenfalls ontologiebasierte Anfragesprache an. Ontologiebasierte Anfragesprachen wurden bereits für das Framework zum Reasoning über verteilte Kontextinformationen in Abschnitt 5.4.2 verglichen. Dabei wurde der W3C-Standard SPARQL [PS08] als ausgereifte und zukunftssichere Anfragesprache identifiziert.

6.6.2.1. Existierende Erweiterungen von SPARQL

Wie bereits erwähnt, reicht SPARQL jedoch nicht aus, um alle topologiebezogenen Anfragen zu formulieren. So ist es mit SPARQL beispielsweise nicht möglich, die Suche nach einem Pfad zwischen zwei Knoten zu formulieren, ohne die genaue Länge dieses Pfades anzugeben. Auch ist nicht klar, wie eine Anfrage nach nächsten Nachbarn spezifiziert wird. Das ist analog zu relationalen Datenbanken, wo die klassische Anfragesprache SQL nicht alle relevanten topologiebezogenen Anfragen ausdrücken kann. Für relationale Datenbanken wurde deshalb SQL/MM *Spatial* (ISO/IEC 13249-3:2006) entwickelt. Entsprechende Erweiterungen für SPARQL werden im Folgenden vorgestellt:

- PSPARQL erweitert SPARQL um die Möglichkeit, reguläre Ausdrücke in den konjunktiven Anfragen von SPARQL – den sogenannten *graph patterns* – zu verwenden [ABE07]. Die Existenz eines beliebig langen Pfades zwischen zwei Knoten kann somit durch $^+$-Quantifizierung[4] der entsprechenden Rolle in der Ontologie geprüft werden. Jedoch ist es mit PSPARQL nicht möglich, die Kanten und Knoten dieses Pfades auch als Ergebnis zu erhalten!

- SPARQLeR [KJ07] und SPARQ2L [AMS07] führen deshalb spezielle Variablen, sogenannte *Pfadvariablen*, ein. Beide Erweiterungen ähneln sich, SPARQLeR ist jedoch etwas ausdrucksstärker. Mit Pfadvariablen können *Pfade* als Lösung einer Anfrage an eine Variable gebunden werden. Auf Basis einer solchen Pfadvariable sind dann weitergehende Operationen möglich: Einerseits kann ein Pfad einfach als Ergebnis zurückgegeben werden. Außerdem können mit Hilfe des `FILTER`-Mechanismus von SPARQL zusätzliche Bedingungen an Pfadvariablen gestellt werden wie beispielsweise, dass der Pfad nur bestimmte Rollen enthalten darf. Listing 6.1 zeigt eine Beispielanfrage für einen Pfad von dem Individuum Dresden zu dem Individuum München, der nur Kanten der Rollen core:source und core:target enthält (vergleiche die ontologische Repräsentation von räumlichen Netzwerken in Abbildung 6.9).

PSPARQL eignet sich also nicht zur Formulierung der hier betrachteten topologiebezogenen Anfragen, weil es nicht möglich ist, die gefundenen Pfade als Ergebnis zurückzugegeben. Das von SPARQLeR und SPARQ2L eingeführte Konzept der Pfadvariablen ist hingegen sehr mächtig, weil es auf diese Weise möglich ist, zusätzliche Bedingungen an die gesuchten Pfade zu stellen. SPARQLeR nutzt dafür das von SPARQL definierte `FILTER`-Konstrukt, während SPARQ2L eigene Konstrukte wie z. B. `PathFilter` einführt. Da die Erweiterungen von SPARQLeR also in Einklang mit dem SPARQL-Standard stehen, wird für die Implementierung des Frameworks auf SPARQLeR aufgebaut.

```
SELECT list(%path)
WHERE { Dresden %path München .
   FILTER( regex(%path, "(-core:source|core:target)+") ). }
```

Listing 6.1: SPARQLeR-Anfrage nach einem Pfad zwischen Dresden und München.

[4] Der $^+$-Quantor entspricht der üblichen Syntax für relationale Ausdrücke und besagt, dass der voranstehende Ausdruck mindestens einmal vorkommen muss, aber auch mehrfach vorkommen darf.

6.6.2.2. Eigene Erweiterungen zu SPARQLeR

SPARQLeR erweitert SPARQL um Pfadvariablen. Unter Verwendung des `FILTER`-Konstrukts von SPARQL können zusätzliche Bedingungen sowohl an normale Variablen als auch an Pfadvariablen gestellt werden. Um die in dieser Arbeit vorgestellten topologiebezogenen Anfragen in SPARQLeR formulieren zu können, werden deshalb folgende `FILTER`-Operatoren eingeführt. Derartige Erweiterungen sind im SPARQL-Standard ausdrücklich vorgesehen [PS08]. Kontextbezogene Bedingungen können zusätzlich wie bisher als Konjunktionen von Anfrageatomen formuliert werden.

- `optimality` dient zur Formulierung von *Optimale-Pfade-Anfragen*. Mit diesem `FILTER`-Operator können das Optimalitätskriterium und der Bezug zu einem Kontexttyp ausgedrückt werden. Ein Beispiel ist in Listing 6.2 dargestellt.
- `shortest` wird zur Formulierung von *Nächste-Nachbarn-Anfragen* benötigt. Dieser `FILTER`-Operator spezifiziert die Anzahl der gesuchten nächsten Nachbarn (siehe Listing 6.3).
- `range` wird bei *Bereichsanfragen* verwendet. Es gibt den Rollennamen, auf den sich die Spezifikation des Bereichs bezieht, und seinen Maximalwert an. Listing 6.4 gibt hierzu ein Beispiel.
- `contains` dient zur Formulierung von *Kontext-Filter-Anfragen*, die ein räumliches Gebiet als Filterbedingung enthalten. Dazu ist in Listing 6.5 ein Beispiel dargestellt.
- `isProperPart`, `isPartiallyOverlapping` und `isContainedIn` werden für Anfragen bezüglich des Abdeckungsgebiets von Netzwerken benötigt. Beispiele hierfür sind in Listings 6.6 und 6.7 gegeben.

```
SELECT list(%path)
WHERE { Dresden %path München .
   FILTER( optimality(%path, RiskLevel, ''min'') ). }
```

Listing 6.2: SPARQLeR-Anfrage nach einem Pfad zwischen Dresden und München, der mit einem minimalen Risiko behaftet ist.

```
SELECT ?node
WHERE { Dresden %path ?node .
   FILTER( shortest(%path, ''3'') ). }
```

Listing 6.3: SPARQLeR-Anfrage nach den drei Knoten, die Dresden bezüglich der topologischen Distanz am nächsten liegen.

```
SELECT ?node
WHERE { Dresden %path ?node .
   FILTER( range(%path, hasLengthInKm, ''30'') ). }
```

Listing 6.4: SPARQLeR-Anfrage nach allen Pfaden, die in Dresden starten und die nicht länger als 30 km sind.

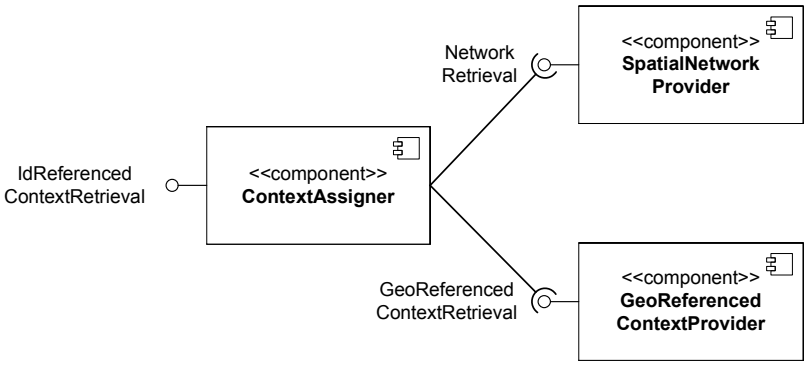

Abbildung 6.10.: Komponenten zur netzbasierten Integration ortsbezogener Kontextinformationen.

```
SELECT ?node
WHERE { ?node coordinates ?coord.
   FILTER( contains(‘‘48.133333 11.566667 49.452778 11.077778
      51.049259 13.73836 50.088611 14.421389’’, ?coord) ). }
```

Listing 6.5: SPARQLeR-Anfrage nach allen Knoten, die sich in dem Polygon befinden, das von München, Nürnberg, Dresden und Prag gebildet wird. Koordinaten können im WGS84-Format, Polygone als Liste von mindestens drei dieser Koordinaten spezifiziert werden.

```
ASK { Network4 coverage ?area.
   FILTER( isProperPart(‘‘48.133333 11.566667 49.452778 11.077778
      51.049259 13.73836 50.088611 14.421389’’, ?area) ). }
```

Listing 6.6: SPARQLeR-Anfrage, die entscheidet, ob ein gegebenes Polygon in dem Abdeckungsgebiet eines gegebenen Teil-Netzwerks vollständig enthalten ist.

```
ASK { Point7 rdf:type Point.
   FILTER( isContainedIn(Point7, ‘‘48.133333 11.566667 49.452778
      11.077778 51.049259 13.73836 50.088611 14.421389’’) ). }
```

Listing 6.7: SPARQLeR-Anfrage, die entscheidet, ob ein gegebener Punkt in einem Polygon enthalten ist.

6.6.3. Komponenten und Schnittstellen des Frameworks

Das Framework zum topologiebezogenen Reasoning umfasst Komponenten zur netzbasierten Integration ortsbezogener Kontextinformationen und zur Beantwortung topologiebezogener Anfragen. Diese werden im Folgenden mit ihren Schnittstellen vorgestellt.

6.6.3.1. Umsetzung der netzbasierten Integration ortsbezogener Kontextinformationen

Zur Integration ortsbezogener Kontextinformationen wird das in Abschnitt 6.4 beschriebene Verfahren umgesetzt. Dieses nutzt kontexttypspezifische Wirkfunktionen, um

```
public SparqlPOut query( SparqlPIn query );
```

Listing 6.8: Schnittstelle `GeoReferencedContextRetrieval`.

```
public SparqlPOut retrieve( SparqlPIn query );
```

Listing 6.9: Schnittstelle `NetworkRetrieval`.

ortsbezogene Kontextinformationen den Elementen eines räumlichen Netzwerks zuzuordnen. Zur Umsetzung dieses Verfahrens wurden folgende Komponenten identifiziert (vergleiche Abbildung 6.10):

GeoReferencedContextProvider Diese Komponente repräsentiert eine Quelle für ortsbezogene Kontextinformationen. Im Unterschied zu einer Komponente, die die `IdReferencedContextRetrieval`-Schnittstelle implementiert, sind die bereitgestellten Kontextinformationen keinem Überwachungsobjekt zugeordnet, sondern besitzen lediglich einen Ortsbezug. Deshalb implementiert diese Komponente auch die `GeoReferencedContextRetrieval`-Schnittstelle (siehe Listing 6.8). Über die Operation `query` können analog zur `IdReferencedContextRetrieval`-Schnittstelle Kontextinformationen mit Ortsbezug abgefragt werden. Der Ortsbezug wird repräsentiert wie in Abschnitt 6.6.1 beschrieben.

SpatialNetworkProvider Diese Komponente verwaltet ein räumliches Teil-Netzwerk in der Hierarchie von räumlichen Teil-Netzwerken. Über die `NetworkRetrieval`-Schnittstelle können andere Komponenten auf das Modell des Teil-Netzwerks zugreifen. Dazu akzeptiert die `retrieve`-Operation SPARQL-Anfragen, die mit Bezug zur ontologiebasierten Repräsentation des Teil-Netzwerks formuliert sind (vergleiche Abschnitt 6.6.1) und liefert als Antwort die entsprechende RDF-Repräsentation.

ContextAssigner Diese Komponente stellt ortsbezogene Kontextinformationen mit Bezug zu *Überwachungsobjekten* zur Verfügung. Sie bietet deshalb die `IdReferencedContextRetrieval`-Schnittstelle an. Zur Zuordnung der ortsbezogenen Kontextinformationen setzt sie das in Abschnitt 6.4 beschriebene Verfahren um. Dazu greift sie über die `GeoReferencedContextRetrieval`-Schnittstelle auf ortsbezogene Kontextinformationen zu. Das räumliche Netzwerk, auf das sich die netzbasierte Integration bezieht, wird über die `NetworkRetrieval`-Schnittstelle ausgelesen.

Mit diesen drei Komponenten können ortsbezogene Kontextinformationen bezüglich eines Infrastrukturnetzes integriert werden: Eine `ContextAssigner`-Komponente bezieht ortsbezogene Kontextinformationen von einer `GeoReferencedContextProvider`-Komponente. Sie nutzt das von einer `SpatialNetworkProvider`-Komponente zur Verfügung gestellte räumliche Teil-Netzwerk, um diese Kontextinformationen den Elementen des Teil-Netzwerks zuzuordnen. Dadurch stellt sie den Bezug zu Überwachungsobjekten her.

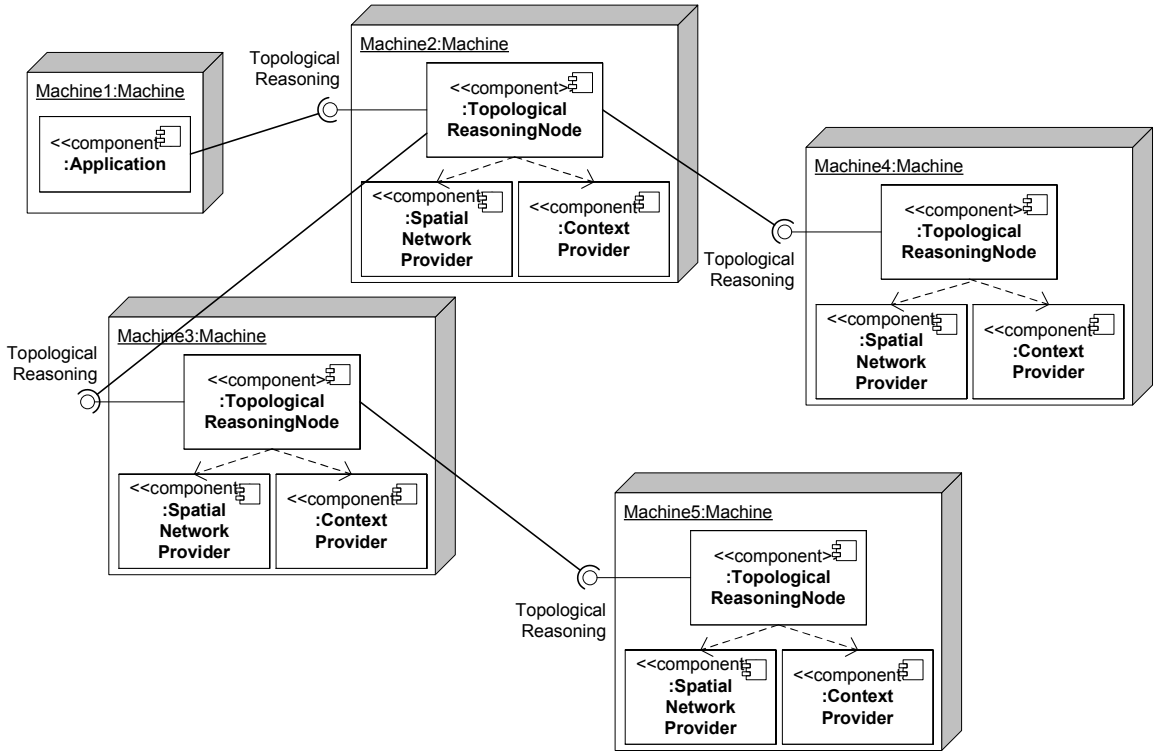

Abbildung 6.11.: Mögliche Verteilung der Komponenten zum Reasoning mit Topologiebezug.

```
public RDFgraph answerTopologicalQuery( SPARQLeR query );
```

Listing 6.10: Schnittstelle `TopologicalReasoning`.

Ein Vorteil ist, dass die `ContextAssigner`-Komponente die `IdReferencedContext-Retrieval`-Schnittstelle anbietet. Diese wurde auf Seite 111 bereits für das Framework zum Reasoning über verteilte Kontextinformationen definiert. Dadurch werden ortsbezogene Kontextinformationen auch für dieses Framework nutzbar (vergleiche Abschnitt 5.4.3).

6.6.3.2. Umsetzung eines topologiebezogenen Reasoning-Systems

Die vorgestellten Verfahren zum topologiebezogenen Reasoning bauen auf dem Konzept eines topologiebezogenen Reasoning-Systems auf, das in Abschnitt 6.5 eingeführt wurde. Zur Umsetzung dieser Verfahren muss also ein topologiebezogenes Reasoning-System realisiert werden. Dazu werden entsprechende Komponenten eingeführt.

Ein topologiebezogenes Reasoning-System besteht aus topologiebezogenen Reasoning-Knoten. Jeder topologiebezogene Reasoning-Knoten hat Zugriff auf ein räumliches Teil-Netzwerk des Infrastrukturnetz-Modells sowie auf Kontextinformationen zu den Elementen dieses räumlichen Netzes. Zur Realisierung eines topologiebezogenen Reasoning-Knotens werden deshalb folgende Komponenten eingeführt:

SpatialNetworkProvider Diese Komponente wurde bereits im vorigen Abschnitt zur Integration ortsbezogener Kontextinformationen beschrieben.

ContextProvider Diese Komponente repräsentiert eine Quelle für Kontextinformationen, die sich auf Überwachungsobjekte beziehen. Sie implementiert die `IdReferencedContextRetrieval`-Schnittstelle. Im Framework zum topologiebezogenen Reasoning stellt eine `ContextProvider`-Komponente Kontextinformationen zu den Infrastrukturelementen zur Verfügung.

TopologicalReasoningNode Diese Komponente realisiert einen topologiebezogenen Reasoning-Knoten und setzt die in Abschnitt 6.5 entwickelten Verfahren zum topologiebezogenen Reasoning um.

Sie bietet die `TopologicalReasoning`-Schnittstelle an (siehe Listing 6.10), über die Anwendungen topologiebezogene Anfragen stellen können. Diese werden unter Verwendung der entsprechenden Erweiterungen in SPARQLeR formuliert (vergleiche Abschnitt 6.6.2.2). Zur Kommunikation wird hier wieder das einfache, aber standardisierte SPARQL-*Protocol* eingesetzt [CFT08].

Zum Zugriff auf das Modell des Infrastrukturnetzes nutzt die `TopologicalReasoningNode`-Komponente die `NetworkRetrieval`-Schnittstelle; zum Zugriff auf Kontextinformationen die `IdReferencedContextRetrieval`-Schnittstelle.

Auf Basis dieser Komponenten kann das in Abschnitt 6.5 beschriebene topologiebezogene Reasoning-System wie folgt umgesetzt werden (vergleiche die Illustration in Anlehnung an UML-Verteilungsdiagramme in Abbildung 6.11):

Ein topologiebezogener Reasoning-Knoten wird durch eine `TopologicalReasoningNode`-Komponente repräsentiert. Diese erhält über eine `SpatialNetworkProvider`-Komponente Zugriff auf das zugeordnete räumliche Teil-Netzwerk. Sie nutzt eine `ContextProvider`-Komponente, um Kontextinformationen zu Elementen in diesem räumlichen Teil-Netzwerk abzufragen. Da eine `ContextProvider`-Komponente dadurch definiert ist, dass sie die `IdReferencedContextRetrieval`-Schnittstelle implementiert, kann beispielsweise auch das in Kapitel 5 beschriebene Framework für verteiltes Reasoning als `ContextProvider`-Komponente dienen.

Die Kommunikationsbeziehungen zwischen den topologiebezogenen Reasoning-Knoten ergeben sich aus der Hierarchie ihrer räumlichen Teil-Netzwerke: Sei G_j ein räumliches Teil-Netzwerk, das Grenzknoten eines räumlichen Teil-Netzwerks G_{j+1} verwendet. Dann besteht eine Kommunikationsbeziehung zwischen den ihnen zugeordneten `TopologicalReasoningNode`-Komponenten T_j und T_{j+1}. Diese Beziehungen werden in der Praxis in der Regel beim Aufsetzen des Systems konfiguriert.

6.6.4. Umsetzung der Verfahren zum topologiebezogenen Reasoning

Topologiebezogene Anfragen beziehen sich auf die gesamte Hierarchie von räumlichen Teil-Netzwerken. Sie können an eine beliebige `TopologicalReasoningNode`-Komponente gestellt werden. Zur Beantwortung der topologiebezogenen Anfragen ist die `TopologicalReasoningNode`-Komponente auf folgende Schnittstellen angewiesen:

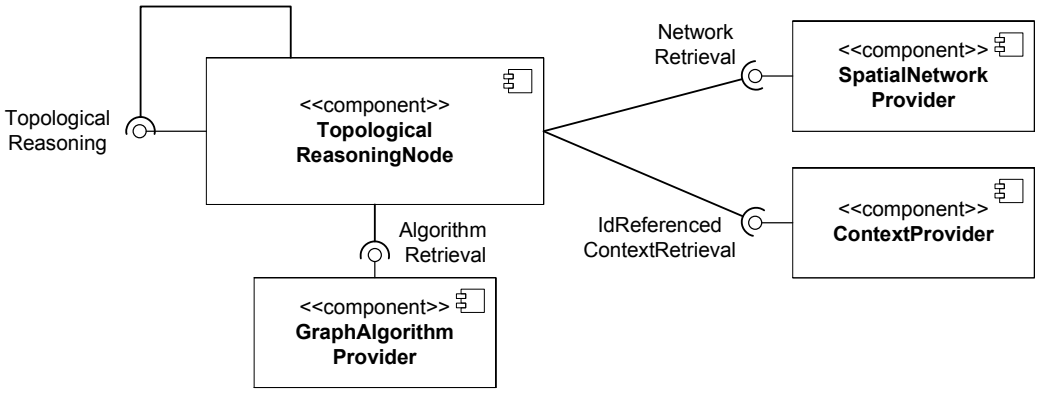

Abbildung 6.12.: Komponentenarchitektur für Reasoning mit Topologiebezug.

```
public Algorithm getAlgorithm( SPARQLeR query );
```

Listing 6.11: Schnittstelle `AlgorithmRetrieval`.

`TopologicalReasoning`, `NetworkRetrieval`, `IdReferencedContextRetrieval` und (hier zusätzlich eingeführt) `AlgorithmRetrieval`. Diese Abhängigkeiten sind auch in Abbildung 6.12 dargestellt.

Eine topologiebezogene Anfrage wird auf Basis dieser Komponenten wie folgt beantwortet: Eine Anwendung stellt eine topologiebezogene Anfrage an eine `TopologicalReasoningNode`-Komponente. Diese nimmt die Anfrage entgegen und koordiniert ihre Beantwortung. Je nach Anfragetyp müssen dafür unterschiedliche Verfahren eingesetzt werden (vergleiche Abschnitt 6.5). Um das Framework flexibel und erweiterbar zu halten, sind diese Algorithmen nicht Teil der Komponente, sondern werden bei Bedarf über die `AlgorithmRetrieval`-Schnittstelle der `GraphAlgorithmProvider`-Komponente abgerufen (siehe Listing 6.11). Dadurch wird es beispielsweise möglich, unterschiedliche Algorithmen für die Suche nach optimalen Pfaden einzusetzen. Algorithmen für die wichtigsten topologiebezogenen Anfragen wurden in Abschnitt 6.5 entwickelt.

Die `TopologicalReasoningNode`-Komponente führt den Algorithmus zur Beantwortung der topologiebezogenen Anfrage aus. In dessen Verlauf kann es nötig sein, Anfragen sowohl bezüglich des zugeordneten räumlichen Teil-Netzwerks als auch bezüglich anderer räumlicher Teil-Netzwerke in der Hierarchie zu beantworten.

Anfragen bezüglich des zugeordneten räumlichen Teil-Netzwerks. Um Teilanfragen bezüglich des zugeordneten räumlichen Teil-Netzwerks zu beantworten, nutzt die `TopologicalReasoningNode`-Komponente sowohl die `NetworkRetrieval`- als auch die `IdReferencedContextRetrieval`-Schnittstelle. Über die `NetworkRetrieval`-Schnittstelle hat sie Zugriff auf Topologie und Geometrie des räumlichen Teil-Netzwerks, während sie über die `IdReferencedContextRetrieval`-Schnittstelle Kontextinformationen zu Elementen des Teil-Netzwerks abfragen kann.

Auf dieser Basis kann die `TopologicalReasoningNode`-Komponente beispielsweise folgende Anfragen bezüglich ihres räumlichen Teil-Netzwerks beantworten:

- *Enthält das Teil-Netzwerk einen bestimmten Knoten oder eine bestimmte Kante?*

  Diese Anfrage kann direkt an die `NetworkRetrieval`-Schnittstelle gestellt werden.

- *Liegt ein Knoten oder eine Kante im Abdeckungsgebiet des Teil-Netzwerks?*

  Über die `NetworkRetrieval`-Schnittstelle werden Abdeckungsgebiet des Teil-Netzwerks und Lage des Knotens bzw. der Kante ermittelt. Dann wird geprüft, ob die Bedingung erfüllt ist.

- *Ist ein Polygon im Abdeckungsgebiet des Teil-Netzwerks vollständig enthalten? Überlappt ein Polygon mit dem Abdeckungsgebiet des Teil-Netzwerks?*

  Auch hier wird das Abdeckungsgebiet des Teil-Netzwerks über die `NetworkRetrieval`-Schnittstelle ermittelt und die Bedingung geprüft.

- *Welche Kanten enthält die transitive Hülle des Teil-Netzwerks?*

  Über die `NetworkRetrieval`-Schnittstelle wird die Topologie des Teil-Netzwerks abgefragt. Dann wird die transitive Hülle berechnet.

- *Wie verläuft ein Pfad zwischen zwei Knoten des Teil-Netzwerks, der optimal bezüglich eines Kontexttyps ist?*

  Über die `NetworkRetrieval`-Schnittstelle ist die Topologie des Teil-Netzwerks verfügbar, über die `IdReferencedContextRetrieval`-Schnittstelle werden die Gewichte der Kanten ermittelt. Dann wird ein Algorithmus zum Finden kürzester Wege eingesetzt, um einen optimalen Pfad zu bestimmen.

Anfragen bezüglich entfernter räumlicher Teil-Netzwerke. Um andere räumliche Teil-Netzwerke bei der Anfragebearbeitung zu berücksichtigen, stellt eine `TopologicalReasoningNode`-Komponente Anfragen an die `TopologicalReasoning`-Schnittstelle anderer `TopologicalReasoningNode`-Komponenten.

Diese Anfragen sind in SPARQLeR formuliert. Dadurch werden Kontext-Filter-Anfragen, Optimale-Pfade-Anfragen sowie Bereichs- und Nächste-Nachbarn-Anfragen (in modifizierter Form) an weitere `TopologicalReasoningNode`-Komponenten weitergeleitet.

Dabei kann die Ausführung eines Algorithmus auch zum Aufbau algorithmenspezifischer Datenstrukturen in einer `TopologicalReasoningNode`-Komponente führen: Bei der Suche nach einem garantiert optimalen Pfad müssen hierarchisch-kodierte Pfad-Sichten, bei der Suche nach fast-optimalen Pfaden Nachbarschaftsgraphen berechnet werden.

Beispiel für die Beantwortung einer atomaren Anfrage. Am Beispiel der atomaren Teil-Netzwerk-Identifikation (vergleiche Abschnitt 6.5.1) kann die Umsetzung eines atomaren Anfragetyps im vorgeschlagenen Framework beschrieben werden. Dabei seien die `TopologicalReasoningNode`-Komponenten wie in Abbildung 6.11 auf Seite 174 verteilt. Bezeichne T_k die `TopologicalReasoningNode`-Komponente auf Rechner k (Machine$k$):

Sei T_3 die `TopologicalReasoningNode`-Komponente, die die Anfrage nach einer Teil-Netzwerk-Identifikation für den Knoten Dresden erhält. (Befinde sich Dresden im räumlichen Teil-Netzwerk von T_4.) Zur Beantwortung dieser Anfrage formuliert T_3 eine SPARQLeR-Anfrage, die testet, ob der Knoten im Teil-Netzwerk bekannt ist. Diese wird über die `TopologicalReasoning`-Schnittstelle zunächst an T_5 auf der darunter liegenden Hierarchieebene weitergegeben. Da sie dort negativ beantwortet wird, reicht T_3 sie an T_1 auf der darüberliegenden Hierarchieebene weiter. T_1 kennt Dresden ebenfalls nicht und stellt die Anfrage deshalb an T_4. Da Dresden im räumlichen Teil-Netzwerk von T_4 gefunden wird, antwortet T_4 an T_1, und T_1 wiederum an T_3. Damit wurde T_4 als `TopologicalReasoningNode`-Komponente identifiziert, die Dresden verwaltet.

Beispiel für die Beantwortung einer komplexen Anfrage. Aufbauend auf den Umsetzungen der atomaren Anfragetypen werden in dem vorgeschlagenen Framework komplexe Anfragetypen realisiert. Dies wird am Beispiel einer Optimale-Pfade-Anfrage beschrieben. Dazu seien die `TopologicalReasoningNode`-Komponenten wieder wie in Abbildung 6.11 auf Seite 174 verteilt mit T_k die `TopologicalReasoningNode`-Komponente auf Rechner k (Machine$k$):

Sei T_3 die `TopologicalReasoningNode`-Komponente, die die Anfrage nach einem Pfad von Dresden nach München mit der geringsten Verspätungswahrscheinlichkeit erhält. T_3 kontaktiert die `GraphAlgorithmProvider`-Komponente, um einen geeigneten Algorithmus zu ermitteln. Da es sich bei „Verspätungswahrscheinlichkeit“ um einen hochdynamischen und aufwändig zu berechnenden Kontexttyp handelt, liefert die `GraphAlgorithmProvider`-Komponente den in Abschnitt 6.5.4.2 beschriebenen Algorithmus zur Suche nach fast-optimalen Pfaden zurück. Dieser wird im Folgenden ausgeführt.

Zuerst muss der kleinste gemeinsame Vorgänger der Teil-Netzwerke identifiziert werden, die Dresden und München enthalten. Dazu wird jeweils eine Anfrage zur Teil-Netzwerk-Identifikation für die Knoten München und Dresden formuliert (siehe oben). Für München wird T_5, für Dresden T_1 identifiziert. Folglich schickt T_3 die Ausgangsanfrage an T_2, weil dieses den kleinsten gemeinsamen Vorgänger darstellt.

T_2 ermittelt nun den kürzesten Pfad mittels Verfeinerung: Dazu berechnet T_2 den Nachbarschaftsgraphen und sucht darin einen kürzesten abstrakten Pfad vom räumlichen Teil-Netzwerk von T_3 zum räumlichen Teil-Netzwerk von T_4. Im diesem Beispiel bestehen direkte Verbindungen zwischen diesen Teil-Netzwerken.

Nun wird dieser abstrakte Pfad verfeinert und potentielle Teilpfade eines kürzesten Pfades von Dresden nach München ermittelt. Dazu formuliert T_2 SPARQLeR-Anfragen nach kürzesten Pfaden zwischen Dresden und allen Grenzknoten, über die das Teil-Netzwerk mit Dresden erreichbar ist. Diese werden über die `TopologicalReasoning`-Schnittstelle an T_4 gestellt. T_4 kann die Pfade lokal ermitteln und liefert sie als Antwort zurück.

Außerdem formuliert T_2 SPARQLeR-Anfragen nach kürzesten Pfaden zwischen allen Grenzknoten, über die das Teil-Netzwerk mit München erreichbar sind, und München. Diese werden über `TopologicalReasoning`-Schnittstelle an T_3 geschickt. T_3 muss analog dazu Anfragen an T_5 stellen, weil München erst dort bekannt ist. Schließlich werden die ermittelten kürzesten Pfade an T_2 zurückgegeben.

T_2 kombiniert die unterschiedlichen Teilpfade zu einem temporären Graphen und ermittelt darin den kürzesten Pfad von Dresden nach München. T_2 schickt diese Antwort an T_3, welches sie an die anfragende Anwendung weitergibt.

Die Beantwortung weiterer topologiebezogener Anfragetypen in einem topologiebezogenen Reasoning-System wurde in Abschnitt 6.5 beschrieben. Diese Verfahren lassen sich analog zu dem oben beschriebenen Verfahren für optimale Pfade ebenfalls mit den Komponenten des vorgeschlagenen Frameworks umsetzen.

6.7. Bewertung der Konzepte zum Reasoning mit Topologiebezug

Die in diesem Kapitel entwickelten Konzepte zum Reasoning mit Topologiebezug werden im Folgenden anhand der Anforderungen aus Abschnitt 6.1 bewertet.

A6.1 Anwendbarkeit auf verschiedenste Infrastrukturnetze. Das Konzept stellt an das Modell des Infrastrukturnetzes lediglich die Annahme, dass es als Hierarchie räumlicher Teil-Netzwerke modelliert ist. Dadurch kann es für unterschiedliche Infrastrukturdomänen eingesetzt werden. Konkret wurde die Anwendung für Schienen- und Stromnetze skizziert.

A6.2 Föderierte Administration der Infrastrukturtopologie. Das Konzept sieht vor, dass das Infrastrukturnetz auf Basis mehrerer räumlicher Teil-Netzwerke modelliert ist. Diese stehen über ihre Grenzknoten in einer hierarchischen Beziehung. Ihre interne Struktur ist jedoch gekapselt, weshalb Änderungen lokal vorgenommen werden können, ohne dass Konsistenzprobleme oder sonstige Auswirkungen auf andere Teil-Netzwerke entstehen. Dadurch lassen sich mit der vorgeschlagenen Modellierung des Infrastrukturnetzes die vorhandenen organisatorischen Strukturen bei der Infrastrukturüberwachung realitätsgetreu abbilden (vergleiche Abschnitt 6.3).

A6.3 Integration ortsbezogener Kontextinformationen. Es wurde ein generischer Ansatz entwickelt, um ortsbezogene Kontextinformationen (Geoobjekte mit geometrischer Komponente) auf Infrastrukturelemente (Geoobjekte mit geometrischer und topologischer Komponente) abzubilden. Dazu werden Wirkfunktionen eingesetzt, die für einen bestimmten Kontexttyp die distanzbezogene Relevanz kodieren (vergleiche Abschnitt 6.4.1). Damit können ortsbezogene Kontextinformationen den betrachteten Überwachungsobjekten zugeordnet werden, wodurch sie sowohl für topologiebezogenes Reasoning als auch Reasoning über verteilte Kontextinformationen nutzbar sind.

A6.4 Beantwortung unterschiedlicher topologiebezogener Anfragen. Das Konzept differenziert zwischen dem Modell des Infrastrukturnetzes, den zugeordneten Kontextinformationen und den topologiebezogenen Reasoning-Verfahren. Dadurch können unterschiedliche Reasoning-Verfahren auf dieselbe Modellierung des Infrastrukturnetzes und dieselben Kontextinformationen angewendet werden. Dies ist möglich,

weil atomare Anfragetypen identifiziert wurden, auf denen Verfahren für die Beantwortung komplexer Anfragetypen aufgebaut werden können. Abschnitt 6.5 hat exemplarisch demonstriert, wie damit bestehende Algorithmen zum Finden optimaler Wege, nächster Nachbarn und zur Beantwortung von Bereichsanfragen realisiert werden können. Dies wird auch bei der Umsetzung als Framework berücksichtigt, indem die Verwaltung der Graphenalgorithmen von ihrer Ausführung getrennt wird (vergleiche Abschnitt 6.6.4).

6.8. Zusammenfassung

In diesem Kapitel wurden Konzepte und Verfahren für Reasoning mit Topologiebezug entwickelt. Die Notwendigkeit dafür ergab sich, da Verfahren für Reasoning über verteilte Kontextinformationen zwar in der Lage sind, Kontextinformationen zu einzelnen Überwachungsobjekten zu integrieren. Viele topologiebezogene Anfragen, wie z. B. die Suche nach optimalen Pfaden oder k-nächsten Nachbarn, lassen sich damit jedoch nicht effizient umsetzen. Deshalb wurde das Konzept eines topologiebezogenen Reasoning-Systems entwickelt, das zwischen dem Modell des Infrastrukturnetzes, den netzbezogenen Kontextinformationen und den Reasoning-Verfahren für unterschiedliche Anfragetypen differenziert. Als generischer Modellierungsansatz für das Infrastrukturnetz wurde eine Hierarchie von räumlichen Teil-Netzwerken entwickelt. Zur Integration ortsbezogener Kontextinformationen wurden kontexttypspezifische Wirkfunktionen vorgeschlagen. Für das topologiebezogene Reasoning-System wurden atomare Anfragetypen identifiziert. Am Beispiel mehrerer topologiebezogener Anfragetypen wurde schließlich gezeigt, wie Verfahren für ihre Beantwortung auf Basis der atomaren Anfragetypen umgesetzt werden können. Auf diese Weise können auf Basis desselben Infrastrukturmodells und derselben Kontextinformationen unterschiedlichste Reasoning-Verfahren realisiert werden. Es wurde beschrieben, wie diese Konzepte in Form eines Frameworks umgesetzt werden können.

7. Integrierende Systemarchitektur und Anwendung

Bisher wurden Lösungsbeiträge zur Formalisierung der Semantik von Kontextinformationen und Infrastrukturzuständen sowie zur automatisierten Auswertung dieser Semantik vorgestellt. Dieses Kapitel entwickelt eine Systemarchitektur, mit der diese Konzepte in einem System umgesetzt und integriert werden können. Sie soll außerdem sicherstellen, dass ein solches System auch zukünftig um zusätzliche Verfahren erweitert werden kann.

Dazu formuliert Abschnitt 7.1 zunächst prinzipielle Anforderungen an eine solche Architektur. Abschnitt 7.2 stellt dann das funktional geschichtete Architekturkonzept vor und spezifiziert die Schnittstellen für jede Schicht. Es basiert auf den Ergebnissen in [FBLP09, FB08, FBP08, BFP08, BFP07, FHP+06]. Abschnitt 7.3 demonstriert schließlich die Anwendbarkeit des entwickelten Konzepts, indem er die Umsetzung eines Zustandsüberwachungssystems für die europäische Schieneninfrastruktur beschreibt (vergleiche [FBLP09, FLP+06]).

7.1. Anforderungen an die Systemarchitektur

Eine Systemarchitektur für die Zustandsüberwachung bei Infrastrukturnetzen muss den grundsätzlichen Anforderungen an solche Systeme genügen, die bereits in Abschnitt 2.3 auf Seite 20 identifiziert wurden. Bezogen auf die Systemarchitektur haben diese folgende Implikationen.

A7.1 *Verteilung der Systemkomponenten*: Systeme für die Zustandsüberwachung müssen geographisch große Gebiete abdecken (A2.5). Außerdem können sie praktisch nicht von einer einzelnen Organisation aufgebaut und betrieben werden (A2.6). Deshalb müssen sie verteilt realisierbar sein, und die Systemarchitektur muss dies unterstützen.

A7.2 *Zukunftssicherheit und Erweiterbarkeit*: Infrastrukturnetze sind sehr langlebig und werden permanent ausgebaut und verändert. Dies muss auch von einem integrierten Überwachungssystem unterstützt werden (A2.7). Aufgrund der vielen beteiligten Organisationen ist zudem davon auszugehen, dass zukünftig zusätzliche Organisationen und Kontextquellen eingebunden werden müssen (A2.5). Deshalb muss die Systemarchitektur für einen kontinuierlichen Ausbau und Erweiterbarkeit geeignet sein.

A7.3 *Robustheit*: Zustandsüberwachung für Infrastrukturnetze ist sehr komplex. Aufgrund der geographischen Verteilung des Systems (A2.5) und der vielen beteiligten Organisationen (A2.6) bestehen viele Abhängigkeiten. Zur Verringerung der

Fehleranfälligkeit muss die Systemarchitektur robust gestaltet sein und Ausfälle einzelner Komponenten kompensieren können.

7.2. Funktionale Schichten und Schnittstellen

Eine Systemarchitektur ist eine strukturierte oder hierarchische Anordnung der Systemkomponenten sowie die Beschreibung ihrer Beziehungen. Zur Zustandsüberwachung von Infrastrukturnetzen wird eine funktional geschichtete Struktur vorgeschlagen.

Ein schichtbasierter Ansatz hat den Vorteil, dass jede Schicht nur auf die Schnittstelle der darunter liegenden Schicht angewiesen ist und sonst keine Abhängigkeiten zu anderen Systemkomponenten bestehen. Deshalb spricht man von *loser Kopplung* der Komponenten. Diese erhöht die Robustheit, Flexibilität und Erweiterbarkeit des Systems. Dazu müssen die Schnittstellen der Schichten wohldefiniert sein.

Die folgenden Schichten werden unterschieden (vergleiche Abbildung 7.1):

Schicht 1. *Bereitstellung von Kontextinformationen*: Diese Schicht umfasst alle Quellen für Kontextinformationen. Komponenten dieser Schicht stellen die Kontextinformationen mit Bezug zur Systemontologie zur Verfügung. Dafür müssen sie diese von der (in der Regel) proprietären Repräsentation der Kontextquelle, die sie produziert, in die ontologiebasierte Repräsentation transformieren.

Schicht 2. *Management der Kontextinformationen*: Diese Schicht übernimmt das Datenmanagement des Überwachungssystems. Komponenten auf dieser Schicht speichern und verwalten Kontextinformationen, die mit Bezug zur Systemontologie repräsentiert sind, und stellen sie mit Bezug zu Überwachungsobjekten zur Verfügung. Sie erhalten die Kontextinformationen üblicherweise von den Kontextquellen auf Schicht 1. Auch weitere Daten, die von dem System benötigt werden, wie beispielsweise das konkrete Modell der Topologie und Geometrie eines Infrastrukturnetzes, werden von spezialisierten Komponenten auf dieser Schicht verwaltet.

Schicht 3. *Reasoning zur Zustandsüberwachung*: Diese Schicht ist für die Auswertung der Kontextinformationen zur Zustandsüberwachung zuständig. Komponenten auf dieser Schicht wenden unterschiedliche Verfahren an, um höherwertige Informationen, wie z. B. Infrastrukturzustände, automatisiert abzuleiten. Sie setzen dazu auf den von Schicht 2 zur Verfügung gestellten Kontextinformationen und Zusatzinformationen auf.

Schicht 4. *Zustandsbasierte Entscheidungsunterstützung*: Diese Schicht nutzt das integrierte Wissen über Infrastrukturzustände, um Betreiber der Infrastrukturen bei ihren Entscheidungen zu unterstützen. Dazu gehören Vorschläge zur Optimierung von Betriebs- und Wartungsabläufen wie auch Planungsunterstützung. Komponenten auf dieser Schicht setzen zu diesem Zweck unterschiedliche Verfahren ein. Als Grundlage nutzen sie jedoch die Auswertungsergebnisse, die von Schicht 3 zur Verfügung gestellt werden.

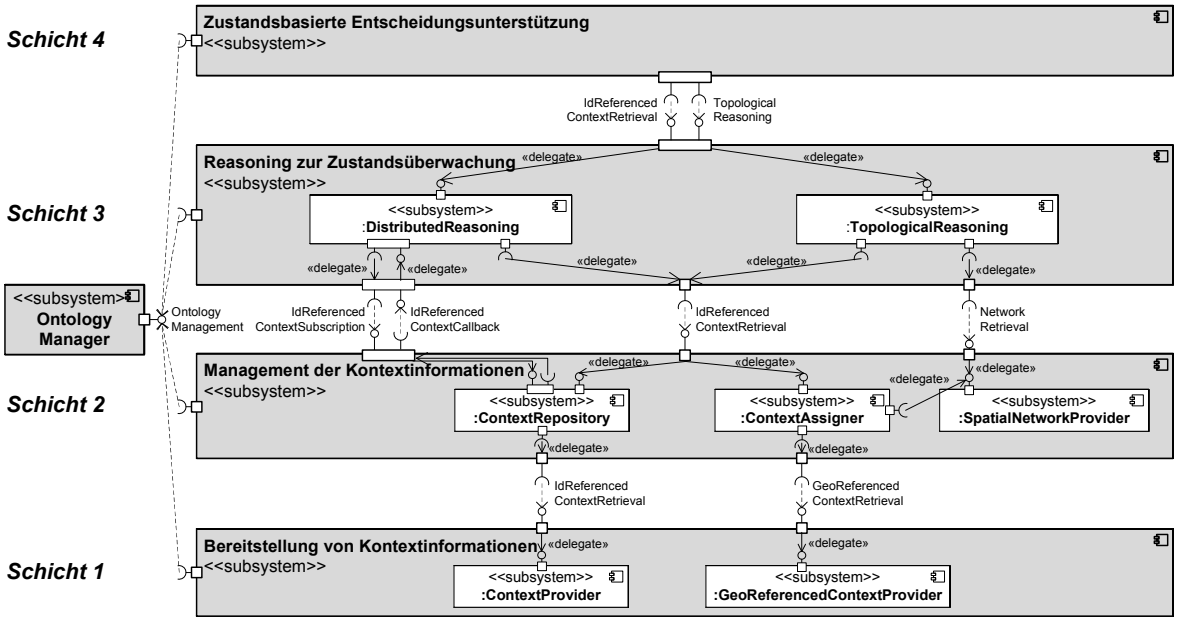

Abbildung 7.1.: Funktionale Schichten der integrierenden Systemarchitektur.

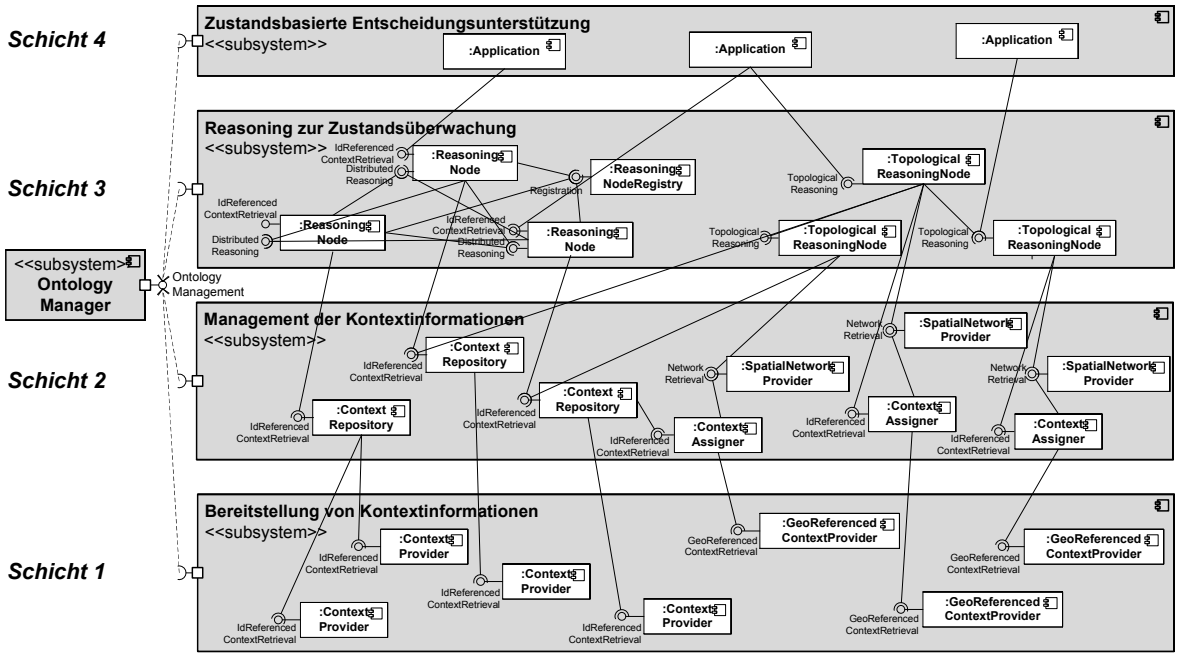

Abbildung 7.2.: Beispielhafte Realisierung der funktionalen Schichten mit den in dieser Arbeit eingeführten Komponenten.

Logisch getrennt von diesen funktionalen Schichten muss außerdem ein *Subsystem zum Ontologie-Management* (*OntologyManager*) angenommen werden. Dieses stellt die Systemontologie zur Verfügung und kümmert sich um ihre Versionierung. Da die Interoperabilität der funktionalen Schichten mittels der einheitlichen Systemontologie realisiert wird, greifen die Komponenten aller Schichten auf dieses Subsystem zu. Es steht damit logisch gesehen orthogonal zu den funktionalen Schichten. Da an anderer Stelle bereits detaillierte Konzepte zur Realisierung eines derartigen Subsystems zum Ontologie-Management entwickelt wurden (siehe z. B. [MMS+03]), wird dies im Rahmen dieser Arbeit nicht weiter ausgeführt.

Im Folgenden werden die funktionalen Schichten im Detail erläutert und ihre Schnittstellen spezifiziert. Diese sind auch in Abbildung 7.1 dargestellt. Abbildung 7.2 illustriert in Anlehnung an UML-Komponentendiagramme eine mögliche Implementierung dieser Schnittstellen mit den in dieser Arbeit eingeführten Komponenten.

7.2.1. Schicht 1: Bereitstellung von Kontextinformationen

Die erste Schicht der Systemarchitektur stellt Kontextinformationen bereit. Sie umfasst Komponenten, die Kontextquellen repräsentieren, wie z. B. (Fern-)Überwachungssysteme mit Sensoren, aber auch Datenbanken, die Ergebnisse manueller Inspektion erfassen. Zentrale Aufgabe dieser Schicht ist es, diese Kontextinformationen mit Bezug zur Systemontologie zu repräsentieren, um syntaktische und semantische Interoperabilität mit den anderen Komponenten des Systems zu erreichen. Dazu transformieren Schicht 1-Komponenten die Kontextinformationen von dem (in der Regel proprietären) Format der Kontextquelle in die ontologiebasierte Repräsentation.

Zusätzlich muss zwischen bezeichner- und ortsbezogenen Kontextquellen differenziert werden, da diese unterschiedliche Schnittstellen implementieren:

- *Bezeichnerbezogene Kontextquellen* liefern Kontextinformationen zusammen mit dem Bezeichner für das Überwachungsobjekt, auf das sie sich beziehen. Sie implementieren die `IdReferencedContextRetrieval`-Schnittstelle (siehe Listing 5.2 auf Seite 111). Dazu gehören beispielsweise Achslastgeber mit Zugidentifikation, die die Kontextinformation „Achslast“ zusammen mit dem Bezeichner des Zuges, für den diese Information ermittelt wurde, zurückgeben.

- *Ortsbezogene Kontextquellen* liefern Kontextinformationen mit einem Ortsbezug, d. h. der Lagekoordinate der Kontextinformation in einem räumlichen Bezugssystem. Sie implementieren die `GeoReferencedContextRetrieval`-Schnittstelle (siehe Listing 6.8 auf Seite 173). Ein Beispiel dafür ist ein Wetterdienst, der die Kontextinformation „Umgebungstemperatur“ mit Längen- und Breitengrad der Messung ermittelt.

7.2.2. Schicht 2: Management der Kontextinformationen

Die zweite Schicht verwaltet Kontextinformationen sowie sonstige relevante Informationen im Überwachungssystem. Dies umfasst ihre Sammlung, Archivierung und Bereitstellung. Je nach Art der Information werden dazu unterschiedliche Schnittstellen angeboten.

Kontextinformationen. Eine Aufgabe der Schicht ist es, anderen Komponenten Kontextinformationen zur Verfügung zu stellen. Eine solche `ContextRepository`-Komponente kann sowohl einen push- als auch einen pull-basierten Ansatz verfolgen:

- Für den *pull*-basierten Ansatz bieten Schicht 2-Komponenten die `IdReferencedContextRetrieval`-Schnittstelle an (siehe Listing 5.2 auf Seite 111). Diese wird von anderen Komponenten genutzt, indem sie eine beliebige SPARQL-Anfrage mit Bezug zur Systemontologie stellen und entsprechend des einfachen Anfrage-Antwort-Musters eine SPARQL-Antwort erhalten.

- Beim *push*-basierten Ansatz werden andere Komponente von einer Schicht 2-Komponente asynchron benachrichtigt, wenn bestimmte Kontextinformationen neu zur Verfügung stehen. Dazu bieten Schicht 2-Komponenten die `IdReferencedContextSubscription`-Schnittstelle an (siehe Listing 7.1). Über diese Schnittstelle registrieren andere Komponenten eine ebenfalls beliebige SPARQL-Anfrage für einen bestimmten Zeitraum. Ergeben sich in diesem Zeitraum neue Antworten für die registrierte Anfrage, so wird die Komponente, die diese Anfrage registriert hat, automatisch darüber benachrichtigt. Schicht 2-Komponenten erwarten, dass die registrierende Komponente dafür die `IdReferencedContextCallback`-Schnittstelle anbietet (siehe Listing 7.2).

Bevor Kontextinformationen zur Verfügung gestellt werden können, müssen sie von `ContextProvider`- bzw. `GeoReferencedContextProvider`-Komponenten auf Schicht 1 gesammelt werden. Diese müssen dazu zunächst ermittelt werden. Es ist davon auszugehen, dass die Menge dieser Komponenten dynamisch ist, weil neue Kontextquellen aufgrund von Mobilität, Komplexität und Erweiterungen hinzukommen und bestehende wegfallen. Deshalb werden geeignete Mechanismen benötigt, um diese Schicht 1-Komponenten zu ermitteln (*context provider discovery*). Dabei ist es sinnvoll, auch hier zwischen bezeichner- und ortsbezogenen Kontextquellen zu unterscheiden. Für die Ermittlung *bezeichnerbezogener Kontextquellen* bieten sich Verfahren an, wie sie in [FBT07, Ber06] entwickelt wurden, hier jedoch nicht im Detail vorgestellt werden können. Bei der Ermittlung von *ortsbezogenen Kontextquellen* sind aufgrund des Ortsbezugs andere Ansätze nötig, wie sie [Säm06a] entwickelt hat.

Wurde eine `ContextProvider`- bzw. `GeoReferencedContextProvider`-Komponente auf Schicht 1 identifiziert, müssen Kontextinformationen gesammelt und archiviert werden. Bei einer *bezeichnerbezogenen* Kontextquelle (`ContextProvider`) kann dazu die `IdReferencedContextRetrieval`-Schnittstelle genutzt werden. Bei einer *ortsbezogenen* Kontextquelle (`GeoReferencedContextProvider`) muss zunächst der Bezug zu den Überwachungsobjekten, also den Elementen des Infrastrukturnetzes, hergestellt werden. Dazu wird eine eigene `ContextAssigner`-Komponente angenommen, die das in Abschnitt 6.4 beschriebene Verfahren umsetzt. Diese bietet dann die `IdReferencedContextRetrieval`-Schnittstelle an, mittels derer `ContextRepository`-Komponenten auch ortsbezogene Kontextinformationen einbeziehen können.

Die `ContextRepository`-Komponenten übernehmen schließlich auch die Verwaltung und Archivierung der Kontextinformationen und ihrer Historie. Da große Datenmengen effizient verwaltet werden müssen, werden klassische Datenbankfunktionalitäten benötigt. Für die Verwaltung von ontologiebezogenen Kontextinformationen bieten

```
public RDFGraph subscribe( Sparql query );
```

Listing 7.1: Schnittstelle `IdReferencedContextSubscription`.

```
public void notify( RDFgraph );
```

Listing 7.2: `IdReferencedContextCallback`-Schnittstelle wie sie von Schicht 2-Komponenten erwartet wird.

sich RDF-*Repositories* bzw. *Triple Stores* an. Eine Vielzahl von Systemen, die auch mit großen Datenmengen umgehen können, sind bereits verfügbar[1] [MYQ+06, HLMS08]. Zunehmend werden auch etablierte relationale Datenbanksysteme mit der Unterstützung für RDF erweitert[2].

Topologie und Geometrie des Infrastrukturnetzes. Zusätzlich hat Schicht 2 die Aufgabe, weitere für die Zustandsüberwachung relevante Informationen zu verwalten. Dazu gehören auch Daten, die überwiegend statisch sind, wie beispielsweise die Topologie und Geometrie des Infrastrukturnetzes. Schicht 2-Komponenten bieten deshalb auch die `NetworkRetrieval`-Schnittstelle an (siehe Listing 6.9 auf Seite 173). Damit können andere Komponenten Netzmodelle abfragen und erhalten diese in der in Abschnitt 6.6.1 beschriebenen, ontologiebasierten Repräsentation.

Weitere relevante Informationen. Um bestimmte Auswertungsverfahren aufbauend auf Schicht 2 realisieren zu können, ist es denkbar, dass Schicht 2 neben Kontextinformationen und Infrastrukturnetz-Modellen weitere Arten von Informationen zur Verfügung stellen muss. Ein Beispiel sind Informationen über gesetzliche Vorschriften und Regulierungen bezüglich der Nutzung einzelner Teile des Infrastrukturnetzes, wie sie von einer Anwendung zur länderübergreifenden Planung von Gütertransporten benötigt würden. Für derartige Fälle sieht die Systemarchitektur vor, dass entsprechende Komponenten auf Schicht 2 eingeführt und geeignete Schnittstellen spezifiziert werden.

7.2.3. Schicht 3: Reasoning zur Zustandsüberwachung

Die dritte Schicht hat zur Aufgabe, Kontextinformationen automatisiert auszuwerten, die von Schicht 2 bereitgestellt werden. Je nach Zweck der Auswertung können dafür unterschiedliche Verfahren zum Einsatz kommen. Abhängig vom eingesetzten Verfahren unterscheiden sich auch die Schnittstellen der Komponenten.

In dieser Arbeit wurden zwei Ansätze zur Auswertung infrastrukturbezogener Kontextinformationen im Detail ausgearbeitet: Reasoning über verteilte Kontextinformationen und Reasoning mit Topologiebezug. Beide haben zum Ziel, die in der Systemontologie formalisierte Semantik automatisiert auszuwerten. Beim Reasoning mit Topologiebezug

[1] Beispiele für RDF-Repositories:
Sesame: http://www.openrdf.org/, OWLIM: http://www.ontotext.com/owlim/,
Joseki: http://www.joseki.org/, D2RQ: www4.wiwiss.fu-berlin.de/bizer/d2rq/

[2] z. B. Oracle 11g RDF, Boca für IBM DB/2

wird darüber hinaus die Topologie und Geometrie des Infrastrukturnetzes berücksichtigt.

`ReasoningNode`-Komponenten auf Schicht 3, die Reasoning über verteilte Kontextinformationen umsetzen, bieten der darüberliegenden Schicht die `IdReferencedContextRetrieval`-Schnittstelle an (vergleiche Kapitel 5). Im Unterschied zu Schicht 2-Komponenten, die ebenfalls die `IdReferencedContextRetrieval`-Schnittstelle anbieten, liefern diese Komponenten für dieselbe Anfrage in der Regel mehr Antworten. Denn sie werten nicht nur Zusammenhänge aus, die explizit zugesichert wurden, sondern erkennen auch implizite Zusammenhänge, die sich aus der Semantik in der Systemontologie ergeben. Die Kontextinformationen (verteilte Wissensbasis) werden jedoch nicht von den `ReasoningNode`-Komponenten selbst verwaltet. Stattdessen ist jeder `ReasoningNode`-Komponente eine `ContextRepository`-Komponente auf Schicht 2 zugeordnet. Eine `ReasoningNode`-Komponente nutzt dann die `IdReferencedContextRetrieval`-Schnittstelle, um die für das Reasoning benötigten Kontextinformationen von der `ContextRepository`-Komponente abzufragen. Dadurch ergeben sich zusätzliche Möglichkeiten, die Effizienz zu erhöhen: Beispielsweise können nur die neuesten Kontextinformationen berücksichtigt werden. Oder es werden, in Abhängigkeit von der Anfrage, immer nur Kontextinformationen zu Individuen abgefragt, die in der Anfrage erwähnt werden. Während die `ReasoningNode`-Komponente also immer nur eine kleine Untermenge aller verfügbaren Kontextinformationen verarbeiten muss, übernimmt Schicht 2 die verteilte Speicherung und Verwaltung sämtlicher Kontextinformationen und ihrer Historie.

Reasoning mit Topologiebezug wird auf Schicht 3 von `TopologicalReasoningNode`-Komponenten umgesetzt. Diese implementieren die `TopologicalReasoning`-Schnittstelle (vergleiche Kapitel 6). Wie `ReasoningNode`-Komponenten benötigen sie Zugriff auf topologiebezogene Kontextinformationen und nutzen dazu ebenfalls `ContextRepository`-Komponenten auf Schicht 2. Darüber hinaus nutzen sie `SpatialNetworkProvider`-Komponenten auf Schicht 2, um das räumliche Teil-Netzwerk des Infrastrukturnetzes auszulesen. Speicherung und Verwaltung dieser Daten wird ebenfalls von Schicht 2 übernommen.

Die vorgestellte Systemarchitektur ist nicht auf die oben genannten Auswertungsverfahren beschränkt: Weitere spezialisierte Verfahren können leicht integriert werden, indem entsprechende Komponenten auf Schicht 3 spezifiziert werden. Denkbar sind Ansätze zum *Data Mining* auf den vorhandenen Kontextinformationen, zur Lösung von *Optimierungsproblemen* oder auch *heuristische Auswertungsverfahren*. Je nach Verfahren können die Komponenten dazu bestehende Schnittstellen implementieren oder neue, verfahrensspezifische Schnittstellen definieren. Alle Erweiterungen können dabei auf den bereits vorhandenen Kontextinformationen aufbauen, die von Schicht 2 über die einheitliche `IdReferencedContextRetrieval`-Schnittstelle zur Verfügung gestellt werden und deren Semantik beschreibungslogisch formalisiert ist. Außerdem können mehrere Verfahren gleichzeitig und unabhängig voneinander eingesetzt werden.

7.2.4. Schicht 4: Zustandsbasierte Entscheidungsunterstützung

Die vierte Schicht umfasst Anwendungen, die Infrastrukturbetreiber bei ihren Entscheidungen unterstützen. Dazu nutzen sie höherwertige Kontextinformationen und

Infrastrukturzustände, die von Schicht 3 zur Verfügung gestellt werden.

Typische Anwendungsgebiete sind zustandsorientierte und vorausschauende Wartung der Infrastruktur sowie Optimierung des Betriebs, wie z. B. Planung und Störfallbehebung (vergleiche auch Kapitel 1). Dabei hängt die Umsetzung der Komponenten auf Schicht 4 vom Anwendungszweck ab und ist deshalb anwendungsspezifisch. Jede Anwendung trifft dazu Annahmen über die Schnittstellen, die von Schicht 3 implementiert werden. So kann es für eine Anwendung zur Priorisierung der Wartung von Schienenfahrzeugen ausreichen, die `IdReferencedContextRetrieval`-Schnittstelle zu nutzen. Eine Anwendung zur Wartung von Stromnetzen könnte hingegen auf die `IdReferencedContextRetrieval`- und `TopologicalReasoning`-Schnittstellen angewiesen sein.

Die konkrete Konzeption derartiger Anwendungen ist jedoch nicht Teil dieser Arbeit. Stattdessen stellen die hier entwickelten Konzepte die Grundlage zur Entwicklung dieser Anwendungen zur Verfügung.

7.3. Fallstudie im europäischen Schienennetz

Die vorgeschlagene Systemarchitektur bildet die Grundlage für die prototypische Realisierung eines Zustandsüberwachungssystems für den europäischen Schienenverkehr im Rahmen des europäischen Forschungsprojekts *Intelligent Integration of Railway Systems* (*InteGRail*) [IGR05]. Dabei werden auch die in Kapitel 4 und 5 vorgestellten Konzepte angewendet. Dieser Abschnitt beschreibt die Realisierung der Komponenten auf den verschiedenen Schichten sowie ihre Verknüpfung. Sie kommen auf verschiedenen Servern des Systems zum Einsatz, die über ganz Europa verteilt sind. Abbildung 7.3 gibt dazu einen Überblick in Anlehnung an UML-Verteilungsdiagramme. Die einzelnen Komponenten werden im Folgenden im Detail erklärt.

7.3.1. Systemontologie für Schienennetze

Im Rahmen des Forschungsprojekts haben Domänenexperten für eine Vielzahl von Überwachungssystemen Kontextinformationen und Infrastrukturzustände beschreibungslogisch modelliert. Beispiele sind Achslasten, Achslager, Waggontüren, der Bereitschaftsstatus (von Zugsystemen), der Lokomotivstatus, die Gleisgeometrie und das Schienenprofil. Dazu wurden die in Kapitel 4 vorgestellten Entwurfsmuster eingesetzt.

Die Struktur der entstandenen Systemontologie ist in Abbildung 7.4 illustriert. Sie nutzt die volle Ausdrucksstärke von $\mathcal{SHIN}$ und enthält insgesamt 748 Konzepte und 162 Rollen. Davon definieren die Kernontologien *InfrastructureMonitoringUpperOntology* und *BahnÜberwachungsObjekte* 377 Konzepte und 69 Rollen. Der Rest teilt sich auf die sieben Domänenontologien und die domänenübergreifend definierten Konzepte in *IntegrierteÜberwachungsObjektZustände* auf. Einen groben Überblick über die so modellierten Überwachungsobjekte, Merkmale, Symptome, Fehler und Zustände gibt Abbildung 7.5. Aufgrund des großen Umfangs der Ontologie kann die Abbildung die Details jedoch nur andeuten.

Die Nutzung des Ontologie-Entwurfsmusters ermöglichte es den Domänenexperten, die Domänenontologien unabhängig voneinander zu erstellen. Unter Verwendung

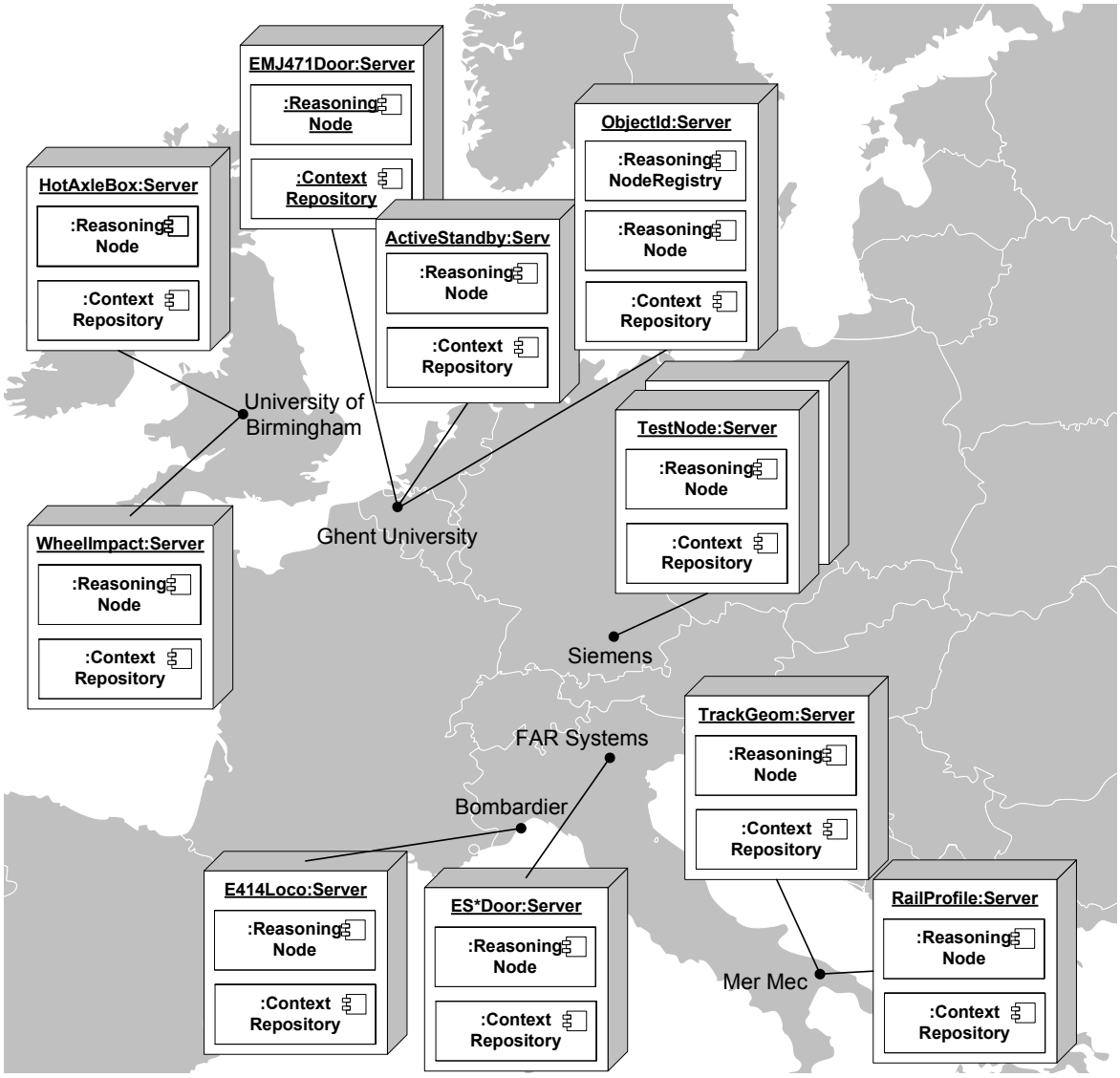

Abbildung 7.3.: Umsetzung von Schicht 2 und 3 der Systemarchitektur für Zustandsüberwachung im europäischen Schienennetz.

des Ontologie-Architekturmusters konnten diese anschließend zu einer konsistenten Systemontologie integriert werden. Da jede Domänenontologie das Entwurfsmuster instantiiert und deshalb Symptome, Fehler und Zustände definiert, konnten in *IntegrierteÜberwachungsObjektZustände* außerdem Überwachungsobjektkonzepte spezifiziert werden, die mit Bezug zu *mehreren* Domänen definiert sind (vergleiche beispielsweise RegularBogieMaintenanceTrain in Abschnitt 4.4.2). Dadurch wurden die zur Zustandsüberwachung relevanten Zusammenhänge zwischen Kontextinformationen für die automatisierte Auswertung formalisiert.

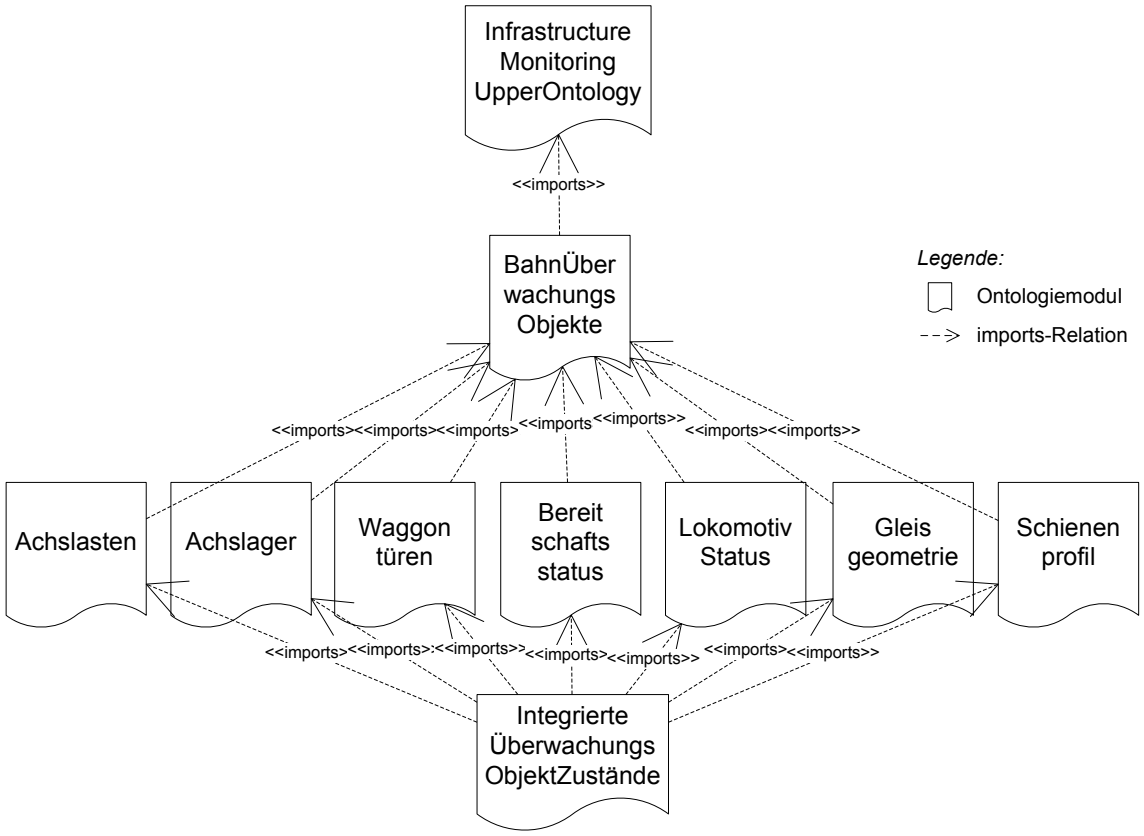

Abbildung 7.4.: Überblick über die Architektur der Ontologie für Zustandsüberwachung bei Schienennetzen.

7.3.2. Überwachungssysteme für Kontextinformationen

In den Domänenontologien sind Kontextinformationen modelliert, die für die Zustandsüberwachung relevant sind. Um diese zur Laufzeit erfassen zu können, mussten entsprechende Überwachungssysteme angebunden werden. Dazu wurden `ContextProvider`-Komponenten für folgende Überwachungssysteme entwickelt:

- *Achslastgeber* liefern Kontextinformationen zu Achslasten. Entsprechende Daten standen von den *WheelChex*-Achslastgeber zur Verfügung, die Network Rail in Großbritannien einsetzt und die von DeltaRail produziert werden.
- *Heißläufer-Ortungsanlagen* liefern Kontextinformationen zu Achslagern. Hier konnten ebenfalls Daten der von Network Rail in Großbritannien eingesetzten Systeme genutzt werden.
- *Türsensoren* liefern Kontextinformationen zu Waggontüren. Diese Daten wurden einerseits von der Tschechischen Bahn und Unicontrols für die von Skoda hergestellten Triebwägen EMJ471 („City Elephant“) und andererseits von FAR Systems für die Eurostar (ES)-Züge in Italien zur Verfügung gestellt.
- *Zugüberwachungssysteme* liefern Kontextinformationen zum Bereitschaftsstatus der Zugsysteme. Der Zugriff auf diese Daten wurde wieder von der Tschechischen

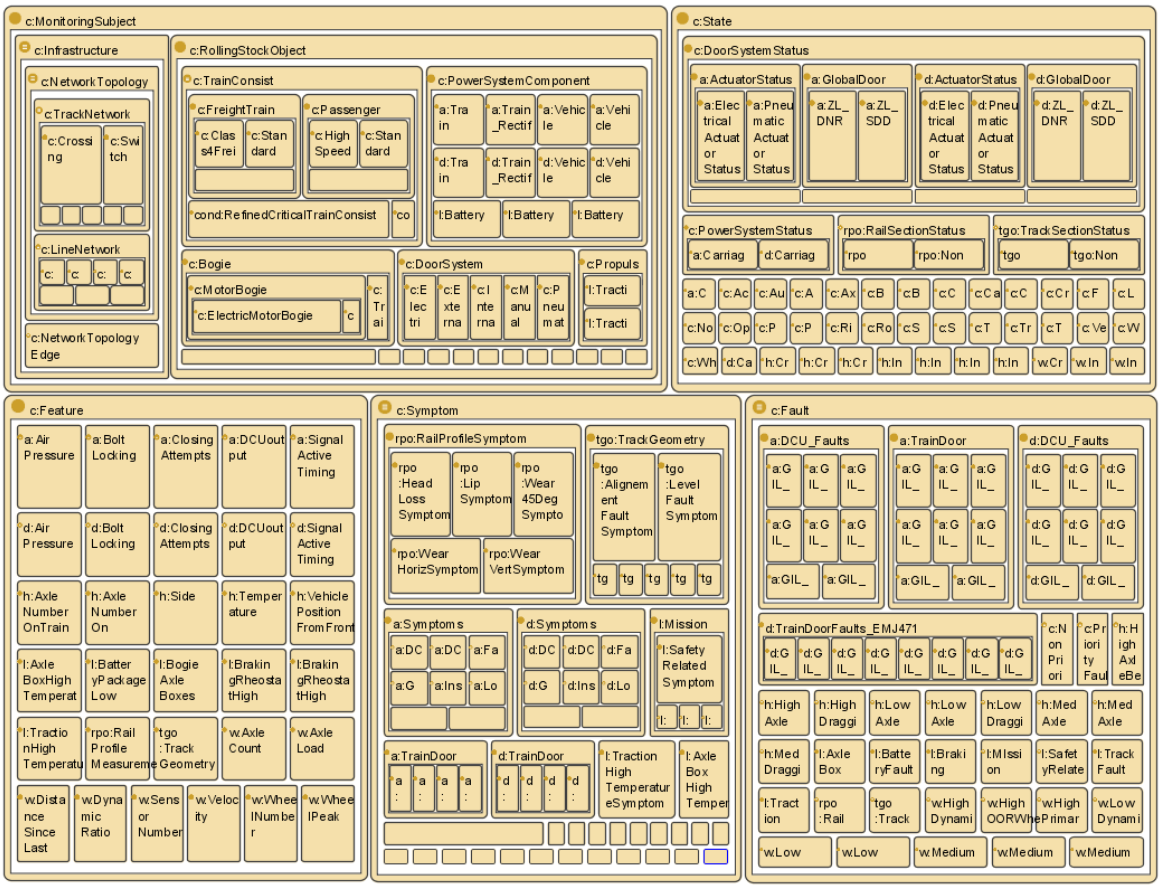

Abbildung 7.5.: Überblick über ausgewählte Überwachungsobjekt-, Zustands-, Merkmals-, Symptom- und Fehlerkonzepte der Ontologie für Zustandsüberwachung bei Schienennetzen. Die Darstellung wurde mit dem Ontologie-Editor *Protégé* generiert: Rechtecke repräsentieren Konzepte, ihre Schachtelung stellt die Konzeptinklusion dar.

Bahn und Unicontrols für die Triebwägen EMJ471 ermöglicht.

- *Lokomotivüberwachungssysteme* liefern Kontextinformationen zum Lokomotivstatus. Diese Daten wurden von Bombardier für E414-Lokomotiven der Eurostar-Züge zur Verfügung gestellt.
- *Optische Messsysteme* auf Messzügen liefern Kontextinformationen zu Gleisgeometrien. Derartige Daten konnte Mer Mec, ein italienischer Hersteller von Messsystemen, beitragen.
- Weitere *optische Messsysteme* auf Messzügen liefern Kontextinformationen zu Schienenprofilen. Diese wurden ebenfalls von Mer Mec zur Verfügung gestellt.

Jede dieser Komponenten bietet die `IdReferencedContextRetrieval`-Schnittstelle als W3C *Webservice* an. Die erfassten Kontextinformationen werden mit Bezug zur Systemontologie – genauer gesagt ihrer domänenspezifischen Erweiterung – repräsentiert. Die Transformation in diese Darstellung muss dabei jeweils spezifisch für das jeweilige

Überwachungssystem vorgenommen werden. Sollen weitere Überwachungssysteme desselben Typs integriert werden (beispielsweise Achslastgeber eines anderen Herstellers), muss lediglich diese Transformation angepasst werden; zur ontologiebasierten Repräsentation der Kontextinformationen sollten hingegen dieselben Konzepte und Rollen verwendet werden. Aus Gründen der Übersichtlichkeit sind diese `ContextProvider`-Komponenten in Abbildung 7.3 nicht dargestellt.

7.3.3. Management der Kontextinformationen

Zur Speicherung und Verwaltung von Kontextinformationen wurden verschiedene Varianten der `ContextRepository`-Komponente realisiert, die auf unterschiedlichen RDF-Repositories aufsetzen. Diese implementieren alle die `IdReferencedContextRetrieval`-Schnittstelle und bieten sie als W3C *Webservice* an. Zur Persistierung der Kontextinformationen werden Joseki/Jena, Sesame und D2RQ[3] als RDF-Repositories eingesetzt.

Jede `ContextRepository`-Instanz ist mit einer der oben beschriebenen `ContextProvider`-Instanzen auf Schicht 1 verknüpft. Sie fragt regelmäßig die aktuellsten Kontextinformationen ab und archiviert sie in dem integrierten RDF-Repository. Dazu wird die in Abschnitt 4.4.3 beschriebene Anwendung des Entwurfsmusters eingesetzt. Abbildung 7.3 zeigt die Verteilung der `ContextRepository`-Komponenten in Europa:

- Die University of Birmingham betreibt zwei `ContextRepository`-Komponenten mit Kontextinformationen zu Achslasten und Heißläufern.
- Die Ghent University bietet drei `ContextRepository`-Komponenten an, in denen jeweils Kontextinformationen zu den Waggontüren und dem Bereitschaftsstatus der EMJ471-Triebwägen sowie zur Identifikation von Zügen verwaltet werden.
- Bombardier betreibt in Vado Ligure `ContextRepository`-Komponenten mit Kontextinformationen zum Status von E414-Lokomotiven der Eurostar-Züge.
- FAR Systems stellt in einer `ContextRepository`-Komponente in Villafranca di Verona Kontextinformationen zum Status von Waggontüren der Eurostar-Züge zur Verfügung.
- Mer Mec bietet in `ContextRepository`-Komponenten in Monopoli Kontextinformationen zu Gleisgeometrien und Schienenprofilen an.

7.3.4. Reasoning über verteilte Kontextinformationen

Zur Auswertung der Kontextinformationen wurde auf Schicht 3 ein verteiltes Reasoning-System umgesetzt. Die `ReasoningNode`-Komponente bietet die `IdReferencedContextRetrieval`-Schnittstelle als W3C *Web Service* an und realisiert die in Kapitel 5 beschriebenen Verfahren zum Reasoning über verteilte Kontextinformationen (siehe auch die Implementierungsbeschreibung in Abschnitt 5.4).

[3] Joseki: http://www.joseki.org/, Sesame: http://www.openrdf.org/, D2RQ: www4.wiwiss.fu-berlin.de/bizer/d2rq/

Für die Platzierung der Komponenten wurde die in Abbildung 7.3 dargestellte Verteilung gewählt: Jeder `ContextRepository`-Instanz auf Schicht 2 ist eine `ReasoningNode`-Instanz auf Schicht 3 zugeordnet und nahe (bezüglich der Netztopologie) platziert. Die `ReasoningNode`-Instanz wird von der Organisation betrieben, die auch die zugeordnete `ContextRepository`-Instanz anbietet. Die in Kapitel 5 beschriebene `ReasoningNodeRegistry`-Komponente wurde in diesem Fall als zentrale Komponente an der Ghent University umgesetzt.

Wie bereits erwähnt, persistieren `ReasoningNode`-Instanzen die Kontextinformationen nicht selbst, sondern nutzen dazu die `ContextRepository`-Instanzen auf Schicht 2. Erhält eine `ReasoningNode`-Instanz eine Anfrage, verwendet sie die `IdReferencedContextRetrieval`-Schnittstelle, um von der zugeordneten `ContextRepository`-Instanz die jeweils aktuellsten Kontextinformationen abzufragen. Dadurch wird erreicht, dass immer die aktuellsten Daten berücksichtigt werden und dass nur ein kleiner Teil der gesamten archivierten Kontextinformationen für das Reasoning berücksichtigt werden muss.

7.3.5. Entscheidungsunterstützung für zustandsorientierte Instandhaltung

Mit der beschriebenen Umsetzung der Schichten 1, 2 und 3 wurde ein stabiler Prototyp für ein integriertes Zustandsüberwachungssystem geschaffen. Ziel des *InteGRail*-Projekts ist es, Mehrwert auf Basis bestehender Systeme zu bieten [IGR05]. Um dies nachzuweisen, wurden im Rahmen des Projekts mehrere Demonstratoren für typische Anwendungsfälle entwickelt [IGRP08]. Diese nutzen die von Schicht 3 bereitgestellte Funktionalität. Zwei von ihnen werden im Folgenden kurz vorgestellt.

Beispielsweise wurde eine Anwendung entwickelt, die den Schienennetz-Betreiber bei der Priorisierung der Gleisinstandsetzung unterstützt. Hier stellen also die Gleise die Überwachungsobjekte dar. In der Systemontologie sind verschiedene Gleiskonzepte, die die Dringlichkeit der Instandhaltung repräsentieren, mit Bezug auf Zustände modelliert. Um die Priorität für ein bestimmtes Gleis zu bestimmen, wird an eine beliebige `ReasoningNode`-Komponente auf Schicht 3 eine Anfrage zur Konzept-Realisierung der Gleis-Instanz gestellt. Entsprechend der Modellierung in der Systemontologie werden beim Reasoning dann automatisch die verteilt vorliegende Kontextinformationen zu Gleisgeometrie und Schienenprofil berücksichtigt. Das Ergebnis der Anfrage repräsentiert die Dringlichkeit der Gleisinstandsetzung und wird dem Anwender entsprechend visualisiert (siehe Screenshot in Abbildung 7.6).

Eine andere Anwendung hilft bei der Entscheidung, wie dringend bestimmte Züge gewartet werden müssen. Hier stellen Züge die Überwachungsobjekte dar. Auch hier sind Zugkonzepte in der Systemontologie spezifiziert, die die Dringlichkeit der Wartung implizieren. Die entsprechende Anfrage an eine beliebige `ReasoningNode`-Komponente auf Schicht 3 führt dazu, dass automatisch die verteilt vorliegenden Kontextinformationen von Achslastgebern und Heißläuferortungsanlagen berücksichtigt werden. Dem Anwender wird das Ergebnis – wie in Abbildung 7.7 dargestellt – visualisiert.

Abbildung 7.6.: Screenshot des Demonstrators für zustandsorientierte Instandhaltung des Schienennetzes [IGRP08]: In der links dargestellten Karte werden Gleisabschnitte ausgewählt. Rechts werden die verfügbaren Kontextinformationen und ein daraus abgeleiteter Gesamtzustand angezeigt. Zusätzlich kann der Ausfall bestimmter Kontextquellen simuliert werden, um zu demonstrieren, dass aufgrund der *open world assumption* dennoch keine falschen Ableitungen getroffen werden.

7.4. Bewertung der vorgestellten Systemarchitektur

Abschließend wird die funktional geschichtete Systemarchitektur vor dem Hintergrund der beschriebenen Fallstudie sowie ihrer konzeptionellen Eigenschaften bewertet. Dazu wird auf die zuvor in Abschnitt 7.1 aufgestellten Anforderungen Bezug genommen.

A7.1 Verteilung der Systemkomponenten. Die Systemarchitektur gibt lediglich funktionale Schichten und ihre Schnittstellen vor. Sie trifft keine Annahmen bezüglich der Verteilung von Komponenten. Wie die Architekturbeschreibung und auch die Fallstudie gezeigt haben, kann die Verteilungsentscheidung für jede Schicht separat getroffen werden. Die für Schicht 3 in den Kapiteln 5 und 6 entwickelten Reasoning-Verfahren sind explizit für eine verteilte Umsetzung entwickelt worden. Bei der Fallstudie wurden beispielsweise Schicht 1, 2 und 3 verteilt realisiert, während Schicht 4 zentral umgesetzt wurde (vergleiche Abschnitt 7.3.2 und Abbildung 7.3). Durch diese Flexibilität eignet

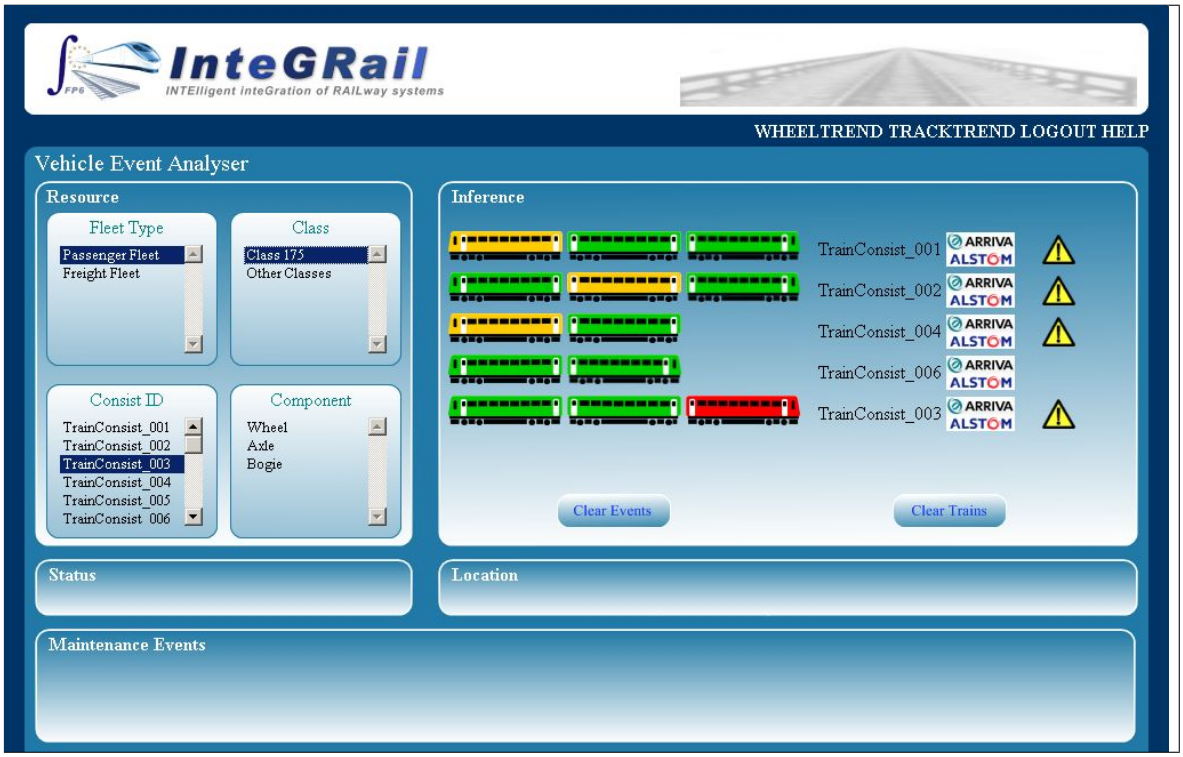

Abbildung 7.7.: Screenshot des Demonstrators für zustandsorientierte Wartung des Rollmaterials [IGRP08]: Auf der linken Seite werden Zugklassen, Züge oder Zugkomponenten ausgewählt. Rechts wird ihr Gesamtzustand angezeigt, der anhand der verfügbaren Kontextinformationen abgeleitet wird. Im unteren Bereichen werden Handlungsempfehlungen gegeben.

sich die vorgeschlagene Architektur auch für Systeme, die stark verteilt und an denen viele verschiedene Organisationen beteiligt sind.

A7.2 Zukunftssicherheit und Erweiterbarkeit. Die geschichtete Struktur der Systemarchitektur verringert die Abhängigkeiten zwischen Komponenten im System. Zudem sind die Schnittstellen klar spezifiziert und durch Verwendung der W3C-Standards Web Service, OWL, SPARQL und SPARQL-Protocol generisch einsetzbar. In Kombination führt beides dazu, dass die Architektur gut erweiterbar und damit zukunftssicher ist. So konnte die ursprüngliche Installation des Systems, die lediglich Achslastgeber und Heißläuferortungsanlagen integrierte, schrittweise um die anderen genannten Kontextquellen erweitert werden.

A7.3 Robustheit. Durch die geringen Abhängigkeiten zwischen den Komponenten im System ergibt sich eine lose Kopplung. Dies vereinfacht den Einsatz von Mechanismen zur Erhöhung der Robustheit. Derartige Mechanismen müssen jedoch schichtspezifisch umgesetzt werden. Beispielsweise ist das in Kapitel 5 vorgeschlagene Verfahren zum Reasoning über verteilte Kontextinformationen aufgrund seiner Dezentralität bereits sehr robust. So führt der Ausfall einer `ReasoningNode`-Komponente nicht dazu, dass keine Zustände mehr erkannt werden, sondern lediglich, dass die Erkennung ungenauer

wird (vergleiche Abschnitt 5.3.5). Diese Eigenschaft bewährte sich beispielsweise in der Aufbauphase des Systems, wo einzelne `ReasoningNode`-Komponenten zeitweise nicht verfügbar waren und das System dennoch Antworten lieferte. Dabei erhöht die Verwendung eines beschreibungslogischen Datenmodells, für das die *open world assumption* gilt, zusätzlich die Robustheit, weil dadurch verhindert wird, dass falsche Antworten abgeleitet werden.

7.5. Zusammenfassung

Dieses Kapitel stellte eine skalierbare und erweiterbare Systemarchitektur für die Zustandsüberwachung bei Infrastrukturnetzen vor. Mit ihrer Hilfe lassen sich alle in der Arbeit vorgestellten Konzepte zur semantischen Modellierung und zum Reasoning in einem Gesamtsystem umsetzen. Durch die Schichtung und präzise spezifizierten Schnittstellen lässt sich die Architektur um zusätzliche Verfahren erweitern.

Die praktische Anwendbarkeit der Architektur wurde durch eine Fallstudie im europäischen Schienenverkehr nachgewiesen: Mit Hilfe des Entwurfsmusters aus Kapitel 4 wurden Domänenontologien für Achslasten, Achslager, Waggontüren, den Bereitschaftsstatus von Zugsystemen, den Lokomotivstatus, die Gleisgeometrie und das Schienenprofil entwickelt. Zur Erfassung dieser Kontextinformationen wurden entsprechende Überwachungssysteme angebunden. Die Daten wurden in europaweit verteilten RDF-Repositories gesammelt. Die Auswertung erfolgte mittels der in Kapitel 5 entwickelten Verfahren zum Reasoning über verteilte Kontextinformationen. Anhand zweier Demonstratoren wurde beschrieben, wie sich Anwendungen diese Funktionalität zunutze machen können.

8. Zusammenfassung

Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen. Dazu werden in Abschnitt 8.1 die beschriebenen Lösungsbeiträge anhand der Forschungsfragestellungen noch einmal aufgegriffen und bezüglich der Ausgangsanforderungen aus Kapitel 2 beurteilt. Darauf aufbauend gibt Abschnitt 8.2 einen Ausblick auf zukünftige Erweiterungsmöglichkeiten und neu identifizierte Fragestellungen für zukünftige Arbeiten.

8.1. Ergebnisse

Die Themenstellung der Arbeit motiviert sich aus aktuellen und zukünftigen Herausforderungen an Infrastrukturnetze: Ihre Betreiber müssen immer höhere Kapazitäten bereitstellen, sie müssen ihre Netze verstärkt mit denen anderer Betreiber und über Landesgrenzen hinweg integrieren, und gleichzeitig muss die Kosteneffizienz von Betrieb und Wartung stetig gesteigert werden. Voraussetzung hierfür ist wesentlich präziseres Wissen über den aktuellen technischen Zustand der verschiedenen Infrastrukturkomponenten. Denn obwohl viele (Fern-)Überwachungssysteme auch heute schon in Infrastrukturnetzen im Einsatz sind, verhindern die semantische Heterogeneität der gesammelten Daten und die Beteiligung unterschiedlicher Organisationen ihre aussagekräftige Integration.

Die Ausgangsidee dieser Arbeit beruht auf dem Verständnis eines Zustandsüberwachungssystems als kontextsensitives System: Verschiedenartigste Kontextinformationen müssen integriert werden, um höherwertige Situationen bzw. Zustände von Überwachungsobjekten abzuleiten. Während bei kontextsensitiven Systemen auf dem Gebiet der *Ambient Intelligence* dazu in letzter Zeit erfolgreich Techniken der formalen Wissensrepräsentation eingesetzt werden, gibt es bei Zustandsüberwachungssystemen dazu bisher keine bzw. lediglich Ad-hoc-Ansätze. Das führt unter anderem dazu, dass die Erweiterbarkeit um zusätzliche Kontextquellen nicht oder nur mit erheblichem Aufwand möglich ist und dass kein Informationsaustausch zwischen den beteiligten Organisationen stattfindet. Um eine formale Grundlage für die Integration beliebiger Überwachungssysteme zu schaffen, wendet diese Arbeit deshalb Techniken der formalen Wissensrepräsentation für die Zustandsüberwachung in Infrastrukturnetzen an. Die Voraussetzungen dafür wurden erst durch aktuelle Fortschritte im Bereich der *Semantic Web*-Technologien geschaffen.

8.1.1. Beantwortung der Forschungsfragestellungen

Zur Zusammenfassung der in dieser Arbeit beschriebenen Lösungsbeiträge werden im Folgenden die in Kapitel 1 auf Seite 6 formulierten Forschungsfragestellungen aufgegriffen. Diese beziehen sich auf die Bereiche Modellierung, Auswertungsverfahren und praktische Umsetzung.

Als Grundlage stellte Kapitel 2 die Herausforderungen und den aktuellen Stand der Technik bei der Zustandsüberwachung für Infrastrukturnetze vor. Dazu wurden exemplarisch Schienennetze und Stromnetze herausgegriffen. Auf dieser Basis konnten generische Anforderungen abgeleitet werden, wie beispielsweise die Notwendigkeit, bei der *Modellierung* mehrere Domänenexperten einzubeziehen, bei der *Auswertung* unvollständige und verteilte Kontextinformationen zu berücksichtigen und bei der *praktischen Umsetzung* gute Erweiterbarkeit und Wartbarkeit sicherzustellen.

Kapitel 3 gab einen Überblick über die zentralen Konzepte der logikbasierten Wissensrepräsentation und -verarbeitung und führte sowohl Beschreibungslogiken als auch die darauf aufbauenden *Semantic Web*-Standards wie OWL DL ein. Damit wurde die formale Basis der Arbeit gelegt.

Zur *Modellierung* von Kontextinformationen und Infrastrukturzuständen wurden in Kapitel 4 *Entwurfsmuster* konzipiert. Diese wurden auf Basis internationaler Standards in diesem Bereich gestaltet, um ihre breite Anwendbarkeit zu gewährleisten. Gleichzeitig wurde darauf geachtet, dass damit erstellte Ontologien eine hohe Schlussfolgerbarkeit besitzen, was essenziell für die automatisierte Auswertung ist. Der musterbasierte Ansatz ermöglicht es Domänenexperten auch ohne Modellierungskompetenz, qualitativ hochwertige Ontologien in ihrem Wissensgebiet zu entwickeln.

Zur *Auswertung* der so formalisierten Semantik müssen die Besonderheiten von Infrastrukturnetzen berücksichtigt werden. Dazu gehört vor allem, dass Kontextinformationen verteilt anfallen und verteilt gehalten werden. Bestehende Reasoning-Verfahren sind dafür nicht einsetzbar, weil sie von zentral verfügbaren Daten ausgehen. Deshalb wurde in Kapitel 5 ein Konzept für das *Reasoning über verteilte Kontextinformationen* entwickelt, mit dem konjunktive Anfragen korrekt und vollständig für verschiedenartig verteilte Wissensbasen beantwortet werden können. Dies gilt aufgrund der *open world assumption* auch beim Ausfall von Reasoning-Knoten. Dieser Ansatz wurde sowohl theoretisch analysiert als auch seine praktische Umsetzbarkeit und Vorteilhaftigkeit anhand einer prototypischen Implementierung und Evaluierung nachgewiesen.

Die Ausdrucksstärke konjunktiver Anfragen reicht jedoch nicht zur Formulierung topologiebezogener Anfragen aus, die ebenfalls bei der Infrastrukturüberwachung benötigt werden. Bestehende Reasoning-Verfahren bietet dafür ebenfalls keine Unterstützung. Deshalb hat diese Arbeit in Kapitel 6 einerseits einen generischen Modellierungsansatz für Infrastrukturnetze entworfen und andererseits ein Konzept für das *Reasoning mit Topologiebezug* entwickelt, auf dessen Basis unterschiedliche topologiebezogene Anfragetypen beantwortet werden können. Dies wurde unter anderem für die Suche nach optimalen Pfaden und nächsten Nachbarn demonstriert.

Schließlich war auch die *praktische Umsetzbarkeit* der entwickelten Konzepte eine zentrale Fragestellung. Bei der Modellierung wurde dies anhand von Beispielen demonstriert, bei den Auswertungsverfahren wurden konkrete Komponenten für die Implementierung beschrieben. Kapitel 7 hat schließlich eine funktional geschichtete Systemarchitektur vorgestellt, mit deren Hilfe die beschriebenen Konzepte integriert und in Zukunft auch um zusätzliche Konzepte erweitert werden können. Die Praxistauglichkeit dieser Systemarchitektur wurde anhand eines prototypisch realisierten Systems zur Zustandsüberwachung im europäischen Schienenverkehr nachgewiesen. In diesem Rahmen erstellten mehrere Domänenexperten auf Basis der entwickelten Modellierungsmuster die erste umfassende Ontologie für Kontextinformationen und Infrastrukturzustände im

Schienenverkehr. Zu den bisher sieben berücksichtigten Domänen gehören beispielweise Achslasten, Waggontüren, Gleisgeometrien und Schienenprofile. Für die prototypische Umsetzung des integrierten Zustandsüberwachungssystems stellten mehrere Projektbeteiligte – von Network Rail in Großbritannien bis zu Unicontrols in Tschechien – Kontextinformationen für diese Domänen zur Verfügung. Diese wurden in mehreren europaweit verteilten Repository-Komponenten gesammelt. Darauf aufbauend wurde ein verteiltes Reasoning-System realisiert, dessen Reasoning-Knoten ebenfalls verteilt und jeweils bei einer Repository-Komponenten platziert sind. Auf dieser Basis können nun Anwendungen für die integrierte Zustandsüberwachung umgesetzt werden. Bisher wurden die zustandsorientierte Wartung des Rollmaterials und des Schienennetzes demonstriert.

8.1.2. Bewertung der Lösungsbeiträge bezüglich der Ausgangsanforderungen

Zur abschließenden Beurteilung der vorgeschlagenen Konzepte werden die in Kapitel 2 in Abschnitt 2.3 identifizierten generischen Anforderungen an die Zustandsüberwachung in Infrastrukturnetzen herangezogen:

A2.1 Integration von Kontextinformationen. Zentrale Herausforderung war, ein generisches Konzept zur Integration beliebiger Kontextinformationen zu entwickeln. Diese Arbeit schlägt dazu einen wissensbasierten Ansatz vor, bei dem die Semantik der Kontextinformationen maschinenverarbeitbar formalisiert wird, um sie automatisiert auswerten zu können. Die Erstellung geeigneter Wissensmodelle wird in Kapitel 4 behandelt, während Kapitel 5 und 6 Konzepte zur automatisierten Auswertung entwickeln.

A2.2 Unterstützung komplexer Zusammenhänge. Um praktikabel zu sein, muss der gewählte Ansatz ausdrucksstark genug sein, um die komplexen Zusammenhänge zwischen Kontextinformationen und Infrastrukturzuständen modellieren zu können. Diese Arbeit nutzt dafür $\mathcal{SHIN}$, welche einerseits eine sehr ausdrucksstarke Beschreibungslogik und die wesentliche Basis für den W3C Standard OWL DL darstellt, andererseits aber dennoch entscheidbar ist. Kapitel 4 konzipiert dazu Entwurfsmuster, die die Modellierung komplexer Zusammenhänge auf Basis von $\mathcal{SHIN}$ erleichtern. Angewendet werden diese Entwurfsmuster beispielsweise für die in Kapitel 7 beschriebene Fallstudie.

A2.3 Modellierung durch unterschiedliche Domänenexperten. Aufgrund der Komplexität einer Infrastrukturdomäne muss davon ausgegangen werden, dass das Domänenwissen über viele verschiedene Experten verteilt ist. Dazu muss es möglich sein, mehrere Experten an der Modellierung zu beteiligen. In Kapitel 4 wird ein Architekturmuster entwickelt, mit dessen Hilfe auch mehrere Domänenexperten gemeinsam qualitativ hochwertige Ontologien entwickeln können. Die Wahl der $\mathcal{SHIN}$-Beschreibungslogik als formale Basis ermöglicht es darüber hinaus, automatisiert die logische Konsistenz einer Systemontologie zu prüfen und damit Modellierungskonflikte frühzeitig zu erkennen.

A2.4 Umgang mit unvollständigen Daten. Bei der Infrastrukturüberwachung kann einerseits nicht vorausgesetzt werden, dass alle Kontextinformationen immer vollständig verfügbar sind. Andererseits dürfen trotz fehlender Informationen keine falschen Ableitungen getroffen werden. Auch vor diesem Hintergrund eignen sich Beschreibungslogiken zur Formalisierung der Semantik, weil sie die *open world assumption* verwenden und damit prinzipiell von unvollständigen Informationen ausgehen. Fehlen Informationen, kommt es dadurch nicht zu falschen Ableitungen, sondern es werden trotzdem die – auf Basis der verfügbaren Informationen – bestmöglichen Schlussfolgerungen gezogen.

A2.5 Unterstützung unterschiedlicher Organisationen. Am Betrieb eines Infrastrukturnetzes sind viele Organisationen beteiligt, denen es möglich sein muss, Kontextinformationen zu integrieren und auszutauschen. Auch hier eignet sich der wissensbasierte Ansatz dieser Arbeit, weil die Semantik von Kontextinformationen explizit in Wissensmodellen (Ontologien) formalisiert wird. Deshalb können Organisationen mittels Austauschs dieser Ontologien semantische Interoperabilität erreichen.

A2.6 Eignung für räumlich stark verteilte Systeme. Ihre geographische Ausdehnung und die große Zahl beteiligter Organisationen führt bei Infrastrukturnetzen dazu, dass Überwachungssysteme räumlich stark verteilt sind. Dies ist untypisch für wissensbasierte Systeme, die üblicherweise von einem zentralisierten System ausgehen. Diese Arbeit entwickelt in Kapitel 5 und 6 deshalb neuartige Konzepte, bei denen die Auswertungsverfahren auch verteilt ablaufen können. Auch die in Kapitel 7 vorgeschlagene Systemarchitektur sieht vor, dass einzelne Schichten verteilt realisiert werden können, was anhand des Fallbeispiels zum europäischen Schienenverkehr praktisch demonstriert wird.

A2.7 Eignung für lange Lebensdauer und permanenten Ausbau. Die lange Lebensdauer von Infrastrukturnetzen führt dazu, dass Überwachungssysteme sowohl einfach wartbar als auch leicht erweiterbar sein müssen. Auch hier zahlt sich ein wissensbasierter Ansatz aus, weil das domänenspezifische Wissen in Form des Wissensmodells (Ontologie) dort explizit vom domänenunabhängigen Reasoning-Algorithmus getrennt ist. Deshalb ist zur Wartung oder Erweiterung des Systems lediglich eine Veränderung des Wissensmodells notwendig, was auch zur Laufzeit geschehen kann. Dabei ist vorteilhaft, dass die $\mathcal{SHIN}$-Beschreibungslogik das Wissen deklarativ repräsentiert, was die Verständlichkeit für den Menschen erhöht. Außerdem erleichtert die Struktur des in Kapitel 4 entwickelten Entwurfsmusters die Vornahme von Änderungen. Auch hier können logische Modellierungsfehler automatisiert erkannt werden.

Insgesamt zeigt diese Arbeit, wie ein wissensbasierter Ansatz in Kombination mit dem Verständnis eines Zustandsüberwachungssystems als kontextsensitives System erstmals ein semantisch integriertes Zustandsüberwachungssystem für Infrastrukturnetze ermöglicht, das einfach wartbar und erweiterbar ist. Dazu wurde sowohl auf den neuesten Entwicklungen im Bereich des *Semantic Web* als auch auf Techniken der *Ambient Intelligence* aufgebaut. Neben einer wohldefinierten theoretischen Grundlage wurde immer auch die praktische Umsetzbarkeit der Verfahren sichergestellt.

8.2. Ausblick

Dieser Abschnitt stellt einige Fragestellungen vor, die – ausgehend von der Themenstellung dieser Arbeit – Potential für darauf aufbauende Forschungsvorhaben bieten.

Auf dem Gebiet der Modellierung werden dringend allgemeingültige Ansätze zur systematischen Beurteilung der Qualität von Ontologien benötigt. Dazu müssen zunächst geeignete Metriken identifiziert werden. Diese werden auch abhängig vom Anwendungszweck sein. Einige qualitative Kriterien wurden bereits in Kapitel 4 vorgestellt. Mit Hilfe entsprechender Metriken könnte dann die Qualität von Ontologien, die auf Basis von Entwurfsmustern erstellt wurden, mit der Qualität frei erstellter Ontologien verglichen werden. Dadurch wäre es möglich, den Nutzen des musterbasierten Ansatzes zu quantifizieren.

Eine weitere interessante Untersuchung wäre, mögliche Erweiterungen der Ausdrucksstärke über die hier verwendete $\mathcal{SHIN}$-Beschreibungslogik hinaus zu evaluieren. Hier steht die Abwägung zwischen dem erzielten Nutzen und der damit verbundenen Erhöhung der Berechnungskomplexität im Mittelpunkt. Ansatzpunkte sind einerseits noch ausdrucksstärkere Beschreibungslogiken wie beispielsweise $\mathcal{SROIQ}$ [KHS06]. Aber auch probabilistische Erweiterungen für Beschreibungslogiken sind vielversprechend für die Modellierung nicht eindeutiger Zusammenhänge. Derartige Erweiterungen werden zunehmend untersucht [Luk07, Hub06] und wurden inzwischen erstmals auch in eine der etablierten Reasoner-Implementierungen integriert[1]. Für Anwendungen, bei denen die Schnelligkeit der Antwort besonders wichtig ist, kann andererseits auch eine Beschränkung der Ausdrucksstärke der Modellierungssprache interessant sein. Denn damit ergibt sich in der Regel eine geringere Komplexität des Reasonings [HLMS08]. Ein Beispiel hierfür ist die Beschreibungslogik $\mathcal{EL}^{++}$, für die die grundlegenden Reasoning-Probleme im Wesentlichen in polynomieller Zeit entscheidbar sind [BBL08].

Im Bereich der Auswertungsverfahren kann die Vielseitigkeit des vorgestellten Ansatzes besonders genutzt werden, um zusätzliche Auswertungsverfahren zu konzipieren. Diese sollten auf den semantisch modellierten Kontextinformationen aufsetzen und spezialisierte Probleme lösen. Denkbar sind beispielsweise Trendvorhersagen, die die Veränderungen von Kontextinformationen über die Zeit auswerten, oder numerische Modelle, die bestimmte Systeme modellieren und damit Aussagen treffen können, die über die logik-basierten Ableitungsmöglichkeiten hinaus gehen. Gleichzeitig kann die hohe Datenqualität der vorhandenen Kontextinformationen zur Informationsextraktion genutzt werden. So ist es vielversprechend, *Data Mining*-Verfahren einzusetzen, um bislang unbekannte Zusammenhänge zwischen (beispielsweise) Symptomen und Fehlern zu entdecken. Mit diesem Wissen können wiederum die Ontologien angepasst werden.

Auch innerhalb der realisierten Frameworks für das Reasoning über verteilte Kontextinformationen und das Reasoning mit Topologiebezug sind Erweiterungen denkbar. Die Voraussetzungen dafür wurden bereits bei der Konzeption dieser Frameworks geschaffen. Für das Reasoning über verteilte Kontextinformationen kann die Effizienz der Bearbeitung konjunktiver Anfragen beispielsweise besonders durch die Verbesserung der Anfrageplan-Optimierung erhöht werden. Bei der Entwicklung geeigneter Heuristiken

[1] Pronto – Probabilistische Erweiterung für Pellet: http://pellet.owldl.com/pronto

kann auf umfangreicher Literatur aufgesetzt werden [Cha98]. Beim Reasoning mit Topologiebezug bietet es sich an, weitere topologiebezogene Anfragetypen umzusetzen, wie beispielsweise nächste Paare (engl. *closest-pairs*) und entfernungsbezogene Joins (engl. *e-distance joins*) [PZMT03].

Über den bestehenden prototypischen Einsatz im europäischen Schienenverkehr hinaus ist es vor einem Produktiveinsatz nötig, weitere praktische Erfahrungen mit den vorgeschlagenen Konzepten zu sammeln. Dazu gehört eine genauere Untersuchung des Systemverhaltens unter Last und die Entwicklung eines detaillierten Betriebskonzepts. Beides hätte den Rahmen dieser Arbeit deutlich überschritten.

Stattdessen hat die Arbeit theoretisch fundierte Grundlagen für die Realisierung eines derartigen Systems gelegt und neuartige Konzepte für semantische Modellierung und Reasoning für Kontextinformationen in Infrastrukturnetzen entwickelt und umgesetzt.

Anhang A.

LeHigh University Benchmark

Dieser Anhang enthält ergänzende Informationen zum *LeHigh University Benchmark* (LUBM). Dieser wurde in Abschnitt 5.5 zur Evaluierung der Verfahren für Reasoning über verteilte Kontextinformationen eingesetzt.

A.1. Test-Anfragen des Benchmarks

Dieser Abschnitt enthält die 14 Test-Anfragen des *LeHigh University Benchmarks* [GPH05].

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:GraduateStudent.
        ?x ub:takesCourse
                <http://www.Department0.University0.edu/
                    GraduateCourse0>.
}
```

Listing A.1: LUBM-Anfrage 1.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x, ?y, ?z
WHERE { ?x rdf:type ub:GraduateStudent.
        ?y rdf:type ub:University.
        ?z rdf:type ub:Department.
        ?x ub:memberOf ?z.
        ?z ub:subOrganizationOf ?y.
        ?x ub:undergraduateDegreeFrom ?y.
}
```

Listing A.2: LUBM-Anfrage 2.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:Publication.
        ?x ub:publicationAuthor
        <http://www.Department0.University0.edu/AssistantProfessor0>.
}
```

Listing A.3: LUBM-Anfrage 3.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x, ?y1, ?y2, ?y3
WHERE { ?x rdf:type ub:Professor.
        ?x ub:worksFor <http://www.Department0.University0.edu>.
        ?x ub:name ?y1. ?x ub:emailAddress ?y2. ?x ub:telephone ?y3.
}
```

Listing A.4: LUBM-Anfrage 4.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:Person.
        ?x ub:memberOf <http://www.Department0.University0.edu>.
}
```

Listing A.5: LUBM-Anfrage 5.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:Student. }
```

Listing A.6: LUBM-Anfrage 6.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT *?x, ?y
WHERE { ?x rdf:type ub:Student. ?y rdf:type ub:Course.
        <http://www.Department0.University0.edu/AssociateProfessor0>
                ub:teacherOf ?y.
        ?x ub:takesCourse ?y.
}
```

Listing A.7: LUBM-Anfrage 7.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x, ?y, ?z
WHERE { ?x rdf:type ub:Student. ?y rdf:type ub:Department.
        ?x ub:memberOf ?y.
        ?y ub:subOrganizationOf <http://www.University0.edu>.
        ?x ub:emailAddress ?z.
}
```

Listing A.8: LUBM-Anfrage 8.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x, ?y, ?z
WHERE { ?x rdf:type ub:Student. ?y rdf:type ub:Faculty.
        ?z rdf:type ub:Course. ?x ub:advisor ?y. ?x ub:takesCourse ?z
           .
        ?y ub:teacherOf ?z.
}
```

Listing A.9: LUBM-Anfrage 9.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:Student.
        ?x ub:takesCourse
                <http://www.Department0.University0.edu/
                   GraduateCourse0>.
}
```

Listing A.10: LUBM-Anfrage 10.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:ResearchGroup.
        ?x ub:subOrganizationOf <http://www.University0.edu>.
}
```

Listing A.11: LUBM-Anfrage 11.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x, ?y
WHERE { ?x rdf:type ub:Chair. ?y rdf:type ub:Department.
        ?x ub:worksFor ?y.
        ?y ub:subOrganizationOf <http://www.University0.edu>.
}
```

Listing A.12: LUBM-Anfrage 12.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:Person.
        <http://www.University0.edu> ub:hasAlumnus ?x.
}
```

Listing A.13: LUBM-Anfrage 13.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/%7Ezhp2/2004/0401/univ-bench.owl#>
SELECT ?x
WHERE { ?x rdf:type ub:UndergraduateStudent. }
```

Listing A.14: LUBM-Anfrage 14.

A.2. Ergänzende Evaluierungsergebnisse

Dieser Abschnitt enthält die vollständigen Ergebnisse der Evaluierung des Reasonings über verteilte Kontextinformationen auf Basis des *LeHigh University Benchmarks* (siehe dazu die Bemerkungen in Abschnitt 5.5.3.1 auf Seite 121).

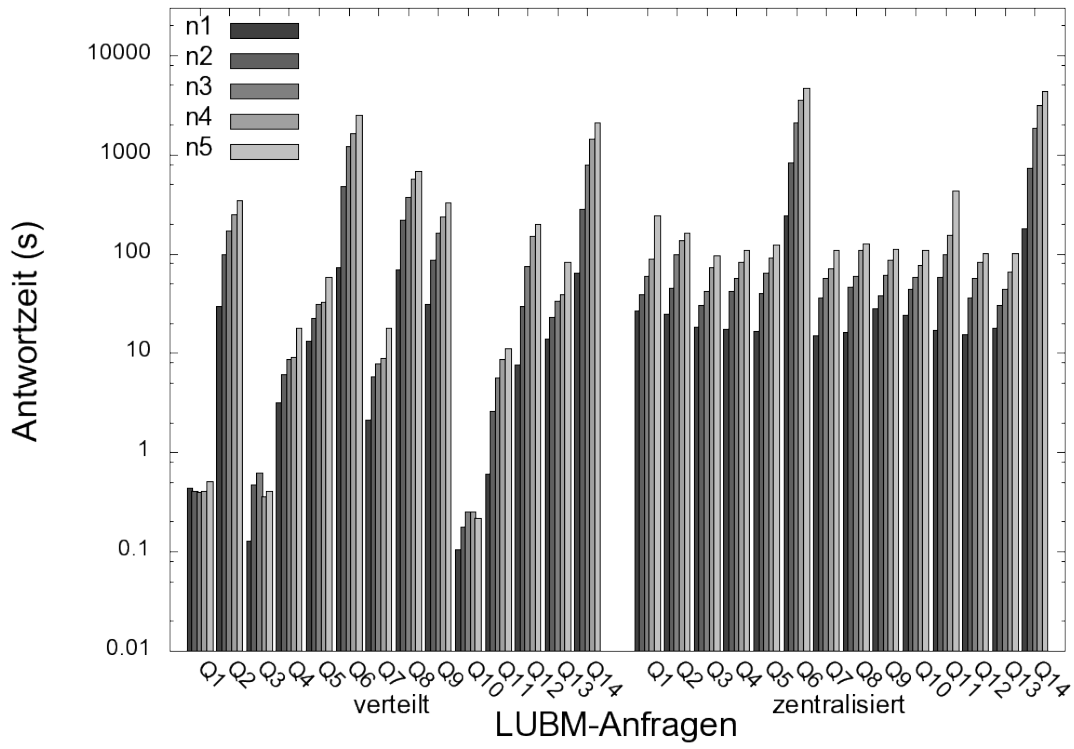

Abbildung A.1.: Gegenüberstellung der Antwortzeiten für alle LUBM-Test-Anfragen bei zwei Teil-Wissensbasen.

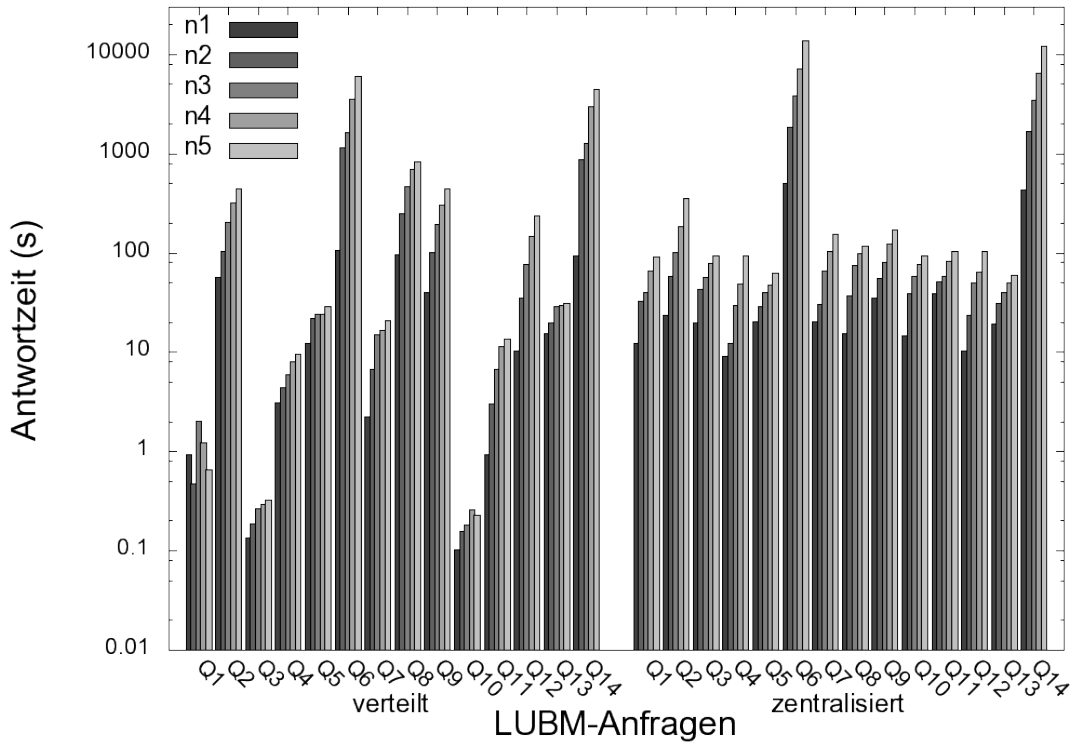

Abbildung A.2.: Gegenüberstellung der Antwortzeiten für alle LUBM-Test-Anfragen bei drei Teil-Wissensbasen.

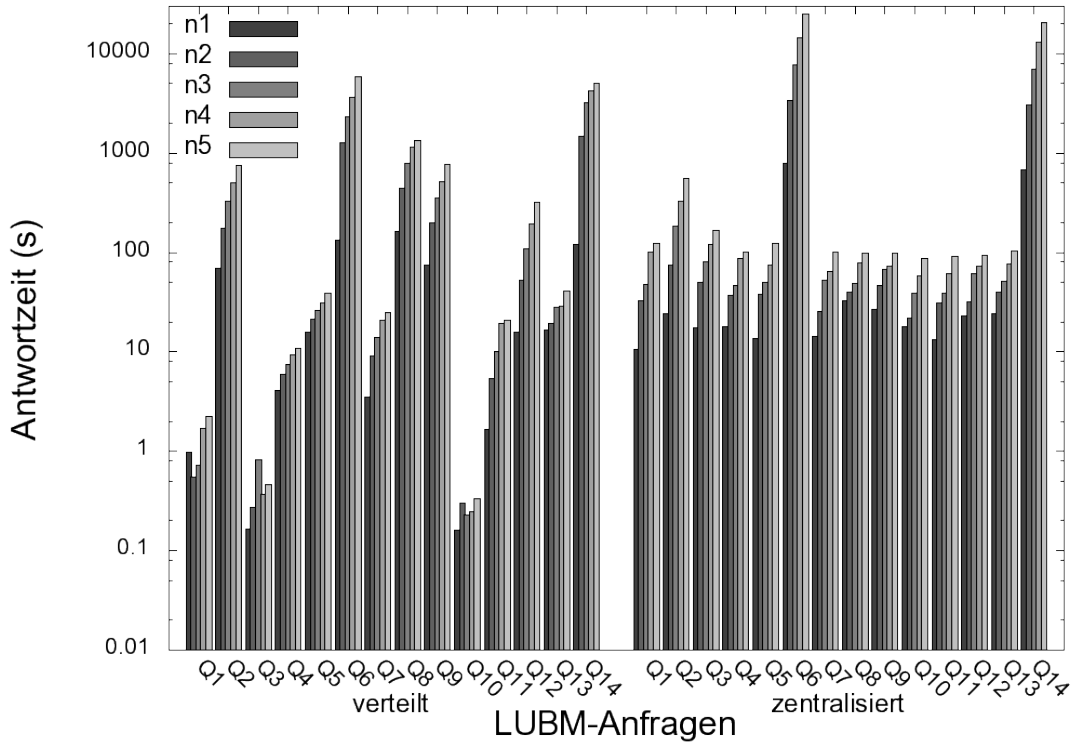

Abbildung A.3.: Gegenüberstellung der Antwortzeiten für alle LUBM-Test-Anfragen bei vier Teil-Wissensbasen.

Literaturverzeichnis

[ABC+05] Ahmed, R. ; Boutaba, R. ; Cuervo, F. ; Iraqi, Y. ; Li, T. ; Limam, N. ; Xiao, J. ; Ziembicki, J.: Service Naming in Large-Scale and Multi-Domain Networks. In: *IEEE Communications Surveys* 7 (2005), Nr. 3, S. 2–18

[ABE07] Alkhateeb, F. ; Baget, J.-F. ; Euzenat, J.: RDF with Regualar Expressions / Institut National de Recherche en Informatique et en Automatique. 2007. – Forschungsbericht

[ALM05] Arampatzis, T. ; Lygeros, J. ; Manesis, S.: A Survey of Applications of Wireless Sensors and Wireless Sensor Networks. In: *Proceedings of the 13th Mediterranean Conference on Control and Automation*. Limassol, Cyprus, Juni 2005

[AMS07] Anyanwu, K. ; Maduko, A. ; Sheth, A.: SPARQ2L: Towards Support for Subgraph Extraction Queries in RDF Databases. In: *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA : ACM, 2007, S. 797–806

[ASSC02] Akyildiz, I. F. ; Su, W. ; Sankarasubramaniam, Y. ; Cayirci, E.: Wireless Sensor Networks: A Survey. In: *Computer Networks* 38 (2002), März, Nr. 4, S. 393–422

[Bao07] Bao, J.: *Representing and Reasoning with Modular Ontologies*, Iowa State University, Diss., 2007

[BBK02] Balazinska, M. ; Balakrishnan, H. ; Karger, D.: INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In: *Proceedings of the International Conference on Pervasive Computing*. Zürich, Switzerland, August 2002

[BBL08] Baader, F. ; Brandt, S. ; Lutz, C.: Pushing the EL Envelope Further. In: *Proceedings of the 4th International Workshop on OWL: Experiences and Directions (OWLED)*, 2008

[BCH06] Bao, J. ; Caragea, D. ; Honavar, V.: On the Semantics of Linking and Importing in Modular Ontologies. In: *Proceedings of International Semantic Web Conference*, 2006

[BCM+03] Baader, F. ; Calvanese, D. ; McGuinness, D. ; Nardi, D. ; Patel-Schneider, P.: *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press New York, NY, USA, 2003

[BD04] Brügge, B. ; Dutoit, A.: *Object-Oriented Software Engineering: Conquering Complex and Changing Systems*. Prentice Hall International, 2004

[Ber06] Berndl, D.: *Ein Konzept zur entitätsbezogenen Lokalisierung von Kontexterbringerdiensten*, Ludwig-Maximilians-Universität München, Institut für Informatik, Lehrstuhl für Mobile und Verteilte Systeme, Diplomarbeit, 2006

[BFP06] Berger, M. ; Fuchs, F. ; Pirker, M.: Agent Technologies for Ambient Information Systems. In: *Proceedings of Workshop on Software-Agents in Information Systems and Industrial Applications (SAISIA)*. Karslruhe, Germany : Fraunhofer Gesellschaft, Februar 2006

[BFP07] Berger, M. ; Fuchs, F. ; Pirker, M.: Ambient Intelligence – From Personal Assistance to Intelligent Megacities. In: Augusto, J. C. (Hrsg.) ; Shapiro, D. (Hrsg.): *Advances in Ambient Intelligence* Bd. 164. Amsterdam, Netherlands : IOS Press, Frontiers in Artificial Intelligence and Applications, November 2007, S. 21–35

[BFP08] Berger, M. ; Fuchs, F. ; Pirker, M.: From Personal Assistance to Industrial Solutions. In: *it – Information Technology* Special Issue on Ambient Intelligence (2008), Januar

[BHH+04] Bechhofer, S. ; Harmelen, F. van ; Hendler, J. ; Horrocks, I. ; McGuinness, D. L. ; Patel-Schneider, P. F. ; Stein, L. A.: *OWL Web Ontology Language Reference*. W3C Recommendation, Februar 2004

[BHT05] Bechhofer, S. ; Horrocks, I. ; Turi, D.: The OWL Instance Store: System Description. In: *Proceedings of 20th International Conference on Automated Deduction (CADE-20)*. Tallinn, Estonia : Springer, Juli 2005

[BKI06] Beierle, C. ; Kern-Isberner, G.: *Methoden wissensbasierter Systeme*. Vieweg, 2006

[BL04] Brachman, R. ; Levesque, H.: *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2004

[BLA07] Barnes, M. ; Leather, H. ; Arvind, D. K.: Emergency Evacuation using Wireless Sensor Networks. In: *Proceedings of 32nd IEEE Conference on Local Computer Networks (LCN 2007)*, 2007

[BLHL01] Berners-Lee, T. ; Hendler, J. ; Lassila, O.: The Semantic Web. In: *Scientific American* (2001), Mai, Nr. 284, S. 34–43

[Blo04] Blomqvist, E.: State of the Art: Patterns in Ontology Engineering / Jönköping University. 2004 (04:8). – Forschungsbericht

[BMR+96] Buschmann, F. ; Meunier, R. ; Rohnert, H. ; Sommerlad, P. ; Stal, M.: *Pattern-oriented Software Architecture*. Wiley, 1996

[BPW+07] Botts, M. ; Percivall, G. ; Williams, R. ; Reed, K. S. C. ; Davidson, J.: *OGC Sensor Web Enablement: Overview and High Level Architecture (OGC 07-165)*. OGC White Paper, Dezember 2007

[Bri07] Brinkhoff, T.: Räumliche Netzwerk- und Topologiedatenbanken. In: *Datenbank-Spektrum* 21 (2007), Nr. 20, S. 15–21

[BS03] Borgida, A. ; Serafini, L.: Distributed Description Logics: Assimilating Information from Peer Sources. In: *Journal on Data Semantics I* (2003)

[BS05] Blomqist, E. ; Sandkuhl, K.: Patterns in Ontology Engineering: Classification of Ontology Patterns. In: *Proceedings of 7th International Conference on Enterprise Information Systems*. Miami, USA, 2005

[BSI04] Bundesamt für Sicherheit in der Informationstechnik (BSI): *Kritische Infrastrukturen in Deutschland*. 2004

[Bük08] Büker, T.: *Railways through Europe*. `http://www.bueker.net/trainspotting/maps_germany.php`. Version: Mai 2008

[Bun08] Bundesnetzagentur: *Bericht gemäß §63 Abs. 4 a EnWG zur Auswertung der Netzzustands- und Netzausbauberichte der deutschen Elektrizitätsübertragungsnetzbetreiber*. Januar 2008

[CFJ04] Chen, H. ; Finin, T. ; Joshi, A.: An Ontology for Context-Aware Pervasive Computing Environments. In: *Knowledge Engineering Review, Special Issue on Ontologies for Distributed Systems* 18 (2004), Mai, Nr. 3, S. 197–207

[CFT08] Clark, K. G. ; Feigenbaum, L. ; Torres, E.: *SPARQL Protocol for RDF*. W3C Recommendation, Januar 2008

[Cha98] Chaudhuri, S.: An Overview of Query Optimization in Relational Systems. In: *Proceedings of the 17th ACM Symposium on Principles of Database Systems*. New York, NY, USA : ACM, 1998, S. 34–43

[Chu36] Church, A.: An Unsolvable Problem of Elementary Number Theory. In: *American Journal of Mathematics* 58 (1936), S. 345–363

[Cyg05] Cyganiak, R.: A Relational Algebra for SPARQL / Digital Media Systems Laboratory, HP Laboratories Bristol. 2005. – HPL-2005-170

[DB06] Deutsche Bahn AG: *Geschäftsbericht 2005*. März 2006

[DB07] Deutsche Bahn AG: *Infrastrukturzustands und -entwicklungsbericht*. Juni 2007

[Del02] Delin, K.: The Sensor Web: A Macro-Instrument for Coordinated Sensing. In: *Sensors* 2 (2002), Nr. 1, S. 270–285

[Del04] Delin, K.: The Sensor Web: A Distributed, Wireless Monitoring System. In: *Sensors* (2004), April

[Del06] Delin, K. A.: The Sensor Web: Distributed Sensing for Collective Action. In: *Sensors Online* (2006), Nr. 18

[Dey01] Dey, A. K.: Understanding and Using Context. In: *Personal and Ubiquitous Computing* 5 (2001), Nr. 1, S. 4–7

[DFK+07] Dolby, J. ; Fokoue, A. ; Kalyanpur, A. ; Kershenbaum, L. A. a. A. andMa ; Schonberg, E. ; Srinivas, K.: Scalable Semantic Retrieval through Summarization and Refinement. In: *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*. Vancouver, British Columbia : AAAI Press, 2007, S. 209–304

[DG04] DG-FER Consortium: *Roadmapping of the Paths for the Introduction of Distributed Generation in Europe*. Project Report Distributed Generation - Future Energy Resources, März 2004

[DHM96] Dailey, D. ; Haselkorn, M. ; Meyers, D.: A Structured Approach to Developing Real-Time, Distributed Network Applications for ITS. In: *ITS Journal*, 1996

[Dij59] Dijkstra, E. W.: A Note on Two Problems in Connexion with Graphs. In: *Numerische Mathematik* Bd. 1. Mathematisch Centrum, Amsterdam, The Netherlands, 1959, S. 269–271

[DJ01] Delin, K. A. ; Jackson, S. P.: The Sensor Web: A New Instrument Concept. In: *Symposium on Integrated Optics*, Society of Photo-Optical Instrumentation Engineers, 2001

[DMPG02] Dailey, D. ; Meyers, D. ; Pond, I. ; Guiberson, K.: Traffic Data Acquisition and Distribution (TDAD) / Washington State Transportation Center (TRAC). 2002. – Forschungsbericht

[EIU07] Economist Intelligence Unit (Hrsg.): *Megacities und ihre Herausforderungen*. Mit Unterstützung der Siemens AG, 2007

[FB06] Fuchs, F. ; Berger, M.: Towards Scalable Retrieval of Distributed and Dynamic Ontology Instances. In: *Proceedings of International Semantic Web Conference (ISWC), Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS)*. Athens, GA, USA : Springer Lecture Notes in Computer Science, November 2006

[FB07] Fuchs, F. ; Berger, M.: Using Off-the-Shelf Reasoners for Reasoning over Distributed ABoxes. In: *Proceedings of International Workshop on Description Logics (DL2007)* Bd. 250, CEUR Workshop Proceedings, Juni 2007

[FB08] Fuchs, F. ; Berger, M.: A Decentralized and Ontology-Based Approach to Infrastructure Management. In: *Proceedings of Multi-Conference Information Systems – Decentralization as a Design Principle*, Gesellschaft für Informatik, Februar 2008

[FBLP09] Fuchs, F. ; Berger, M. ; Linnhoff-Popien, C.: Smart Monitoring for Physical Infrastructures. In: Nakashima, H. (Hrsg.) ; Augusto, J. C. (Hrsg.) ; Aghajan, H. (Hrsg.): *Handbook on Ambient Intelligence and Smart Environments*. Springer, 2009. – to appear

[FBP08] Fuchs, F. ; Berger, M. ; Pirker, M.: Semantic Processing of Monitoring Data in Industrial Applications. In: Cardoso, J. (Hrsg.) ; Lytras, M. D. (Hrsg.): *Semantic Web Engineering in the Knowledge Society*. Hershey, PA, USA : IGI Global, Information Science Reference, Oktober 2008

[FBT07] Fuchs, F. ; Berndl, D. ; Treu, G.: Towards Entity-centric Wide-area Context Discovery. In: *Proceedings of MDM2007, Workshop on Mobile Services-oriented Architectures and Ontologies (MoSO07)*. Mannheim, Germany : IEEE Computer Society, Mai 2007

[FHH04] Fikes, R. ; Hayes, P. ; Horrocks, I.: OWL-QL – A Language for Deductive Query Answering on the Semantic Web. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 2 (2004), Nr. 1, S. 19–29

[FHKB05] Fuchs, F. ; Hochstatter, I. ; Krause, M. ; Berger, M.: A Meta-Model Approach to Context Information. In: *Proceedings of International Conference on Pervasive Computing and Communications (PerCom), Workshop on Context Modelling and Reasoning (CoMoRea)*. Hawaii, USA : IEEE Computer Society, März 2005

[FHP+06] Fuchs, F. ; Henrici, S. ; Pirker, M. ; Berger, M. ; Langer, G. ; Seitz, C.: Towards Semantics-based Monitoring of Large-scale Industrial Systems. In: *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA)*. Sydney, Australia : IEEE Computer Society, November 2006

[FKM+06] Fokoue, A. ; Kershenbaum, A. ; Ma, L. ; Schonberg, E. ; Srinivas, K.: The Summary ABox: Cutting Ontologies Down to Size. In: *Proceedings of International Semantic Web Conference (ISWC)*, 2006

[FLP+06] Fuchs, F. ; Lewis, R. ; Pirker, M. ; Roberts, C. ; Berger, M. ; Langer, G.: Appplying Semantic Technologies to Railway Decision Support. In: *Proceedings of Semantics 2006*. Vienna, Austria : Austrian Computer Society, November 2006

[For06] Foresight Programme, UK Office of Science and Technology: *Intelligent Infrastructure Futures – Project Overview*. Januar 2006

[FPN+05] Floreen, P. ; Przybilski, M. ; Nurmi, P. ; Koolwaaij, J. ; Tarlano, A. ; Wagner, M. ; Luther, M. ; Bataille, F. ; Boussard, M. ; Mrohs, B. u. a.: Towards a Context Management Framework for MobiLife. In: *Proceedings of the IST Summit*, 2005

[FSWF05] Feng, J. ; Smith, J. ; Wu, Q. ; Fitch, J.: Condition Assessment of Power System Apparatuses Using Ontology Systems. In: *Proceedings of Transmission and Distribution Conference*. Dalian, China : IEEE/PES, 2005

[FWF04] Feng, J. ; Wu, Q. ; Fitch, J.: An Ontology for Knowledge Representation in Power Systems. In: *Proceedings of Control 2004 Conference*. University of Bath, UK, 2004

[GHJV95] Gamma, E. ; Helm, R. ; Johnson, R. ; Vlissides, J.: *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1995

[GHLS08] Glimm, B. ; Horrocks, I. ; Lutz, C. ; Sattler, U.: Conjunctive Query Answering for the Description Logic $\mathcal{SHIQ}$. In: *Journal of Artificial Intelligence Research* 31 (2008), S. 157–204

[GHT06] Gardiner, T. ; Horrocks, I. ; Tsarkov, D.: Automated Benchmarking of Description Logic Reasoners. In: *Proceedings of International Workshop on Description Logics (DL'06)*, 2006

[GMPQ+97] Garcia-Molina, H. ; Papakonstantinou, Y. ; Quass, D. ; Rajaraman, A. ; Sagiv, Y. ; Ullman, J. ; Vassalos, V. ; Widom, J.: The TSIMMIS Approach to Mediation: Data Models and Languages. In: *Journal of Intelligent Information Systems* 8 (1997), Nr. 2, S. 117–132

[GMUW01] Garcia-Molina, H. ; Ullman, J. D. ; Widom, J. D.: *Database Systems: The Complete Book*. Prentice Hall, 2001

[GPH05] Guo, Y. ; Pan, Z. ; Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3 (2005), Oktober, Nr. 2-3, S. 158–182

[Gra93] Graefe, G.: Query Evaluation Techniques for Large Databases. In: *ACM Computing Surveys* 25 (1993), Juni, Nr. 2, S. 73—169

[Gru93] Gruber, T. R.: A Translation Approach to Portable Ontologies. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220

[Gru95] Gruber, T.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: *International Journal of Human-Computer Studies* 43 (1995), Nr. 5/6, S. 907–928

[Gru08] Gruber, T. R.: Ontology. In: Liu, L. (Hrsg.) ; Özsu, M. T. (Hrsg.): *Encyclopedia of Database Systems*. Springer-Verlag, 2008

[GTH06] Gardiner, T. ; Tsarkov, D. ; Horrocks, I.: Framework for an Automated Comparison of Description Logic Reasoners. In: *Proceedings of International Semantic Web Conference*. Athens, GA : Springer, 2006, S. 654–667

[GTPZ05] Gu, T. ; Tan, E. ; Pung, H. K. ; Zhang, D.: ContextPeers: Scalable Peer-to-Peer Search for Context Information. In: *Proceedings of the Workshop on Innovations in Web Infrastructure (WWW)*. Japan, 2005

[GW02] Guarino, N. ; Welty, C.: Evaluating Ontological Decisions with OntoClean. In: *Communications of the ACM* 45 (2002), Nr. 2, S. 61–65

[GW04] GridWise Alliance: *GridWise Alliance – A Collaborative Venture of the US Department of Energy and the GridWise Alliance*. http://www.gridwise.org/, 2004

[GWPZ04] Gu, T. ; Wang, H. ; Pung, H. K. ; Zhang, D. Q.: An Ontology-based Context Model in Intelligent Environments. In: *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*. San Diega, CA, 2004

[Hal01] Halevy, A.: Answering Queries Using Views: A Survey. In: *International Journal on Very Large Data Bases* 10 (2001), Nr. 4, S. 270–294

[HDP+94] Holte, R. ; Drummond, C. ; Perez, M. ; Zimmer, R. ; MacDonald, A. J.: Searching With Abstractions: A Unifying Framework and New High-Performance Algorithm. In: *Canadian Conference on Artificial Intelligence (CAI)*, 1994, S. 263–270

[HJR95] Huang, Y.-W. ; Jing, N. ; Rundensteiner, E. A.: Hierarchical Path Views: A Model Based on Fragmentation and Transportation Road Types. In: *ACM-GIS*, 1995, 93–100

[HLMS08] Hepp, M. (Hrsg.) ; Leenheer, P. D. (Hrsg.) ; Moor, A. de (Hrsg.) ; Sure, Y. (Hrsg.): *Ontology Management – Semantic Web, Semantic Web Services, and Business Applications*. Springer, 2008

[HLTB04] Horrocks, I. ; Li, L. ; Turi, D. ; Bechhofer, S.: The Instance Store: DL Reasoning with Large Numbers of Individuals. In: *Proceedings of the 2004 Description Logic Workshop (DL)*, 2004

[HM03] Haarslev, V. ; Möller, R.: Racer: A Core Inference Engine for the Semantic Web. In: *Proceedings of the 2nd International Semantic Web Conference (ISWC), Workshop on Evaluation of Ontology-based Tools*. Sanibel Island, SA, 2003, S. 27–36

[HM05] Haase, P. ; Motik, B.: A Mapping System for the Integration of OWL-DL Ontologies. In: *Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems*, ACM, New York, NY, 2005, S. 9–16

[HMS04] Hustadt, U. ; Motik, B. ; Sattler, U.: Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In: *Proceedings of the 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 2004, S. 152–162

[HMS05] Hustadt, U. ; Motik, B. ; Sattler, U.: Data Complexity of Reasoning in Very Expressive Description Logics. In: *Proceedings of 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005

[HMW04] Haarslev, V. ; Möller, R. ; Wessel, M.: Querying the Semantic Web with Racer + nRQL. In: *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL)*. Ulm, Germany, September 2004

[HMZM96] Holte, R. C. ; Mkadmi, T. ; Zimmer, R. M. ; MacDonald, A. J.: Speeding up Problem Solving by Abstraction: A Graph Oriented Approach. In: *Artificial Intelligence* 85 (1996), Nr. 1-2, S. 321–361

[HP06] Hobbs, J. ; Pan, F.: *Time Ontology in OWL*. W3C Working Draft, http://www.w3.org/TR/owl-time/, September 2006

[HST00a] Horrocks, I. ; Sattler, U. ; Tobies, S.: Practical Reasoning for Very Expressive Description Logics. In: *Logic Journal of the IGPL* 8 (2000), Nr. 3, S. 239–264

[HST00b] Horrocks, I. ; Sattler, U. ; Tobies, S.: Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In: *Proceedings of 17th International Conference on Automated Deduction (CADE)*. London, UK : Springer-Verlag, 2000, S. 482–496

[Hub06] Hubauer, T. M.: *Situationserkennung im Kontext unsicherer Informationen*, Ludwig-Maximilans-Universität München, Diplomarbeit, November 2006

[HW07] Haase, P. ; Wang, Y.: A Decentralized Infrastructure for Query Answering over Distributed Ontologies. In: *Proceedings of the 22nd ACM Symposium on Applied Computing (SAC)*. Seoul, Korea : ACM Press, März 2007

[HWH07] Halaschek-Wiener, C. ; Hendler, J.: Toward Expressive Syndication on the Web. In: *Proceedings of the 16th International World Wide Web Conference (WWW)*, 2007

[HWPS06a] Halaschek-Wiener, C. ; Parsia, B. ; Sirin, E.: Description Logic Reasoning with Syntactic Updates. In: *Proceedings of 5th International Conference on Ontologies, Databases, and Applications of Semantics (ODBase)*, 2006

[HWPS06b] Halashek-Wiener, C. ; Parsia, B. ; Sirin, E.: Towards Continuous Query Answering on the Semantic Web. In: *Proceedings of the 2nd International Workshop on Scalable Semantic Web Systems (ISWC2006)*, 2006

[IGR05] EU-Forschungsprojekt im Rahmenprogramm 6 der Europäischen Kommission: *InteGRail – Intelligent Integration of Railway Systems*. http://www.integrail.info/, 2005

[IGRP08] *InteGRail Web Portal*. `http://www.integrail.eu/`. Version: August 2008

[JHR98] Jing, N. ; Huang, Y.-W. ; Rundensteiner, E.: Hierarchical Encoded Path Views for Path Query Processing: An Optimal Model and Its Performance Evaluation. In: *IEEE Transactions on Knowledge and Data Engineering* Bd. 10, 1998, S. 409–432

[JKPT03] Jensen, C. ; Kolářvr, J. ; Pedersen, T. ; Timko, I.: Nearest Neighbor Queries in Road Networks. In: *Proceedings of the 11th ACM International Symposium on Advances inGeographic Information Systems* (2003), S. 1–8

[Jul07] Julliard, Y.: *Smart Substation Automation - Intelligent Lösungen zur Stationsautomatisierung*. Vortrag beim Energieforum Life Needs Power (Hannover Messe), 2007

[KG05] Katz, Y. ; Grau, B. C.: Representing Qualitative Spatial Information in OWL-DL. In: *Proceedings of Workshop on OWL: Experiences and Directions (OWL-ED)*, 2005

[KHS06] Kutz, O. ; Horrocks, I. ; Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: *Proceedings of the 10th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006)*. Lake District, UK : AAAI Press, Juni 2006, S. 68–78

[KJ07] Kochut, K. J. ; Janik, M.: Extended SPARQL for Semantic Association Discovery. In: *The Semantic Web: Research and Applications*, Springer Berlin / Heidelberg, 2007 (Lecture Notes in Computer Science), S. 145–159

[KLWZ04] Kutz, O. ; Lutz, C. ; Wolter, F. ; Zakharyaschev, M.: E-connections of Abstract Description Systems. In: *Artificial Intelligence* 156 (2004), Nr. 1, S. 1–73

[KOM05] Kiryakov, A. ; Ognyanov, D. ; Manov, D.: OWLIM – a Pragmatic Semantic Repository for OWL. In: *Proceedings of International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005)*. New York City, USA : Springer, 2005

[Kre03a] Kreiss, D.: Non-operational Data: The Untapped Value of Substation Automation. In: *Utility Automation & Engineering / T & D* 8 (2003), September, Nr. 5

[Kre03b] Kreiss, D.: Utilities Can Enhance Bottom Line by Leveraging Non-Operational Data. In: *Utility Automation & Engineering / T & D* 8 (2003), November, Nr. 6

[KRL05] Kezunovic, M. ; Ren, Z. ; Latisko, G.: Automated Monitoring and Analysis of Circuit Breaker Operation. In: *Transactions on Power Delivery* 20 (2005), Juli, Nr. 3, S. 1910–1918

[KS04] Kolahdouzan, M. ; Shahabi, C.: Voronoi-based K-Nearest Neighbor Search for Spatial Network Databases. In: *Proceedings of the 30th VLDB Conference*. Toronto, Canada, 2004

[KS07a] Kolas, D. ; Self, T.: Spatially-Augmented Knowledgebase. In: *Proceedings of 6th International Semantic Web Conference (ISWC07)* Bd. 4825. Busan, Korea : Springer, November 2007 (Lecture Notes in Computer Science)

[KS07b] Krummenacher, R. ; Strang, T.: Ontology-Based Context Modeling. In: *Proceedings of Workshop on Context-Aware Proactive Systems*, 2007

[Küp05] Küpper, A.: *Location-Based Services – Fundamentals and Operation*. John Wiley & Sons, Ltd, 2005

[Kur92] Kurbel, K.: *Entwicklung und Einsatz von Expertensystemen*. Springer-Verlag, 1992

[Lag07] Lagnebäck, R.: *Evaluation of Wayside Condition Monitoring Technologies for Condition-basedMaintenance of Railway Vehicles*, Luleå University of Technology, Diss., 2007

[Lan02] Lange, N. de: *Geoinformatik in Theorie und Praxis*. Springer, 2002

[LFP+06] Lewis, R. ; Fuchs, F. ; Pirker, M. ; Roberts, C. ; Langer, G.: Using Ontology to Integrate Railway Condition Monitoring Data. In: *Proceedings of International Conference on Railway Condition Monitoring*. Birmingham, UK : IET, November 2006

[Lie06] Liebig, T.: Reasoning with OWL – System Support and Insights / Ulm University. Ulm, Germany, September 2006 (TR-2006-04). – Forschungsbericht

[LMW+05] Luther, M. ; Mrohs, B. ; Wagner, M. ; Steglich, S. ; Kellerer, W.: Situational Reasoning – a Practical OWL Use Case. In: *Proceedings of International Symposium on Autonomous Decentralized Systems*, 2005, S. 461–468

[LRO+96] Levy, A. ; Rajaraman, A. ; Ordille, J. u. a.: Querying Heterogeneous Information Sources Using Source Descriptions. In: *Proceedings of VLDB* Bd. 96, 1996, S. 251–262

[LS05] Loske, R. (Hrsg.) ; Schaeffer, R. (Hrsg.): *Die Zukunft der Infrastrukturen – Intelligente Netzwerke für eine nachhaltige Entwicklung.* Marburg : Metropolis-Verlag, 2005 (Ökologie und Wirtschaftsforschung 57)

[Luk07] Lukasiewicz, T.: Probabilistic Description Logics for the Semantic Web / Institut für Informationssysteme, Technische Universität Wien. 2007 (1843-06-05). – Forschungsbericht

[Maz05] Mazza, P.: *Powering Up the Smart Grid: A Northwest Initiative for Job Creation, Energy Security and Clean, Affordable Energy.* Poised for Profit in Clean Energy Report, Juli 2005

[MFSW08] Mieth, S. ; Fuchs, F. ; Stoffel, E.-P. ; Weiss, D.: Reasoning on Geo-Referenced Sensor Data in Infrastructure Networks. In: *Proceedings of the 11th International Conference on Geographic Information Science (AGILE), Workshop on Semantic Web Meets Geospatial Applications*, 2008

[MHW06] Möller, R. ; Haarslev, V. ; Wessel, M.: On the Scalability of Description Logic Instance Retrieval. In: *29. Deutsche Jahrestagung für Künstliche Intelligenz*, 2006 (Lecture Notes in Artificial Intelligence)

[Mie07] Mieth, S.: *Reasoning on Location-Related Sensor Data in Infrastructure Networks*, Technische Universität Dresden, Diplomarbeit, 2007

[Min75] Minsky, M.: A Framework for Representing Knowledge. The Psychology of Computer Vision. In: *Mc-Graw & Hill, New-York* (1975)

[MKL+03] Milojicic, D. S. ; Kalogeraki, V. ; Lukose, R. ; Nagaraja, K. ; Pruyne, J. ; Richard, B. ; Rollins, S. ; Xu, Z.: Peer-to-Peer Computing / HP Laboratories Palo Alto. 2003 (HPL-2002-57 (R.1)). – Forschungsbericht

[MMS+03] Maedche, A. ; Motik, B. ; Stojanovic, L. ; Studer, R. ; Volz, R.: An Infrastructure for Searching, Reusing and Evolving Distributed Ontologies. In: *Proceedings of WWW2003.* Budapest, Hungary, 2003

[MRS04] Maly, T. ; Rumpler, M. ; Schweinzer, K.: Joining Sensor Systems for Railway Vehicle Inspection. In: *Proceedings of IEEE Sensors*, IEEE, Oktober 2004, S. 12–15

[MRSS05] Maly, T. ; Rumpler, M. ; Schweinzer, H. ; Schoebel, A.: New Development of an Overall Train Inspection System for Increased Operational Safety. In: *Proceedings of Intelligent Transportation Systems*, IEEE, 2005

[MS05] Maly, T. ; Schweinzer, H.: Measurement data treatment in multisensor applications for railway vehicle inspection. In: *Journal of Physics: Conference Series* 13 (2005), Januar, S. 381–384

[MS06a] Moodley, D. ; Simonis, I.: New Architecture for the Sensor Web: the SWAP Framework. In: *Proceedings of 5th International Semantic Web Conference (ISWC)*. Athens, Georgia, November 2006

[MS06b] Motik, B. ; Sattler, U.: A Comparison of Techniques for Querying Large Description Logic ABoxes. In: *Proceedings of the 13th International Conference on Logic Programming Artificial Intelligence and Reasoning (LPAR)* Bd. 4246. Phnom Penh, Cambodia : Springer, 2006 (Lecture Notes in Computer Science)

[MS06c] Motik, B. ; Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In: *Proceedings of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)* Bd. 4246. Phnom Penh, Cambodia : Springer, November 2006 (LNCS), S. 227–241

[MSH07] Motik, B. ; Shearer, R. ; Horrocks, I.: Optimized Reasoning in Description Logics using Hypertableaux. In: *Proceedings of the 16th Int. Conf. on Automated Deduction (CADE-16), July*, Springer, 2007, S. 17–20

[MYQ+06] Ma, L. ; Yang, Y. ; Qiu, Z. ; Xie, G. ; Pan, Y.: Towards a Complete OWL Ontology Benchmark. In: *Proceedings of the Third European Semantic Web Conference (ESWC 2006)*, 2006

[Nai84] Naisbitt, J.: *Megatrends – 10 Perspektiven, die unser Leben verändern werden*. Hestia Verlag GmbH, 1984

[NLZ07] Nath, S. ; Liu, J. ; Zhao, F.: SensorMap for Wide-Area Sensor Webs. In: *IEEE Computer Magazine* 40 (2007), Juli, Nr. 7, S. 90–93

[NR07] Network Rail: *The 2009 Network Statement*. Oktober 2007

[Oll06] Ollier, B. D.: Intelligent Infrastructure the Business Challenge. In: *Proceedings of IET International Conference on Railway Condition Monitoring*. Birmingham, UK : IET, November 2006, S. 1–6

[OSHA07] U.S. Department of Labor – Occupational Safety & Health Administration: *Distribution Systems*. `http://www.osha.gov/SLTC/etools/electric_power/illustrated_glossary`. Version: Dezember 2007

[PAG06] Perez, J. ; Arenas, M. ; Gutierrez, C.: Semantics and Complexity of SPARQL. In: *Proceedings of International Semantic Web Conference*, 2006

[Pei31] Peirce, C.: *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931

[PG05] Pothipruk, P. ; Governatori, G.: A Formal Ontology Reasoning with Individual Optimization: A Realization of the Semantic Web. In: *Proceedings of Conference on Web Information Systems Engineering*, 2005, S. 119–132

[PS08] Prud'hommeaux, E. ; Seaborne, A.: *SPARQL Query Language for RDF*. W3C Recommendation, Januar 2008

[Pup91] Puppe, F.: *Einführung in Expertensysteme*. Springer, 1991

[PZMT03] Papadias, D. ; Zhang, J. ; Mamoulis, N. ; Tao, Y.: Query Processing in Spatial Network Databases. In: *Proceedings of the 29th International Conference on Very Large Data Bases* (2003), S. 802–813

[Qui67] Quillian, M. R.: Word Concepts: a Theory and Simulation of Some Basic Semantic Capabilities. In: *Behavioral Science* 12 (1967), S. 410–430

[RAC+07] Rendschmidt, D. ; Arms, H. ; Cord, M. ; Gottschalk, M. ; Maxelon, M.: Die Zukunft der deutschen Stromnetze – veränderte Eigentümerstrukturen und intelligente Technologien. In: *Energiewirtschaftliche Tagesfragen* 57 (2007), Nr. 11, S. 74–77

[RB01] Ruback, L. G. ; Balke, K.: Integrating Train Information for Advanced Transportation Management / Texas Transportation Institute. 2001. – Forschungsbericht

[Rec03] Rector, A.: Modularisation of Domain Ontologies Implemented in Description Logics and Related Formalisms including OWL. In: *Proceedings of International Conference on Knowledge Capture*, ACM, 2003, S. 121–128

[Rec05] Rector, A.: *Representing Specified Values in OWL: "value partitions" and "value sets"*. W3C Working Group Note, http://www.w3.org/TR/swbp-specified-values/, Mai 2005

[Rei99] Reich, J. R.: Ontological Design Patterns for the Integration of Molecular Biological Information. In: *Proceedings of the German Conference on Bioinformatics (GCB)*. Hanover, Germany, 1999, S. 156–166

[RG03] Ramakrishnan, R. ; Gehrke, J.: *Database Management Systems*. McGraw-Hill Professional, 2003

[RKT05] Russomanno, D. J. ; Kothari, C. R. ; Thomas, O. A.: Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models. In: *Proceedings of International Conference on Artificial Intelligence*. Las Vegas, NV, 2005

[RM06] Risson, J. ; Moors, T.: Survey of Research Towards Robust Peer-to-Peer Networks: Search Methods. In: *Computer Networks* 50 (2006), Dezember, Nr. 17, S. 3485–3521

[RN03] Russell, S. J. ; Norvig, P.: *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003

[RW05] Rector, A. ; Welty, C.: *Simple part-whole relations in OWL Ontologies*. W3C's Editor Draft, http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/, August 2005

[Säm06a] Sämann, A.: *Ortsbezogene Lokalisierung von Kontexterbringerdiensten*, Ludwig-Maximilians-Universität München, Institut für Informatik, Lehrstuhl für Mobile und Verteilte Systeme, Diplomarbeit, 2006

[Sam06b] Samet, H.: *Foundations of Multidimensional and Metric Data Structures*. Amsterdam : Elsevier, 2006

[SAS05] Sankaranarayanan, J. ; Alborzi, H. ; Samet, H.: Efficient Query Processing on Spatial Networks. In: *Proceedings of 13th Annual ACM International Workshop on Geographic Information Systems (GIS05)*, 2005

[SBF98] Studer, R. ; Benjamins, V. R. ; Fensel, D.: Knowledge Engineering: Principles and Methods. In: *Data & Knowledge Engineering* 25 (1998), März, Nr. 1–2, S. 161–197

[Sch05] Schossig, W.: 10 Jahre elektrische Wiedervereinigung Deutschland. In: *ew* 104 (2005), Nr. 21-22, S. 80–83

[SEM01] Staab, S. ; Erdmann, M. ; Maedche, A.: Engineering Ontologies using Semantic Patterns. In: *Proceedings of IJCAI-01 Workshop on E-Business and the Intelligent Web*, 2001

[SHG05] Stuckenschmidt, H. ; Harmelen, F. van ; Giunchiglia, F.: Query Processing in Ontology-Based Peer-to-Peer Systems. In: *Ontologies for Agents: Theory and Experiences*. Birkhauser, 2005

[SHHT98] Steen, M. van ; Hauck, F. J. ; Homburg, P. ; Tanenbaum, A. S.: Locating Objects in Wide-Area Systems. In: *IEEE Communications Magazine* 36 (1998), Januar, Nr. 1, S. 104–109

[SLP04] Strang, T. ; Linnhoff-Popien, C.: A Context Modeling Survey. In: *Workshop on Advanced Context Modelling, Reasoning and Management (UbiComp)*. Nottingham, England, 2004

[SLW+07] Schröder, A. ; Laresgoiti, I. ; Werlen, K. ; Brelie, B. S. d. ; Schnettler, A.: Intelligent Self-Describing Power Grids. In: *Conference on Electricity Distribution*. Vienna, Austria : CIRED, 2007

[SM02] Shoemaker, T. M. ; Mack, J. E.: *The Lineman's and Cableman's Handbook*. Bd. 10. McGraw-Hill Professional, 2002

[SM06] Schweinzer, H. ; Maly, T.: Automatische Zugüberwachung durch Checkpoint-Systeme. In: *e & i Elektrotechnik und Informationstechnik* 123 (2006), Mai, Nr. 5, S. 178–182

[SP07] Sirin, E. ; Parsia, B.: SPARQL-DL: SPARQL Query for OWL-DL. In: *Proceedings of 3rd International Workshop on OWL: Experiences and Directions (OWLED)*. Innsbruck, Austria, Juni 2007

[SPG+07] Sirin, E. ; Parsia, B. ; Grau, B. ; Kalyanpur, A. ; Katz, Y.: Pellet: A Practical OWL-DL Reasoner. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 5 (2007), Nr. 2, S. 51–53

[SPK06] Schöbel, A. ; Pisek, M. ; Karner, J.: Hot Box Detection Systems as a Part of Automated Train Observation in Austria. In: *Proceedings of 14th International Symposium EURNEX-ZEL "Towards the competitive rails systems in Europe"*. Zilina, May 2006, S. 157–161

[SRAE06] European Commission, Directorate-General for Research, SustainableEnergy Systems: *European SmartGrids Technology Platform: Vision and Strategy for Europe's Electricity Networks of the Future*. EUR 22040, 2006

[SRAE07] European Commission, Directorate-General for Research, SustainableEnergy Systems: *Strategic Research Agenda for Europe's Electricity Networks of the Future*. EUR 22580, 2007

[SRRA02] The European Rail Research Advisory Council (ERRAC): *Strategic Rail Research Agenda*. 2002

[SRRA07] The European Rail Research Advisory Council (ERRAC): *Strategic Rail Research Agenda 2020*. Mai 2007

[SS07] Stocker, M. ; Seaborne, A.: ARQo: The Architecture for an ARQ Static Query Optimizer / Digital Media Systems Laboratory, HP Laboratories Bristol. 2007 (HPL-2007-92). – Forschungsbericht

[SSI+07] Shaw, K. ; Sample, J. ; Ioup, E. ; Abdelguerfi, M. ; Mansion, O.: Graph Processing For Spatial Network Queries. In: *Proceedings of International Conference on Information & Knowledge Engineering (IKE)*, 2007

[SSS91] Schmidt-Schauss, M. ; Smolka, G.: Attributive Concept Descriptions with Complements. In: *Artificial Intelligence* 48 (1991), S. 1–26

[ST98] Steets, P. ; Tse, Y.: Conrail's Integrated Automated Wayside Inspection. In: *Railroad Conference, 1998. Proceedings of the 1998 ASME/IEEE Joint* (1998), S. 113–125

[ST05] Serafini, L. ; Tamilin, A.: DRAGO: Distributed Reasoning Architecture for the Semantic Web. In: *Proceedings of European Semantic Web Conference (ESWC)*. Heraklion, Greece : Springer Lecture Notes in Computer Science, Mai 2005

[Stu03] Stuckenschmidt, H.: *Modularization of Ontologies*. Deliverable D.21 of the EU IST Project: WonderWeb – Ontology Infrastructure for the Semantic Web, 2003

[Sva04] Svatek, V.: Design Patterns for Semantic Web Ontologies: Motivation and Discussion. In: *Proceedings of 7th Conference on Business Information Systems*. Poznaň, 2004

[SW01] Smith, B. ; Welty, C.: Ontology: Towards a New Synthesis. In: *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*. Ogunquit, Maine, USA : ACM Press, Oktober 2001

[SW08] World Wide Web Consortium (W3C): *W3C Semantic Web Activity*. http://www.w3.org/2001/sw/, 2008

[SWBP08] World Wide Web Consortium (W3C): *W3C Semantic Web Best Practices and Deployment Working Group*. http://www.w3.org/2001/sw/BestPractices/, 2008

[TA06] Tsetsos, V. ; Anagnostopoulos, C.: Semantically Enriched Navigation for Indoor Environments. In: *International Journal of Web and Grid Services* 2 (2006), Dezember, Nr. 4, S. 453–478

[Tes01] Tessaris, S.: *Questions and Answers: Reasoning and Querying in Description Logic*, University of Manchester, Diss., 2001

[TH06] Tsarkov, D. ; Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR)* Bd. 4130, Springer, 2006, S. 292–297

[TK07] Terziyan, V. ; Katasonovic, A.: Global Understanding Environment: Applying Semantic Web to Industrial Automation. In: Cardoso, J. (Hrsg.) ; Hepp, M. (Hrsg.) ; Lytras, M. (Hrsg.): *Real-world Applications of Semantic Web Technology and Ontologies*. Springer, 2007

[TSB06] Turhan, A. ; Springer, T. ; Berger, M.: Pushing Doors for Modeling Contexts with OWL DL a Case Study. In: *Proceedings of the 4th International Conference on Pervasive Computing and Communication, Workshop on Context Modeling and Reasoning (CoMoRea)*, 2006, S. 13–17

[UJFP04] Undercoffer, J. ; Joshi, A. ; Finin, T. ; Pinkston, J.: A Target Centric Ontology for Intrusion Detection: Using DAML+OIL to Classify Intrusive Behaviors. In: *Knowledge Engineering Review* 18 (2004), S. 221–241

[Und04] Undercoffer, J.: *Intrusion Detection: Modeling System State to Detect and Classify Aberrant Behaviors*, University of Maryland, Baltimore, Diss., 2004

[Usc03] Uschold, M.: Where are the Semantics in the Semantic Web? In: *AI Magazine* 24 (2003), S. 25–36

[Usl05] Uslar, M.: Semantic interoperability within the power systems domain. In: *Proceedings of the 1st International Workshop on Interoperability of Heterogeneous Information Systems (IHIS)*. New York, NY, USA : ACM Press, 2005, S. 39–46

[Vac06] Vachtsevanos, G.: *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. Wiley, 2006

[VDE03] Informationstechnische Gesellschaft im VDE (ITG): *Positionspapier Bahnsysteme in Europa*. 2003

[VRYK03] Venkatasubramanian, V. ; Rengaswamy, R. ; Yin, K. ; Kavuri, S.: A Review of Process Fault Detection and Diagnosis. Part I: Quantitative Model-based Methods. In: *Computers and Chemical Engineering* 27 (2003), S. 293–311

[WLL+07] Weithöner, T. ; Liebig, T. ; Luther, M. ; Böhm, S. ; Henke, F. von ; Noppens, O.: Real-World Reasoning with OWL. In: *Proceedings of European Semantic Web Conference (ESWC)* Bd. 4519, Springer-Verlag, Juli 2007 (Lecture Notes in Computer Science)

[WLLB06] Weithöner, T. ; Liebig, T. ; Luther, M. ; Böhm, S.: What's Wrong with OWL Benchmarks? In: *Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS)*, 2006

[WM05] Wessel, M. ; Möller, R.: A High Performance Semantic Web Query Answering Engine. In: *Proceedings of International Workshop on Description Logics*, 2005

[WVV+01] Wache, H. ; Vögele, T. ; Visser, U. ; Stuckenschmidt, H. ; Schuster, G. ; Neumann, H. ; Hübner, S.: Ontology-Based Integration of Information – A Survey of Existing Approaches. In: *Proceedings of IJCAI-01 Workshop on Ontologies and Information Sharing*. Seattle, WA, August 2001

[YJYQ03] Yimin, W. ; Jianmin, X. ; Yucong, H. ; Qinghong, Y.: A Shortest Path Algorithm Based on Hierarchical Graph Model. In: *Intelligent Transportation Systems*, 2003

[ZML+06] Zhou, J. ; Ma, L. ; Liu, Q. ; Zhang, L. ; Yu, Y. ; Pan, Y.: Minerva: A Scalable OWL Ontology Storage and Inference System. In: *Proceedings of First Asian Semantic Web Conference (ASWC)*, 2006

[Zol08] Zolin, E.: *Description Logic Complexity Navigator*. `http://www.cs.man.ac.uk/~ezolin/dl/`. Version: März 2008

Symbolverzeichnis

$\mathfrak{A}$	verteilte ABox
$\mathcal{A}$	ABox
$\mathsf{clos}(\mathfrak{K})$	Konzepthülle der verteilten Wissensbasis $\mathfrak{K}$
$\mathcal{F}$	Komplettierungswald
$\mathfrak{G}$	Menge aller räumlichen Teil-Netzwerke in einer Hierarchie
$\text{HEPV}(G)$	hierarchisch-kodierte Pfad-Sicht für das räumliche Netzwerk G
$\mathsf{Ind}_{\cap}(\mathfrak{K})$	Menge der verteilt beschriebenen Individuennamen in der verteilten Wissensbasis $\mathfrak{K}$
$\mathsf{Ind}(\mathcal{K})$	Menge der Individuennamen in der Wissensbasis $\mathcal{K}$
$\mathcal{I}$	Interpretation
$\mathfrak{K}$	verteilte Wissensbasis
$\mathcal{K}$	Wissensbasis
$[\mathfrak{K}]_i$	Wissensbasis-Kern der verteilten Wissensbasis $\mathfrak{K}$ für Individuum i
$\mathcal{L}$	Menge der Konzept- bzw. Rollenbeschreibungen, die für einen Knoten bzw. eine Kante in einem Komplettierungswald gelten müssen
$\mathsf{msc}_i^{\mathcal{K}}$	spezifischstes Konzept, zu dem das Individuum i in der Wissensbasis $\mathcal{K}$ gehört
$\mathsf{msr}_{(i_1,i_2)}^{\mathcal{K}}$	spezifischste Rolle, die zwischen den Individuen i_1 und i_2 in der Wissensbasis $\mathcal{K}$ gilt
$M_{Q,\mathcal{K}}$	Menge der Lösungen zu einer konjunktiven Anfrage Q bezüglich Wissensbasis $\mathcal{K}$
N_C	abzählbar unendliche Menge von Konzeptnamen
N_I	abzählbar unendliche Menge von Individuennamen
N_R	abzählbar unendliche Menge von Rollennamen
N_V	abzählbar unendliche Menge von Variablennamen
Q	konjunktive Anfrage, d. h. eine Konjunktion von Anfrageatomen
$\mathfrak{R}$	verteiltes Reasoning-System
R	Reasoning-Knoten
T	Tableaux-Algorithmus
$\mathcal{T}$	TBox
$\mathsf{Terms}(Q)$	Menge aller Terme in Q, d. h. aller Variablen- und Individuennamen
$\mathsf{Var}(Q)$	Menge der Variablennamen in Q
G	räumliches Netzwerk
G_k^A	Nachbarschaftsgraph für das räumliche Netzwerk G_k
$p_k^{A,s\to t}$	abstrakter Pfad von s nach t im Nachbarschaftgraphen G_k^A

Index

Lebenslauf

Florian Fuchs

geboren am 11. Mai 1980 in München

Ausbildung

06/05 – 10/08 **Ludwig-Maximilians-Universität München**
Externer Doktorand am Lehrstuhl für Mobile und Verteilte Systeme
Promotionsstipendium von Siemens AG, Corporate Technology, Intelligent Autonomous Systems

10/99 – 04/05 **Technische Universität München**
Diplom in Informatik (Nebenfach Wirtschaftswissenschaften)
Zweijähriges, interdisziplinäres Zusatzstudium am Center for Digital Technology & Management;
Stipendium nach dem Bayerischen Begabtenförderungsgesetz

08/03 – 12/03 **University of California at Berkeley**
Visiting Student Researcher an der School of Information Management and Systems

08/02 – 05/03 **National University of Singapore**
LAOTSE-Stipendiat der Siemens AG und TU München

10/90 – 07/99 **Max-Born-Gymnasium Germering**
Allgemeine Hochschulreife

Berufserfahrung

08/02 – 11/02 **Siemens Singapore Pte. Ltd.**
Praktikant bei CIO Asia/Australia

03/01 – 04/01 **software design & management AG**
Praktikant

10/96 – 07/02 **DocuWare AG**
Softwarearchitekt und Freier Mitarbeiter