Fraunhofer IOSB
Institut für Optronik,
Systemtechnik
und Bildauswertung

Zur Behandlung von Unsicherheit und unvollständiger Information bei der videobasierten Situationserkennung

Studienarbeit
von
Ann-Kristin Grosselfinger

15. November 2011

Abteilung Objekterkennung: Dr. rer. nat. Michael Arens

Betreuer: Prof. Dr. Ing. Rainer Stiefelhagen

Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung

Gutleuthausstr. 1, 76275 Ettlingen

Direktor: Prof. Dr. Maurus Tacke

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen, als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ettlingen, den 15. November 2011

Ann-Kristin Grosselfinger

Inhaltsverzeichnis

Kapitel 1

Einleitung

Unter videobasierter Situationserkennung versteht man die automatische Verknüpfung von Videodaten mit begrifflichen Beschreibungen des Inhalts der betrachteten Videos. Anwendungen einer solchen automatischen Beschreibung sind z.B. die Suche in großen Videobeständen, die Unterstützung bei der Videoüberwachung und die semantische automatische Protokollierung. In allen Fällen verknüpft die videobasierte Situationserkennung in einem unter Umständen mehrstufigen Prozess unsichere, unvollständige und ggf. verrauschte Eingangsdaten mit vagen, dem Menschen gebräuchlichen Begriffen.

Am Fraunhofer IOSB sind Werkzeuge und Verfahren vorhanden, um in Videos etwa eines Überwachungsnetzwerkes Personen zu detektieren und zu verfolgen. Daneben sind am Fraunhofer IOSB Werkzeuge vorhanden, welche die begriffliche Schlussfolgerung auf derartigen Videoauswertungsergebnissen und somit die Verknüpfung komplexer begrifflicher Zusammenhänge mit Videos erlauben. Die hierbei eingesetzten Werkzeuge basieren zum einem auf einer unscharfen temporalen Logik, in welcher sowohl die Unsicherheit der Eingangsdaten wie auch die Vagheit der abzuleitenden Begriffe modelliert werden sollen. Zum anderen werden grafische Modelle - sogenannte Situationsgraphenbäume - eingesetzt, welche die Erkennung komplexer begrifflicher Zusammenhänge (Situationen) erlauben.

1.1 Problemstellung und Ziel der Arbeit

Im Rahmen dieser Studienarbeit soll die Theorie zum Umgang mit Unsicherheit, Vagheit und deren Verknüpfung aufgearbeitet werden und es sollen die vorhandenen Werkzeuge zur Situationserkennung um einen Mechanismus zur Behandlung der Unsicherheit - auch unter der Annahme der Unvollständigkeit - von Eingabedaten erweitert werden. Die Anwendbarkeit und Auswertung soll an bereits vorhandenen eigenen Daten sowie Standarddatensätzen gezeigt werden.

1.2 Aufbau der Arbeit

Kapitel 2 geht zuerst auf die Arbeit der Gruppe von H.-H. Nagel ein, ein komplettes System zur begrifflichen Auswertung von Bildfolgen zu realisieren, aus der einige der in dieser Arbeit verwendeten Werkzeuge hervorgegangen sind. Danach werden verschiedene, in der Literatur verfolgte, Herangehensweisen zur Situationserkennung und jeweils die wesentlichen Arbeiten dazu betrachtet. Der Abschluss des Kapitels liefert einen historischen Überblick zum Umgang mit Unsicherheit und Vagheit.
Kapitel 3 behandelt diverse Grundlagen, die einerseits dem Verständnis der vorhandenen Werkzeuge und andererseits als theoretische Grundlage möglicher Weiterentwicklungen dienen. Zu ersterem zählen der Aufbau der Logik FMTHL aus [Sch96] sowie die Grundlagen zu Situationsgraphenbäumen [Are04] und deren Traversierung [MJA11]. Für zweiteres wurde allgemeines Wissen über Unsicherheit und Vagheit unter Verwendung von [Zim01] zusammengetragen und die in [Wei96] vorgestellte Theorie über unscharfes Schließen aufgearbeitet. Kapitel 4 beschäftigt sich mit der für diese Arbeit benötigten Anpassung der Werkzeuge in Form einer neuen F-LIMETTE Regelbibliothek und Kapitel 5 beschreibt die umgesetzte Implementierung im SGT-Editor zum verbesserten Umgang mit unvollständigen Daten.
Kapitel 6 beinhaltet die Experimente auf eigenen und vorhandenen Daten und deren Auswertung. Zum Abschluss der Arbeit rekapituliert Kapitel 7 das Erreichte und liefert einen Ausblick auf mögliche Weiterentwicklungen.

Kapitel 2

Verwandte Arbeiten

2.1 Videobasierte Situationserkennung in der Arbeitsgruppe von H.-H. Nagel

Eines der Ziele der Arbeitsgruppe von H.-H. Nagel ist es einen virtuellen Kommentator zu erschaffen, der zu einem gegebenen Video die darin zu erkennenden Aktionen und Situationen in natürlicher Sprache wiedergeben kann. Der in den 80ern bis ins erste Jahrzehnt dieses Jahrhunderts vornehmlich betrachtete Diskursbereich der Arbeitsgruppe ist im Straßenverkehr angesiedelt und beschäftigt sich mit Verkehrsfluss auf Kreuzungen und mit dem Verhalten an Tankstellen. Es wurde ein komplettes System zur begrifflichen Auswertung von Bildfolgen aufgebaut. Dabei orientierte man sich an folgenden zu bewältigenden Teilaufgaben: Detektion, Objektbeschreibung, Umgebungsbeschreibung, Klassifikation, Bewegungsbeschreibung, Initialisierung, Verfolgung, Geschehensbeschreibung, Erkennung von Abläufen, Situationserkennung und Erkennung von Zielen der Agenten.
Die ursprünglichen Ansätze kann man dem 10-Jahres Bericht [BCN95] entnehmen und einen Überblick der weiteren Entwicklung findet man in [Nag04].
In der Arbeitsgruppe entstanden die Tracking-Systeme XTrack und dessen Nachfolger MOTRIS (MOdel-based TRacking in Image Sequences), siehe Abbildung 2.1. Dieses Teilsystem basiert zum Teil auf optischem Fluss, dessen Einsatz seit [Kol95] kontinuierlich weiterentwickelt wurde, beispielsweise in den Arbeiten [Mid04] und [ON08]. Es werden generische Fahrzeugmodelle verwendet, siehe [Kol92], sowie ein Modell für automatische Ermittlung der Szenen-Beleuchtung, das man der Arbeit [ON03] entnehmen kann. Es wird datengetriebene Fahrbahnermittlung, nach [Müc00] verwendet. Die Verfolgung basiert auf einem Iterativen Erweiterten Kalman Filter und nutzt unter anderem Bewegungsmodelle aufbauend auf [Kol92] oder [LN99]. Neben optischem Fluss werden auch Kantenelemente zur Positionsbestimmung bei der Verfolgung verwendet.

Das Tracking-System MOTRIS liefert zu einer gegebenen Bildfolge Zustandsvektoren je Agent und Einzelbild. Jeder Zustandsvektor enthält Informationen über Position,

Geschwindigkeit und Lage, meist in Form von zwei Koordinaten in der Fahrbahnebene, Translationsgeschwindigkeit, Winkelgeschwindigkeit oder Lenkwinkel und Orientierung bezüglich einer Referenzrichtung.

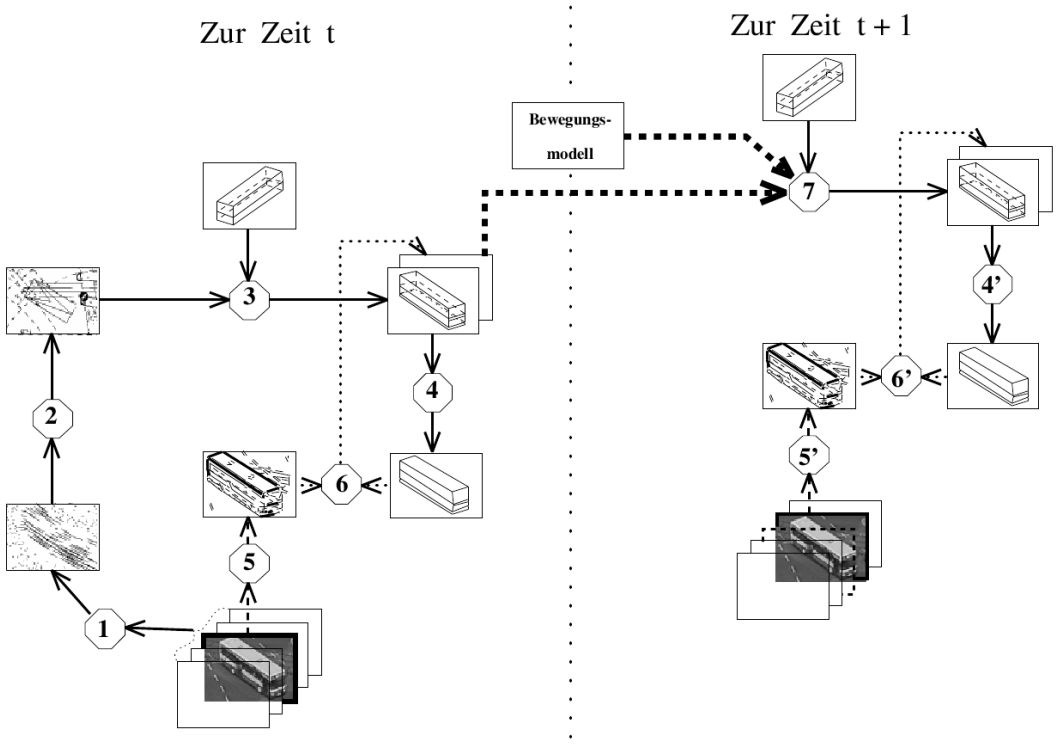

Abbildung 2.1: Iteratives Bewegungsmodell des Trackingsystems MOTRIS. Ausgehend von einer Bildfolge wird mittels optischem Fluss ein Objektbildkandidat bestimmt (1), zur Bestimmung von Szenenbereichshinweisen in die Straßenebene zurückprojiziert (2), aus der daraus gewonnenen Hypothese über Modellparameter ein Objektkandidat bestimmt (3) und um Wissen aus der Kalibrierung ergänzt. Es werden Bildbereichshinweise in Form von Kantenelementen bestimmt (5) und diese den Modellsegmenten zugeordnet (6), wonach das Modell iterativ angepasst wird. Der Teilprozess (7) liefert anhand eines Bewegungsmodells und der Modellausprägung zum Zeitpunkt t eine Vorhersage für jene zum Zeitpunkt $t + 1$. Abbildung aus [Nag96].

Das begriffliche Teilsystem verwendet diese Zustandsvektoren, um auf rechnerinterne Repräsentationen von Aussagen schließen zu können. Hierzu benötigt man zunächst Geschehensbeschreibungen, das heißt Zuordnungen elementarer begrifflicher Beschreibungen zu geometrischen Bewegungsbeschreibungen. Die Arbeitsgruppe beschäftigte sich hierzu mit der rechnerinternen Repräsentation von Bewegungsverben. Es konnten in der Arbeit [Cah88] ungefähr 90 Bewegungsverben unterschiedlicher Bedeutung ermittelt werden, die sich aus Bildfolgen extrahierten Fahrzeug-Trajektorien zuord-

nen lassen, siehe Abbildung 2.2. Zur Bewerkstelligung dieses Übergangs (Semantische Lücke) von quantitativen zu qualitativen Daten und zur Schlussfolgerung wird unscharfe metrisch-temporale Logik eingeschränkt auf das Hornfragment (FMTHL) und das dazugehörige Logiksystem F-LIMETTE, wie man es [Sch96] entnehmen kann, verwendet. Ausgehend von Bewegungsverben werden geschehensspezifische Sätze von je drei Prädikaten für Vorbedingung, Monotonie-Bedingung und Nachbedingung in FMTHL gebildet. Die Situationserkennung geschieht daraufhin unter Verwendung von unscharfen Zuordnungsautomaten, wie es in der Arbeit [GN08] beschrieben ist.

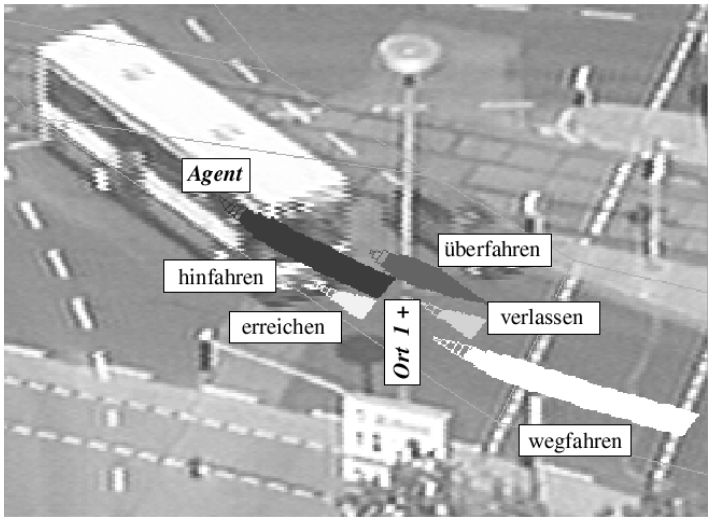

Abbildung 2.2: Trajektorien zugeordnete Bewegungsverben. Abbildung aus [Nag96].

Eine andere, in der Arbeitsgruppe von H.-H. Nagel weiterentwickelte, Art der Repräsentation und Nutzung von Hintergrundwissen zur Situationserkennung ist der Aufbau und die Traversierung eines Situationsgraphenbaumes (SGT), siehe [Are04]. Jüngste Arbeiten zur Nutzung von SGTs und F-LIMETTE sind [AGN08], [MJA11] und [MIAS11]. Ebenfalls interessant ist die Arbeit [Nag04], worin SGTs nicht nur zur Bereitstellung von Hintergrundwissen und zur Verknüpfung einzelner elementarer Fahrzeugmanöver genutzt werden, sondern auch zur Vorhersage von Verhalten eines Agenten als Feedback für das Bewegungsmodell zur Überbrückung von Verdeckungssituationen.

2.2 Ansätze zur Situationserkennung

Bei der Situationserkennung unterscheidet man allgemein zwischen direkten Ansätzen, die komplett ohne menschliches Verhaltenswissen auskommen, dafür allerdings größere Trainingsdatensätze benötigen und hierarchischen Ansätzen, die durch Einfließenlassen

von menschlichem Verhaltenswissen komplexere Aktivitäten aus Abfolgen oder dem Zusammenspiel einfacherer Aktivitäten bilden. Eine Übersicht liefert [AR11]. Ein Beispiel für einen direkten Ansatz ist die Arbeit [LMSR08] zur Erkennung menschlicher Aktionen wie 'Hände schütteln', 'Umarmung', 'sich hinsetzen' oder 'aus dem Auto steigen' in Spielfilmen.

Die hierarchischen Ansätze für die Situationserkennung in Bildfolgen bilden mit Hilfe von Hintergrundwissen schichtweise immer komplexere Aktivitäten aus der Kombination einfacherer. Zur Erkennung atomarer Aktionen, für die unterste Schicht dieser Hierarchie, werden wiederum direkte Ansätze verwendet. Man kann die hierarchischen Ansätze in drei Typen unterteilen: Statistische Ansätze, syntaktische Ansätze und beschreibungsbasierte Ansätze.

2.2.1 Statistische hierarchische Ansätze

Statistische hierarchische Ansätze nutzen mehrere Schichten von dynamischen Bayes-Netzen meist in Form von Hidden Markov Models (HMMs) oder deren Weiterentwicklungen. Ein HMM ist ein stochastischer endlicher Automat, der zwei Zufallsprozesse modelliert. Der eine Zufallsprozess entspricht einer Markov-Kette und bildet den Zustandsübergang des Modells ab, ist jedoch nicht direkt beobachtbar, daher der Name 'hidden', der andere liefert die beobachtbaren Auswirkungen des ersten. Da man die Zustände des Modells nicht direkt aus der Beobachtung erschließen kann, werden zu Beobachtungen, in Form von annotierten Trainingsdaten oder Ergebnissen der darüberliegenden Schicht, die wahrscheinlichsten Sequenzen von Zuständen ermittelt, die diese Beobachtungen produziert haben könnten. Dieser Ansatz kommt sehr gut mit verrauschten Eingangsdaten zurecht. Er ist gut geeignet für sequentielle Handlungen, aber nicht für solche mit zeitlich komplexer Struktur oder tiefen hierarchischen Strukturen.

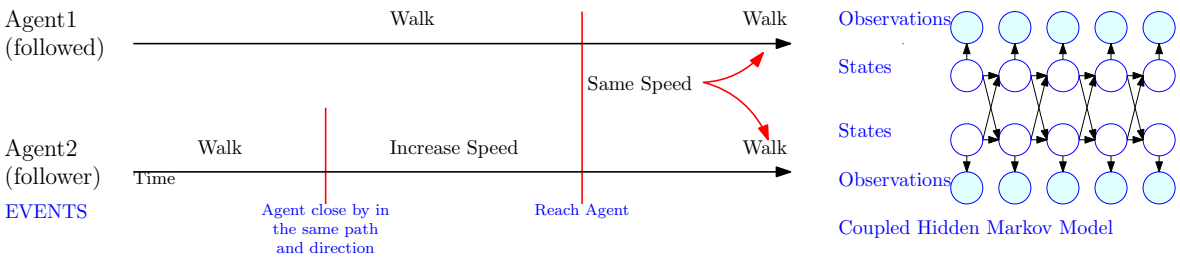

Abbildung 2.3: Rosario et al. [ROP99] nutzen gekoppelte HMMs für das gemeinsame Verhalten zweier Personen. Beispielsweise sind die HMMs für die Beobachtung des Verhaltens eines Verfolgers und des Verhaltens eines Verfolgten hierbei im 'hidden' Teil gekoppelt, um das gemeinsame Verhalten einer Person, die eine andere verfolgt zu modellieren.

Mittlerweile werden allerdings verschiedene, an die Bedürfnisse bei der Bildfolgenauswertung angepasste, Varianten von HMMs verwendet, wie Variable Length Hidden Markov Models (VLMMs) mit Markov-Ketten variabler Länge, siehe [GCMH02] oder Coupled Hidden Markov Models (CHMMs), wobei HMMs für zeitlich oder räumlich zusammenhängende Komponenten einer Aktion gekoppelt werden. Beispiele für gekoppelte Komponenten sind die rechte und linke Hand bei beidhändigen Aktionen [BOP97] oder Verfolger und Verfolgter im gemeinsam betrachteten Verhalten von zwei Personen [ROP99], siehe Abbildung 2.3. Weitere Beispiele für die Verwendung von HMMs sind die Arbeiten [HTG09] und [NPVB05, KHM11], wo hierarchische Hidden Markov Models (HHMMs) zur Erkennung von Situationen eingesetzt werden. Es gibt ebenfalls verschiedene Ansätze die auf Bayes-Netzen aufbauen, so beispielsweise 'Propagation Network' in der Arbeit [SHM+04].

Abbildung 2.4: Aktionserkennung mittels Kontextfreier Grammatik: start, put, put, grab, touch, touch, put, grab, end. [Bra97].

2.2.2 Syntaktische hierarchische Ansätze

Bei syntaktischen hierarchischen Ansätzen werden Situationen symbolisch als Zeichenketten repräsentiert, wobei jedes Symbol eine atomare Aktion repräsentiert. Diese Zeichenketten werden mittels verschachtelter Produktionsregeln, wie man sie von formalen Grammatiken kennt, generiert. Zur Situationserkennung erfolgt eine Syntaxanalyse dieser Zeichenfolgen. Diese Ansätze eignen sich vor allem für zyklische Aktivitäten und solche mit einer tiefen hierarchischen Struktur, allerdings nicht für fehlerbehaftete oder unsichere Systeme.
Vertreter hierfür sind [Bra97] worin beispielsweise die Arbeitsschritte zum Aufschrauben eines Computers erkannt werden, siehe Abbildung 2.4, die Arbeiten [BI98] und [IB00], siehe Abbildung 2.5, in denen es um Erkennen von Zeigegesten geht und [ME02] die sich mit dem Kartenspiel Blackjack befasst, sowie die Arbeit [GFA07]. Stochastisch Kontextfreie Grammatiken (SCFGs) werden in [IB00, ME01] zur Erkennung von Situationen mit mehreren Agenten eingesetzt. Die Arbeit [KSS08] behandelt den Umgang mit Unsicherheit bei solchen grammatikbasierten Ansätzen.

2.2.3 Beschreibungsbasierte hierarchische Ansätze

Beschreibungsbasierte hierarchische Ansätze nutzen Beschreibungslogiken oder sprachlich gefasste zeitliche und örtliche Zusammenhänge und Begrifflichkeiten, sogenannte

$$
\begin{aligned}
G_{square} : \text{SQUARE} &\rightarrow \text{RH} \quad | \quad \text{LH} \\
\text{RH} &\rightarrow \text{TOP} \quad \text{UD} \quad \text{BOT} \quad \text{DU} \\
\text{LH} &\rightarrow \text{BOT} \quad \text{DU} \quad \text{TOP} \quad \text{UD} \\
\text{TOP} &\rightarrow \text{LR} \quad | \quad \text{RL} \\
\text{BOT} &\rightarrow \text{RL} \quad | \quad \text{LR} \\
\text{LR} &\rightarrow left - right \\
\text{UD} &\rightarrow up - down \\
\text{RL} &\rightarrow right - left \\
\text{DU} &\rightarrow down - up
\end{aligned}
$$

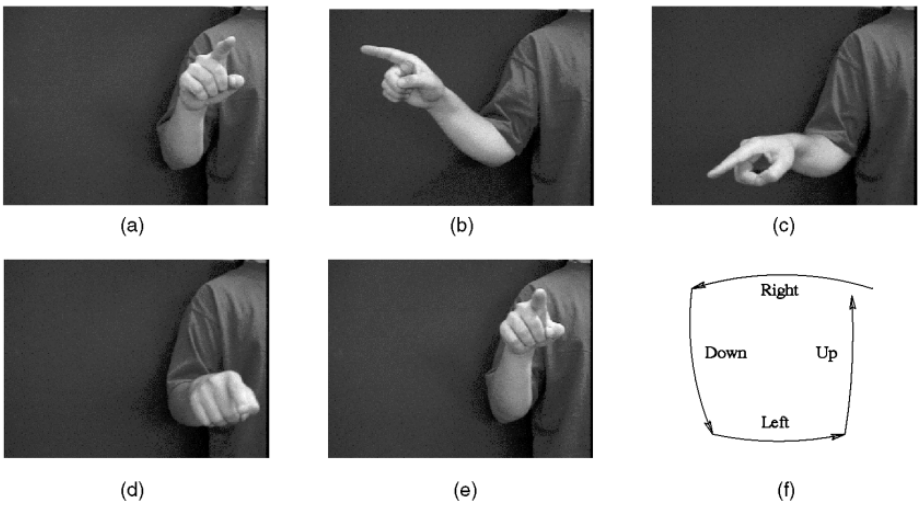

Abbildung 2.5: Kontextfreie Grammatik zur Gestenerkennung, in diesem Fall zur Erkennung eines gezeigten Quadrates, aus [IB00].

Ontologien, als Grundlage zur Formulierung von Eigenschaften verschiedener Situationen. Die sprachliche Repräsentation erleichtert es dem Nutzer Hintergrundwissen einzupflegen. Aktionen werden üblicherweise durch Verkettung von Handlungen charakterisiert, die wiederum selbst Verkettungen von Teilhandlungen sind. Die meisten Ansätze nutzen dabei Logik zur Repräsentation zeitlicher Zusammenhänge, sei es Allens Intervallalgebra [All83] (siehe Abb. 2.6) in [PB98, NZH03], oder temporale Logik wie beispielsweise FMTHL von [Sch96] in [Are04, AGN08, MJA11, MIAS11].

Zur Repräsentation des Hintergrundwissens entwickelten Pinhanez und Bobick [PB98] das PNF-Netzwerk (past-now-future-network), siehe Abbildung 2.7, aufbauend auf den Intervall-Algebra-Netzwerken (IA-network) von Allen [All83].

Relation	Symbol	inverses Symbol	Piktogramm
X before Y	<	>	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Abbildung 2.6: Die 13 möglichen einfachen Beziehungen zwischen zwei Zeitintervallen, aus [All83].

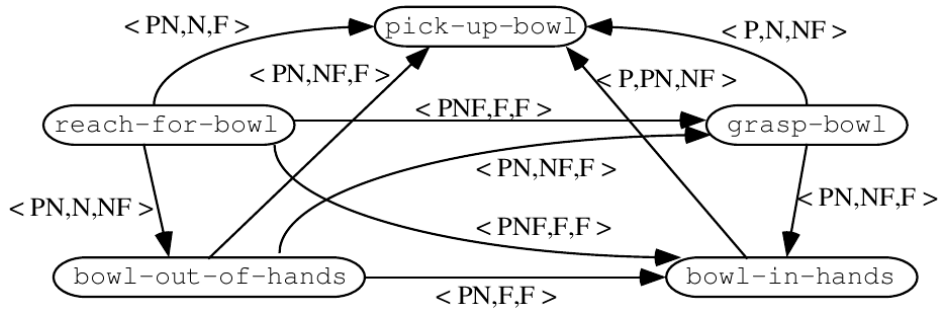

Abbildung 2.7: Past-Now-Future-Netzwerk zu einer Küchenszene aus [PB98]. Die gerichteten Kanten modellieren die temporalen Bedingungen zwischen den verschiedenen atomaren Handlungen. Die Tripel < . , . , . > geben zu möglichen Bedingungs-Tupeln die zweiten Stellen nach *past*, *now* beziehungsweise *future* an. Dabei steht P für *past*, N für *now* und F für *future* sowie entsprechend PN für *past* oder *now*.
< PN, F, F > ist somit die Kurzschreibweise für (P,P),(P,N),(N,F),(F,F).

In den Arbeiten von Intille und Bobick [IB99a, IB99b, IB01] werden Ziele einzelner Agenten durch 'Visual Goal Networks', einer Weiterentwicklung von Bayes'schen Netzen, repräsentiert. Diese Netzwerke bilden ein Baukastensystem für Aktivitäten an denen viele Agenten beteiligt sind, wie beispielsweise bei einem American Football Spiel.

In einigen Ansätzen werden die Informationen vom Benutzer in einer Art Programmiersprache eingegeben und danach in einen Graphen umgewandelt. So bei der Sprache VERL (Video Event Representation Language) von [NZH03], in der hierarchisch angeordnete Datentypen ähnlich einer objektorientierten Sprache bestimmte Eigenschaften besitzen. Ein Mensch ist hierbei zum Beispiel ein bewegliches Objekt und als solches ein Objekt und hat somit die Eigenschaften dieser drei Datentypen und damit auch beispielsweise die Eigenschaft 'Geschwindigkeit' von beweglichen Objekten. Einfache Abläufe, die sich aus Eigenschaften zusammensetzen, können auf unterschiedliche Art und Weise (UND-verknüpft, sequentiell oder durch ein Bayes'sches Netz) zu sogenannten Single-Thread Events kombiniert werden und diese wiederum zu Multi-Thread Events.

```
Scenario(Attack,
  Characters((cashier : Person), (robber : Person))
  SubScenarios(
    (cas_at_pos, inside_zone, cashier, "Back_Counter")
    (rob_enters,changes_zone,robber, "Entrance_zone","Infront_Counter")
    (cas_at_safe, inside_zone, cashier, "Safe")
    (rob_at_safe, inside_zone, robber, "Safe") )
  Constraints( (rob_enters during cas_at_pos)
    (rob_enters before cas_at_safe)
    (cas_at_pos before cas_at_safe)
    (rob_enters before rob_at_safe)
    (rob_at_safe during cas_at_safe) ) )
```

Abbildung 2.8: Bankraub-Scenario 'Attack' aus [VBT03] für einen Kassierer und einen Bankräuber bestehend aus vier Teilscenarios ((1) Der Kassierer ist an seiner Position hinter dem Schalter. (2) Der Bankräuber betritt die Bank und geht auf den Schalter zu. (3) und (4) Kassierer und Bankräuber sind am Safe.) und zugehörigen zeitlichen Abfolgebedingungen.

Scenarios von [VBT03] ähneln einem Drehbuch für zu erkennende Situationen, bestehend aus Name, beteiligten Charakteren und Objekten, Teilhandlungen sowie zeitlichen Abfolgebedingungen wie in Abbildung 2.8. Dies verwendet die Arbeit [PBC+08], in der die 'beteiligten Charaktere' eingeteilt werden in zu beobachtende 'bewegliche Objekte', unterteilt in semantische Klassen wie Person, Gruppe, Menge, Gepäck, und sogenannte 'contextual objects' für die vom Benutzer eingegebene Kulisse aus unbeweglichen, oder vordefinierbar beweglichen Teilen der Ausstattung in der Szene wie beispielsweise eine Tür. Die verwendeten Ereignisse wurden beispielsweise wie folgt definiert: stays_inside_zone(o, z, T) was bedeutet, dass ein Objekt *o* sich für mindestens

T Sekunden innerhalb der Zone z befindet.
Eine Variante, die Bedingungen wie in den programmiersprachenähnlichen Varianten zulässt, die zugleich direkt als Graph angelegt ist, liefert der in Kapitel 2.1 bereits erwähnte SGT. Neben den dort genannten Arbeiten zu Situationen im Straßenverkehr beschäftigt sich in Barcelona eine Arbeitsgruppe mit dem Einsatz von SGTs zur Erkennung von Situationen in U-Bahn-Stationen [GRVR09], sowie am Fraunhofer IOSB in Karlsruhe mit Situationen in einem Smart-Control-Room und in Ettlingen mit Situationen von Fußgängern [MJA11]. Eine detaillierte Beschreibung von SGTs findet man im Kapitel Grundlagen 3.2.

2.3 Der Umgang mit Unsicherheit und Vagheit

Der Ursprung vieler Theorien zum Umgang mit Unsicherheit und Vagheit liegt in der Wahrscheinlichkeitstheorie. Einen Überblick liefert [JB03], der schreibt, dass seine Herangehensweise unter anderem auf [Pól54] beruht. Eines der vielen Grundlagenbücher ist [Jef65].
Die Grundsteine heutiger Theorien wurden meist schon vor längerer Zeit gelegt, so wurde beispielsweise die Bayes'sche Regel, die u.a. in der Spieltheorie den Lernprozess eines Spielers in einem Spiel mit unvollkommener Information beschreibt, bereits 1763 in [Bay63] vorgestellt. Den axiomatischen Aufbau der Wahrscheinlichkeitstheorie verdanken wir Kolmogorov [Kol50].
Das Buch [Pea88] und [Pea94] beschäftigen sich umfangreich mit Schlussfolgerung bei Unsicherheit und Verwendung von Bayes'schen Netzen und [Coo90, DL93] zeigen, dass die Verwendung von Bayes'schen Netzen zur Schlussfolgerung NP-Hart ist.
Die Dempster-Shafer-Theorie, siehe auch Kapitel 3.4.9, entstand durch die Arbeit [Dem68], erweitert durch [Sha76]. Ihre weitere Entwicklung kann man [YKF94] entnehmen.
Unscharfe Mengen [Zad65] und unscharfe Logik [Zad75] wurden von Zadeh eingeführt und geprägt. Einen Überblick liefert [DPY93]. Einiges zur Behandlung von Unsicherheit im Zusammenhang mit Schlussfolgerung liefert [Hal03]. Siehe auch Kapitel 3.4.

Kapitel 3

Grundlagen

Die begriffliche Auswertung von Bildfolgen in einem hierarchischen Schichtenmodell mit interaktivem, visuellem und begrifflichem Teilsystem [Nag00], siehe Abbildung 3.1, benötigt a-priori Verhaltenswissen für das begriffliche Teilsystem. Dieses Wissen kann durch Situationsgraphenbäume (SGTs) repräsentiert werden. Zur Bereitstellung von Regeln für einfache Situationsbeschreibungen in SGTs und für automatische Schlussfolgerung auf den begrifflichen Repräsentationen wird unscharfe metrisch-temporale Logik eingeschränkt auf das Hornfragment (FMTHL) und das dazugehörige Inferenzsystem F-LIMETTE verwendet. Da die natürliche Sprache voller vager Begriffe ist und man mit verrauschten Daten vom visuellen Teilsystem rechnen muss, ist die gemeinsame Betrachtung von Unsicherheit und Vagheit nötig. Im Folgenden werden FMTHL, F-LIMETTE und SGTs vorgestellt und einige Grundlagen zu Unsicherheit und Vagheit geliefert.

3.1 Unscharfe Metrisch-Temporale Logik und das Inferenzsystem F-LIMETTE

Die unscharfe metrisch-temporale Logik FMTHL entstand durch schrittweise Erweiterung der Prädikatenlogik erster Ordnung (PL1) durch [Sch96]. Er erweiterte die PL1 einerseits um Behandlung metrisch-temporaler Ausdrücke zur metrisch-temporalen Logik MTL und andererseits um Behandlung von Unschärfe, wodurch die unscharfe Logik erster Ordnung FL1 entstand. Beide Erweiterungen schränkte er auf das Horn-Logische Fragment ein, um effiziente rechnergestützte Schlussfolgerung gewährleisten zu können und erhielt damit die metrisch-temporale Hornlogik (MTHL) und die unscharfe Hornlogik (FHL). Diese beiden vereinigte er schließlich zur unscharfen metrisch-temporalen Logik eingeschränkt auf das Hornfragment FMTHL.

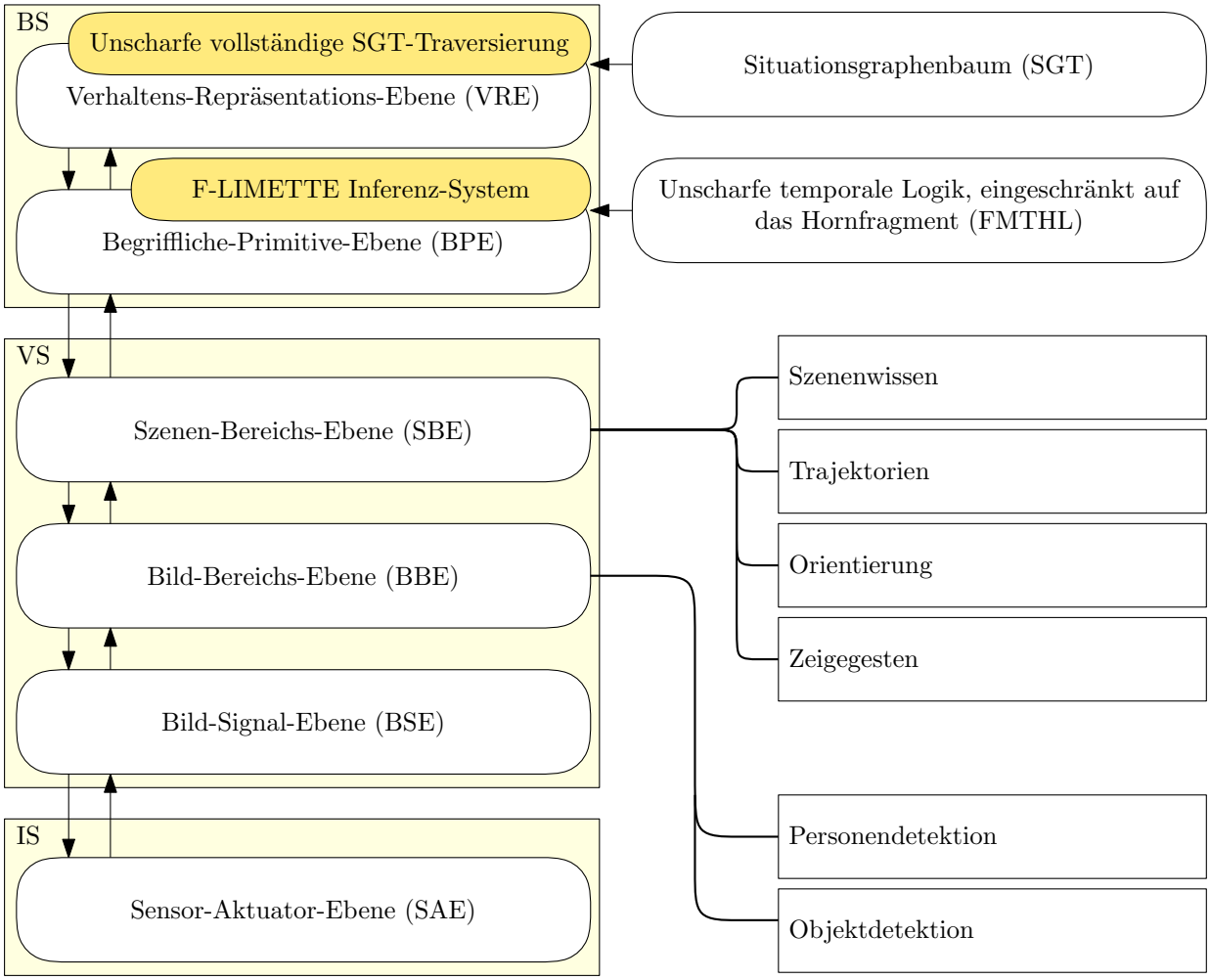

Abbildung 3.1: Schichtmodell eines kognitiven Systems basierend auf [Are04, Nag00, MIAS11]. Das interaktive Teilsystem (IS) sammelt Daten von Sensoren und Aktuatoren. Das visuelle Teilsystem (VS) repräsentiert verschiedene Darstellungsformen visueller Informationen: 1D Bildpunkten (BSE), 2D Bildmerkmalen (BBE) und geometrischen Lagedaten in 3D (SBE). Die Kästen rechts unten stellen mögliche Wahrnehmungsmodalitäten dar. Das begriffliche Teilsystem enthält Hintergrundwissen in Form von FMTHL Regeln und SGTs.

Mit FMTHL kann Wissen in Form von Fakten und Regeln repräsentiert werden, das zu zeitveränderlichen Diskursbereichen gehört und das durch linguistische Variablen [Wei96] vage oder durch verrauschte Eingabedaten unsicher ist [Are04].

Die MTL erweitert die PL1 um eine Zeitstruktur $(\mathcal{T}, t_0, \prec)$ und temporale Operatoren $(\circ, \bullet, \Box, \diamond, \mathcal{S}_S, \mathcal{U}_S)$. Dabei ist $\mathcal{T}$ eine ganzzahlige Zeitpunktmenge, die kein zusammenhängendes Intervall sein muss, t_0 ein Referenzzeitpunkt und $\prec$ eine Ordnungsrelation auf $\mathcal{T}$. Die Operatoren 'next' $\circ$ und 'previous' $\bullet$ bezeichnen den nächsten bzw. vorangehenden Zeitpunkt. Zur Formel $\mathcal{F} \in \mathcal{F}_\Sigma(\mathcal{V})$ bezeichnet $\Box\mathcal{F}$ 'uneingeschränkt immer', $\diamond\mathcal{F}$ 'irgendwann', $\Box_S\mathcal{F}$ 'immer innerhalb einer Menge $S \subseteq \mathbb{Z}$ von Zeitpunkten' und $\diamond_S\mathcal{F}$ 'irgendwann innerhalb einer Menge $S \subseteq \mathbb{Z}$ von Zeitpunkten'. Desweiteren 'since' $\mathcal{F}_1\mathcal{S}_c\mathcal{F}_2$ '$\mathcal{F}_1$ immer seit $\mathcal{F}_2$ innerhalb von S' und 'until' $\mathcal{F}_1\mathcal{U}_c\mathcal{F}_2$ '$\mathcal{F}_1$ immer bis $\mathcal{F}_2$ innerhalb von S'.

Die FL1 entsteht aus der PL1 durch Erweiterung der Wahrheitswerte $\{0, 1\}$ auf das Intervall $[0, 1]$. Außerdem werden Operatoren für 'unscharfe Abschwächung' $\downarrow_\kappa$ (Wahrheitswert von $\mathcal{F} \geq \kappa \Rightarrow \downarrow_\kappa \mathcal{F}$ absolut wahr) und 'unscharfe Verstärkung' $\uparrow_\lambda$ ($\mathcal{F}$ absolut wahr $\Rightarrow \uparrow_\lambda \mathcal{F}$ hat wahrheitswert λ) eingeführt sowie Varianten der logischen Operatoren, die mit weak, medium und strong bezeichnet werden, siehe Tabelle 3.2.

$v \in$	$\{w, m, s\}$	weak	medium	strong
Konjunktion	$x \wedge_v y$	$min\{x, y\}$	$x * y$	$max\{0, x+y-1\}$
Disjunktion	$x \vee_v y$	$min\{1, x+y\}$	$x + y - x * y$	$max\{x, y\}$
Negation	$\neg_v(x)$	$1 - x^2$	$1 - x$	$1 - \sqrt{x}$
Subjunktion	$y \leftarrow_v x$	$\neg_m x \vee_w y$	$\neg_m x \vee_m y$	$\neg_m x \vee_s y$

Abbildung 3.2: Verschiedene Semantiken der unscharfen Operatoren jeweils für Konjunktion $(\wedge_w, \wedge_m, \wedge_s)$, Disjunktion $(\vee_w, \vee_m, \vee_s)$, Negation $(\neg_w, \neg_m, \neg_s)$ und Subjunktion $(\leftarrow_w, \leftarrow_m, \leftarrow_s)$.

Die Einschränkung auf das Horn-Fragment bedeutet, dass nur Hornformeln, also Konjunktion von Klauseln, die maximal ein positives Literal besitzen, verwendet werden:

$$(A \vee \neg B_1 \vee \ldots \vee \neg B_n) \quad \text{in PL1 logisch äquivalent zu} \quad (A \leftarrow B_1 \wedge \ldots \wedge B_n) \tag{3.1}$$

$$(A) \quad \text{in PL1 logisch äquivalent zu} \quad (A \leftarrow 1) \tag{3.2}$$

$$(\neg B_1 \vee \ldots \vee \neg B_n) \quad \text{in PL1 logisch äquivalent zu} \quad (0 \leftarrow B_1 \wedge \ldots \wedge B_n) \tag{3.3}$$

Die Menge der definiten Hornformeln (genau ein positives Literal, 3.1, 3.2) enthält Fakten $\widetilde{\forall} A$ und Regeln $\widetilde{\forall}(A \leftarrow B_1 \wedge \ldots \wedge B_n)$.
Die Menge der indefiniten Hornformeln (kein positives Literal, 3.3) besteht aus Anfragen der Form $\widetilde{\exists}(B_1 \wedge \ldots \wedge B_n)$. Die Klauseldarstellung existiert im temporalen sowie im unscharfen Fall nicht.
Aufbauend auf der Logik FMTHL wurde das Inferenzsystem F-LIMETTE (Fuzzy Logic Programming Integrating Metric Temporal Extensions) als logisches Programm realisiert.
Während im aussagenlogischen Fall die Einschränkung auf Hornformeln bewirkt, dass in linearer Anzahl Schritte entscheidbar ist, ob ein Atom aus einer Wissensbasis folgt, ist in der Prädikatenlogik erster Ordnung das Erfüllbarkeitsproblem nur semi-scheidbar wenn man sich auf das Hornfragment einschränkt. Man kann problemlos Hornklauseln angeben, die zu unendlichen Auswertungssequenzen führen, wie beispielsweise: $\forall x \forall y. KleinerAls(nachfolger(x), y) \leftarrow KleinerAls(x, y)$. Es liegt in der Verantwortung des Benutzers, unendliche Lösungszweige zu vermeiden, was durch die Einschränkung auf das Hornfragment erleichtert ist, siehe [BL04]. Durch die metrisch-temporale Erweiterung sowie die Erweiterung um Unschärfe 'verschlechtert' sich die Erfüllbarkeit. Das von [Sch96] vorgestellte Tableaukalkül für FMTHL, auf dem F-LIMETTE aufbaut, bewältigt jedoch zumindest jede Regelanwendung in konstant beschränkter Zeit.
Die temporale Prädikatenlogik ist unvollständig, daher können Anfragen bezüglich der Lebendigkeitseigenschaft $\Box \diamond \mathcal{F}$ 'immer irgendwann $\mathcal{F}$' in temporallogischen Programmen nicht berücksichtigt werden, ohne die Vollständigkeit aufgeben zu müssen [Brz96]. Der Kopf einer Hornformel kann nicht $\diamond_S \mathcal{F}$ enthalten, da dies einer Disjunktion $\bigvee_{i=1}^{n} \circ^i \mathcal{F}$ entspricht und man somit mehr als ein positives Literal hätte. [Brz96]

Da variable Zeitschranken in Anfragen, also $\Box_x \mathcal{F}$ oder $\diamond_x \mathcal{F}$ für eine Variable x, nicht möglich sind, braucht es ein weiteres Teilsystem zur gezielten Betrachtung einzelner Zeitabschnitte in der Situationsanalyse: Einen zeitabschnittsweise betrachtbaren Situationsgraphenbaum (SGT).

3.2 Situationsgraphenbaum

SGTs werden genutzt um Teilzusammenhänge im zeitlichen Kontext abzubilden. Sie stellen die Verknüpfung von Situationsabfolgen mit Handlungen dar. Auch Zustandsänderungen eines Agenten selbst oder des Diskursbereiches durch Handlung von Agenten können abgebildet werden. Bei den Informationen in SGTs handelt es sich um

vom Mensch eingebrachtes Verhaltenswissen. SGTs können als gerichtete Hypergraphen oder kontextfreie Grammatiken aufgefasst werden, vergleiche [Are04].

3.2.1 Situationsschema

Die Knoten eines Situationsgraphenbaumes werden durch Situationsschemata gebildet. Jedes Situationsschema besitzt einen eindeutigen Namen und besteht aus einem Zustands- und Handlungsschema, siehe Abbildung 3.3. Das Zustandsschema enthält in Form von Prädikaten die Bedingungen für die Ausprägung dieser Situation (Vorbedingung). Das Handlungsschema enthält Prädikate, die falls die Zustandsprädikate erfüllt sind, mit deren erfüllender Variablenbelegung ausgewertet werden (Nachbedingung). Situationsschemata können als Start- oder Endsituation markiert werden und somit in dem, im nächsten Abschnitt beschriebenen, Situationsgraphen als Einstiegspunkt bzw. Endpunkt einer Folge auszuprägender Situationsschemata dienen. Eine Situation kann gleichzeitig Start- und Endsituation sein und ein Situationsgraph kann mehrere Start- und Endsituationen besitzen.

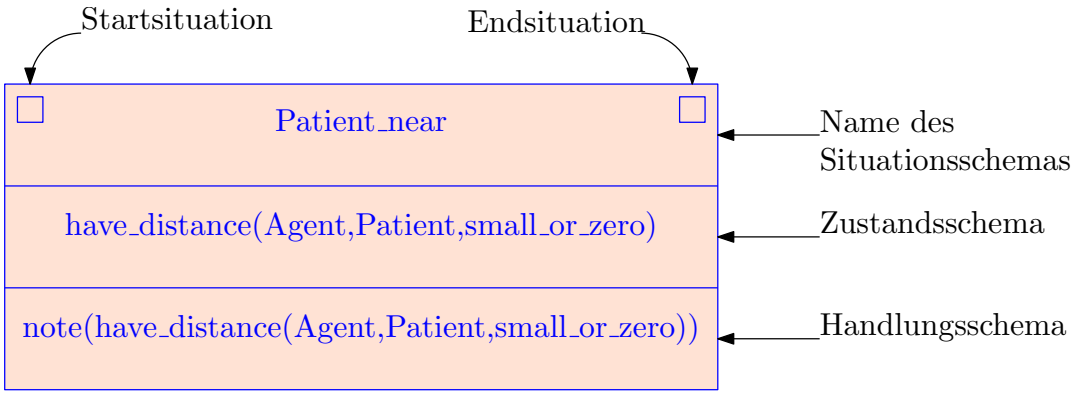

Abbildung 3.3: Darstellung eines Situationsschemas im SGT-Editor.

3.2.2 Situationsgraph

Die Situationsschema-Knoten werden durch gerichteten Prädiktionskanten, die die temporale Abfolge der Situationen modellieren, zu Situationsgraphen zusammengefügt, siehe Abbildung 3.4. Auf mehreren vom selben Schema ausgehenden Prädiktionskanten existiert eine Ordnung. Schleifen und Selbstprädiktionen (Kante von einen Situationsschema zu sich selbst) sind erlaubt.
Möchte man beim Übergang von einem Situationsschema entlang einer Prädiktionskante zu einem anderen die Neubelegung von Variablen zulassen, so muss man (für

jede Kombination aus Übergang und Variable, für die man das möchte) explizit diese Bindung lösen. Dies wird im Bindungsschema festgehalten.

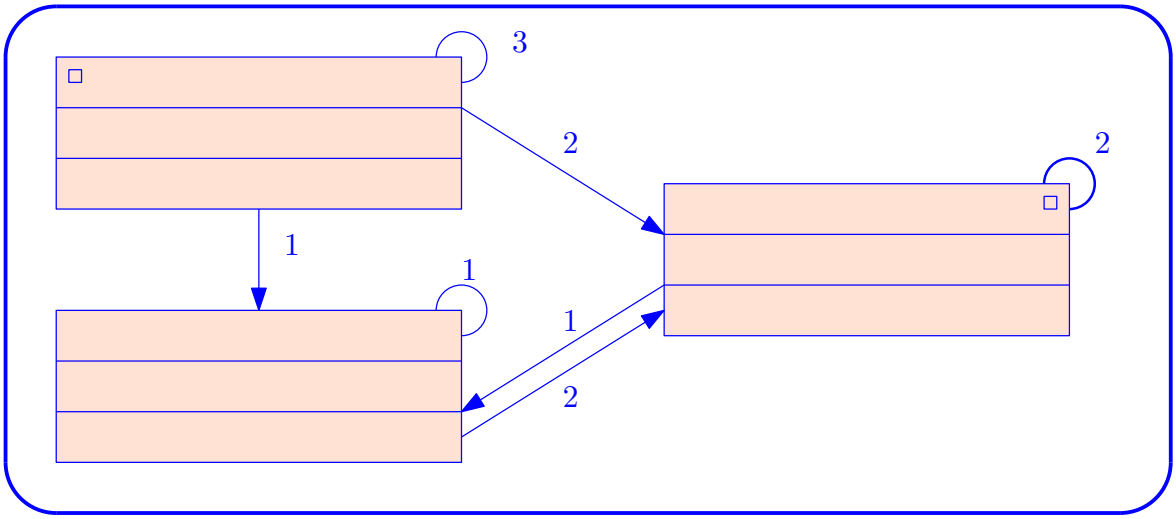

Abbildung 3.4: Darstellung eines Situationsgraphen im SGT-Editor, bestehend aus drei Situationsschemata, die durch geordnete Prädiktionskanten verbunden sind.

3.2.3 Von Situationsgraphen zum Situationsgraphenbaum

Durch die Anordnung von mehreren Situationsgraphen mit Detaillierungskanten erhält man einen Situationsgraphenbaum. Detaillierungskanten führen jeweils von einem Situationsschema zu einem Situationsgraph, der dieses Situationsschema detailliert. Bei der Detaillierung kann es sich um begriffliche Spezialisierung oder zeitliche Zerlegung handeln. Unter ersteres würden beispielsweise ‘zusammen laufen’ und ‘zusammen rennen’ als Spezialisierung von ‘zusammen bewegen’ fallen. Zweiteres wäre beispielsweise die Aufsplittung von ‘Kreuzung überfahren’ zu ‘heranfahren’, ‘Vorfahrt gewähren’, ‘Kreuzung überqueren’ und ‘Kreuzung verlassen’. Auf mehreren Detaillierungskanten, die von derselben Situation ausgehen, existiert eine Ordnung.
Zu jedem Situationsgraph dürfen nur Detaillierungskanten aus einem Situationsschema führen und Detaillierungskanten dürfen nicht zu allgemeineren Situationsschemata zurückführen, dadurch ergibt sich eine Baumstruktur. Der Wurzelgraph ist derjenige unter den Situationsgraphen, der kein Situationsschema detailliert.

3.2.4 SGT-Ablauf und SGT-Verhalten

Unter SGT-Ablauf in einem Situationsgraphen versteht man jede zulässige Folge von Situationsschemata dieses Situationsgraphs entlang Prädiktionskanten von einer Startsi-

tuation zu einer Endsituation. (Durch mehrere Start- und Endsituationen sowie Schleifen kann ein Situationsgraph beliebig viele SGT-Abläufe definieren).
Unter SGT-Verhalten versteht man einen vom Wurzelgraph ausgehenden SGT-Ablauf, der auf den gesamten Baum ausgeweitet wurde, indem Situationsschemata, für die Detaillierungen existieren, durch den SGT-Ablauf des Situationsgraphen, zu dem die Detaillierungskante führt, ersetzt werden können. SGT-Verhalten liefern schematische Verhaltensbeschreibungen von Agenten. [Are04].

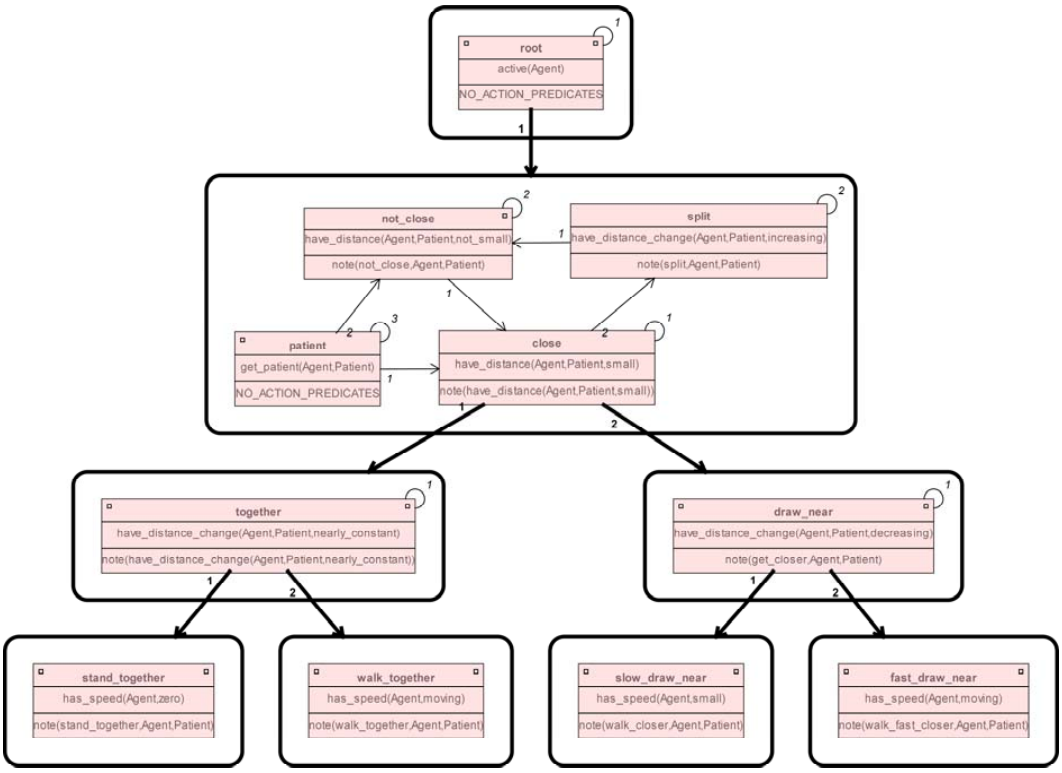

Abbildung 3.5: Situationsgraphenbaum, der zum betrachteten Agenten mögliche Aktionspartner erkennt und anhand des Abstandes zwischen den beiden einstuft. Bei geringem Abstand gibt es Detaillierungen für kleiner werdenden sowie konstant bleibenden Abstand. Diese Detaillierungen werden wiederum durch die Geschwindigkeit, bei der dies geschieht, detailliert.

3.3 Situationsgraphenbaumtraversierung

Die Situationsauswertung geschieht mittels Situationsgraphenbaumtraversierung. Der ursprüngliche Algorithmus hierzu [Are04] lieferte je Zeitpunkt und Objekt, die erstmögliche detaillierteste Situationsausprägung. Wenn verschiedene Situationen den aktuellen

Zustand adäquat beschreiben, oder verschiedene Hypothesen, bedingt durch Unsicherheit der Daten des visuellen Teilsystems, für die aktuelle Situation eines Objektes in Betracht kommen, möchte man mehr als ein SGT-Verhalten in Betracht ziehen. Deshalb erweitert [MJA11] den Algorithmus zu einer unscharfen Situationsgraphenbaumtraversierung, siehe Algorithmus 1.

Algorithm 1: Fuzzy Situation Graph Tree Traversal

```
   Input: SGT, object
1  if object occurs for the first time then
2      G ⟵ SGT root graph;
3      forall the s|s ∈ G ∧ s is start situation do
4          if s can be instantiated then
5              forall the spec|spec ∈ s ∧ spec is specialization do
6                  s := spec, G := graph containing spec;
7                  start recursion goto line 3;
8              evaluate actions of s;
9  else
10     forall the predSit|predSit is prediction situation of s (the last situation of
       the already known object) do
11         if instantiate predSit successful then
12             s := predSit, G := graph containing predSit;
13             start recursion goto line 4;
14         else
15             if predSit is end situation ∧ predSit ∈ G then
16                 instantiation successful;
17             else
18                 instantiation failed;
```

Für jedes neu betrachtete Objekt beginnt die Traversierung im ersten Situationsschema das Wurzelgraphen, das Startsituation ist (Zeile 3f). Falls das Zustandsschema dieses Situationsschemas ausgeprägt werden kann, sucht der Algorithmus nach Detaillierungskanten (Zeile 5). Wurde der Algorithmus fündig und führen gefundene Kanten zu Startsituationen, werden diese, falls möglich, rekursiv ausgeprägt (Zeile 6f). Zu einem Situationsschema dessen Zustandsschema ausgeprägt werden konnte, werden ausgehend von der Wurzel alle Aktionsschemata auf dem Detaillierungspfad zu diesem Situationsschema ausgeführt (Zeile 8). Falls keine Detaillierungskanten von dem Situationsschema ausgehen, oder diese nicht ausgeprägt werden können, wird per Traversierung von Prädiktionskanten mit dem nächsten Zeitschritt fortgefahren (Zeile 10f)

bis eine Endsituation erreicht wird (Zeile 15f). Damit ist die Traversierung dieses Detaillierungsgraph beendet und es geht im allgemeineren Situationsgraphen weiter bis die Endsituation im Wurzelgraph erreicht ist. Falls verschiedene Detaillierungsmöglichkeiten oder zeitliche Abläufe zur Verfügung stehen, werden alle verfolgt, so dass die Traversierung alle möglichen Situationsausprägungen gleichzeitig in Betracht zieht.

3.4 Verknüpfung von Unsicherheit und Vagheit

Bei der videobasierten Situationserkennung hat man es mit zwei Arten von Unschärfe zu tun. Zum einen können durch verrauschtes Bildmaterial oder lückenhaftes Tracking *unsichere* Informationen vorliegen, zum anderen bringt ein begriffliches Ausgabeformat die natürliche Vagheit der menschlichen Sprache mit sich. Die Übergänge von Vagheit und Unsicherheit sind allerdings fließend [Wei96]. Bezeichnet ein Mensch eine Person als 'ziemlich groß' im Wissen der exakten Körpergröße, so ist dies eine vage Information. Diese Aussage kann allerdings bereits versteckt Unsicherheit enthalten, wenn es sich um eine Schätzung handelt, da die exakte Größe gar nicht bekannt ist. Unabhängig davon ist spätestens für einen anderen Menschen, der kein Wissen über den Kontext, wie zum Beispiel Größe und Kulturkreis des Aussagenden [Zim01], besitzt, dieselbe Information eine unsichere Information mangels Hintergrundwissen.

3.4.1 Unsicherheit

Unsicherheit wird von [Wei96] beschrieben als Eingeständnis von fehlendem Wissen. Laut [Zim01] bedeutet Unsicherheit, dass eine Person in einer bestimmten Situation nicht über die nötigen Informationen verfügt, die quantitativ und qualitativ ausreichen, um deterministisch und numerisch ein System, sein Verhalten oder andere seiner Eigenschaften beschreiben, vorhersagen oder vorschreiben zu können. Zimmermann schlägt vor Theorien zu Unsicherheit durch Bestimmen folgender vier Parameter einzuordnen: Ursache der Unsicherheit, Typ der verfügbaren Information (Input), Skalierungsniveau numerischer Information und Typ der benötigten Information (Output).

3.4.2 Ursachen für Unsicherheit

- **Mangel an Information**.
  Quantitativer Informationsmangel: Entscheidungsträger hat keine Information darüber, welcher der möglichen Zustände eintreten wird.
  Qualitativer Informationsmangel: Entscheidungsträger kennt nur die Wahrscheinlichkeiten für das Eintreten diverser Zustände, aber nicht den exakten Zustand.
  Approximation: Erfassen von ausreichender Information für exakte Zustandsbeschreibung wird unterlassen, auch wenn es möglich wäre. Beispiel: Nachkommastellen von π.

- **Überfluss an Information**. Größerer Umfang an Information, als vom Mensch verarbeitet oder kommuniziert werden kann führt zu Skalierung, Vergröberung und Einschränkung auf das Wesentliche.

- **Widersprüchliche Anhaltspunkte**: Zur Verfügung stehende Informationen sprechen für zwei verschiedene Verhalten eines Systems. Zum Beispiel da sich unter ihnen falsche Informationen befinden, die man jedoch nicht lokalisieren kann, oder durch Stützen auf irrelevante Kenndaten des Systems.

- **Mehrdeutigkeit**. Begriffe deren Bedeutung je nach Kontext unterschiedlich sind. Das Unwissen über den nötigen Kontext kann man auch als Mangel an Information einstufen.

- **Messungenauigkeit**. Physikalische Messgeräte können nicht den theoretisch perfekten Wert messen, auch wenn sie immer präziser werden.

- **Überzeugung**. Beobachter erlangt auf der Basis objektiver Daten zu einer Überzeugung, die danach als subjektive Information über das System vorliegt.

3.4.3 Informationstypen

Es ist von Bedeutung, welchen Typ eine unsichere Information hat, da beispielsweise der Unterschied zwischen Agent Nummer 1 und Agent Nummer 2 ein anderer ist, als jener zwischen umgebenden Rechtecken von 70 oder 71 Zentimetern Breite.

- **Numerische Informationen**. Dies können qualitative Werte wie Nominal- oder Ordinalzahlen sein, aber auch metrische Werte der Kardinalskala (Intervall-, Verhältnis- oder Absolutskala).

- **Intervalle**. Scharfe Grenzen und exakte Berechnung durch Intervallarithmetik, auch wenn der exakte Werte selbst nicht bekannt ist.

- **Linguistische Information**. Natürliche Sprache, die beispielsweise von kulturellem Hintergrund oder Bildungsstand abhängt. Hierzu gibt es keine Qualitätskriterien wie zum Beispiel Skalierbarkeit. Man muss zwischen Wörtern als Bezeichnung und ihrer Bedeutung unterscheiden. Oft gibt es weder eine 1:1 Beziehung zwischen diesen beiden, noch eine scharfe kontextunabhängige Definition.

- **Symbole**. Nummern, Buchstaben oder Bilder können offensichtlich für Information in Form von Symbolen verwendet werden. Weniger offensichtlich ist dies bei Worten. Die Nützlichkeit hängt von der jeweiligen Definition der Symbole ab und man benötigt speziell dafür symbolische Informationsverarbeitung.

3.4.4 Vagheit

Bei Vagheit handelt es sich nach [Wei96] um bewussten oder unbewussten Verzicht auf Präzision. Wobei die Unschärfe einer Angabe nicht auf Unwissen beruht. Schäfer erklärt Vagheit in [Sch96] als Bergriffsunschärfe, die durch die subjektive Varianz bei der Interpretation der Begriffe hervorgerufen wird. Zimmermann beschreibt in [Zim01] Vagheit im Gegensatz zu stochastischer Unsicherheit, die den unbekannten zukünftigen Zustand eines wohldefinierten Systems betrifft, als etwas das die Beschreibung der semantischen Bedeutung von Ereignissen, Phänomenen oder Aussagen selbst betrifft. Zimmermann unterscheidet dabei zwischen zwei Arten von Vagheit:
Zum einen intrinsische Vagheit, die durch den Gebrauch menschlicher Sprache entsteht, die viele ungenaue, nicht exakt definierte Ausdrücke enthält, beispielsweise 'große Männer' oder 'hohe Temperatur'. Des Weiteren mag die Bedeutung eines Wortes durchaus wohldefiniert sein, aber bei seiner Benutzung als Bezeichner einer Menge durch den Menschen unscharfe Grenzen existieren. So zum Beispiel bei den Begriffen Fisch oder Vogel. Hat die Vagheit einer Information ihren Ursprung in der menschlichen Sprache, so spricht man von linguistischen Termen ($\widetilde{A}$), Prädikaten (x is $\widetilde{A}$) und Variablen (x) [Wei96].
Unter informationaler Vagheit zum anderen versteht man jene Vagheit, die durch Grenzen der menschlichen Wahrnehmung und Verarbeitung komplexer Informationen entsteht, zum Beispiel 'kreditwürdiger Kunde' dessen komplette Beschreibung möglich, aber mehr als ein Mensch verarbeiten kann, wäre.

3.4.5 Behandlung von Vagheit

In [Wei96] werden unscharfe Mengen als mathematisches Konzept zur Repräsentation vager Informationen vorgestellt. Eine unscharfe Menge $\widetilde{A}$ auf dem Universum $\mathcal{U}$ ist eine Menge, die ihren Elementen eine graduelle Zugehörigkeit, beschrieben durch ihre charakteristische Funktion $\mu_A : \mathcal{U} \to [0,1]$, erlaubt. Die Tilde $\widetilde{}$ kennzeichnet dabei, dass es sich um eine unscharfe Menge handelt, falls dies nicht, wie bei μ_A, bereits aus dem Kontext hervorgeht.
Bei Vereinigung oder Durchschnitt zweier unscharfer Mengen $\widetilde{A}$ und $\widetilde{B}$ mit Zugehörigkeitsfunktionen μ_A und μ_B hängt der Zugehörigkeitsgrad zu den Mengen $\widetilde{A} \cap \widetilde{B}$ und $\widetilde{A} \cup \widetilde{B}$ nur von μ_A und μ_B ab und bei Komplementbildung $\widetilde{A^c}$ nur von μ_A [Wei96]. Für unscharfe Mengenvereinigung, -schnitt und Komplement genügt es also passende Zugehörigkeitsfunktionen anzugeben.

$$\mu_{A\cap B}(u) := t(\mu_A(u), \mu_B(u)) \qquad t : [0,1] \times [0,1] \to [0,1] \tag{3.4}$$

$$\mu_{A\cup B}(u) := s(\mu_A(u), \mu_B(u)) \qquad s : [0,1] \times [0,1] \to [0,1] \tag{3.5}$$

$$\mu_{A^c}(u) := c(\mu_A(u)) \qquad c : [0,1] \to [0,1] \tag{3.6}$$

Eine Funktion $c : [0,1] \to [0,1]$ heißt genau dann unscharfe Komplementfunktion wenn die folgenden Bedingungen $\forall a, b \in [0,1]$ erfüllt sind:

$$\begin{array}{lll} (c1) & c(0) = 1 \wedge c(1) = 0 & (Grenzbedingungen) \\ (c2) & a < b \Rightarrow c(a) \geq c(b) & (Monotonie) \end{array}$$

Während sich für die Funktion c nur das Standardkomplement $c(a) := 1 - a$ durchgesetzt hat, gibt es für s und t eine Vielzahl von Vorschlägen, die lediglich die Bedingung, dass es sich um eine t-Norm und eine s-Norm (auch co-t-Norm genannt) handeln muss, gemeinsam haben und damit die theoretischen Schranken $max\{a, b\} \leq s(a, b)$ und $min\{a, b\} \geq t(a, b)$, da die Maximum-Norm die kleinste aller s-Normen und die Minimum-Norm die größte aller t-Normen darstellt.
Eine Funktion $t : [0,1] \times [0,1] \to [0,1]$ heißt genau dann t-Norm wenn die folgenden Bedingungen $\forall a, b, c \in [0,1]$ erfüllt sind:

$$\begin{array}{lll} (t1) & t(a,1) = a & (Einselement) \\ (t2) & a \leq b \Rightarrow t(a,c) \leq t(b,c) & (Monotonie) \\ (t3) & t(a,b) = t(b,a) & (Kommutativität) \\ (t4) & t(a,t(b,c)) = t(t(a,b),c) & (Assoziativität) \end{array}$$

Eine Funktion $t : [0,1] \times [0,1] \to [0,1]$ heißt genau dann s-Norm (oder auch co-t-Norm) wenn die folgenden Bedingungen $\forall a, b, c \in [0,1]$ erfüllt sind:

$$\begin{array}{lll} (s1) & s(a,0) = a & (Einselement) \\ (s2) & a \leq b \Rightarrow s(a,c) \leq s(b,c) & (Monotonie) \\ (s3) & s(a,b) = s(b,a) & (Kommutativiät) \\ (s4) & s(a,s(b,c)) = s(s(a,b),c) & (Assoziativität) \end{array}$$

[Wei96] rät zur Nutzung von Maximums-Norm, Minimums-Norm und Standardkomplement, wenn kein problemspezifisches Zusatzwissen (über wechselseitige Verstärkung disjunktiver Verknüpfungen oder wechselseitige Abschwächung konjunktiver Verknüpfungen) von globaler Natur die Nutzung anderer Normen rechtfertigt. Diese Normen und das Standardkomplement haben auch den Vorteil, dass sie auf einer Ordinalskala definiert sind, das heißt sie benötigen nicht das Einheitsintervall als Absolutskala und können auch eingesetzt werden, wenn nur relative und keine absoluten Zugehörigkeitsgrade bezüglich einer Grundmenge bekannt sind.
Für vage Informationen erläutert [Wei96] das unscharfe Schließen. Aufbauend auf t-Norm und s-Norm für Schnitt und Vereinigung unscharfer Mengen erhält man für unscharfe Relationen $\widetilde{Q} \in \widetilde{\mathcal{P}}(\mathcal{U} \times \mathcal{V})$ und $\widetilde{R} \in \widetilde{\mathcal{P}}(\mathcal{V} \times \mathcal{W})$ die s-t Komposition $\widetilde{Q} \circ \widetilde{R}$ als unscharfe Relation auf $\mathcal{U} \times \mathcal{W}$ für die gilt:
$\forall (u, v) \in \mathcal{U} \times \mathcal{W} : \mu_{Q \circ R}(u, v) := \underset{v \in \mathcal{V}}{s}(t(\mu_Q(u, v), \mu_R(v, w)))$

Mit Hilfe dieser, meist durch max-min realisierten, s-t Komposition ist Schlussfolgerung per verallgemeinertem Modus Ponens möglich.

$$(x, y)\ is\ \widetilde{\mathcal{R}} \quad (Regel) \tag{3.7}$$

$$x\ is\ \widetilde{A}' \quad (Fakt) \tag{3.8}$$

$$\overline{y\ is\ (\widetilde{A}' \circ \widetilde{\mathcal{R}})} \quad (Schlu\beta) \tag{3.9}$$

3.4.6 Behandlung von Unsicherheit

Als mathematisches Konzept zur Repräsentation unsicherer Information stellt [Wei96] unscharfe Maße vor. Ein unscharfes Maß ordnet sämtlichen Teilmengen des Wertebereiches $\mathcal{U}_x$ der unsicheren Variable x eine Maßzahl aus dem Intervall [0,1] zu, die den Grad der Sicherheit wiederspiegelt, zu dem der Wert von x in der Menge A liegt.
Formal definiert heißt eine Abbildung $U : \mathcal{P}(\mathcal{U}_x) \to [0, 1]$ genau dann unscharfes Maß auf $\mathcal{U}_x$ wenn folgende Bedingungen erfüllt sind:

$$(U1) \quad U(\emptyset) = 0 \wedge U(\mathcal{U}_x) = 1 \qquad (Grenzbedingungen)$$

$$(U2) \quad \forall A, B \subset \mathcal{U}_x : A \subset B \Rightarrow U(A) \leq U(B) \qquad (Monotonie)$$

Zur Konstruktion passender unscharfer Maße für das zu repräsentierende Wissen benötigen wir zunächst eine Maßbasis. Sie gibt an, wie das gesamte Zutrauen, bezüglich irgendwelcher Aussagen über x, auf Teilmengen $A \in \mathcal{U}_x$ verteilt wird. Dabei bezeichnet $m(A)$ den Grad des Zutrauens, dass x exakt in der Menge A und in keiner ihrer echten Teilmengen liegt.
Eine Funktion $m : \mathcal{P}(\mathcal{U}_x) \to [0, 1]$ heißt genau dann Maßbasis zum Universum $\mathcal{U}_x$ wenn folgende Bedingungen erfüllt sind:

$$(m1) \quad m(\emptyset) = 0$$

$$(m2) \quad \sum_{A \subset \mathcal{U}_x} m(A) = 1$$

Zu jeder Maßbasis kann man zwei (im Allgemeinen verschiedene) unscharfe Maße wie folgt konstruieren:

- Man betrachtet die Glaubwürdigkeit $G(A)$ von $x \in A$ indem man über alle Teilmengen $B_i \subseteq A$ das jeweilige Vertrauen in die Aussage 'x liege exakt in B_i und in keiner ihrer echten Teilmengen' aufsummiert. Jedes Zutrauen zu einer Teilmenge von A spricht für $x \in A$. Es werden quasi Zeugen für $x \in A$ gesammelt indem alle Mengen B_i, die '$x \in A$' unterstützen, erhört werden.
  Das Glaubwürdigkeitsmaß zur Maßbasis m G_m wird wie folgt definiert:

$$G_m : \mathcal{P}(\mathcal{U}_x) \to [0, 1]\ mit\ A \mapsto G_m(A) := \sum_{B \subset A} m(B) \tag{3.10}$$

- Oder man betrachtet die Plausibilität $Pl(A)$ von $x \in A$ indem man über alle Mengen B_i, die mit A gemeinsame Elemente besitzen das jeweilige Vertrauen in die Aussage 'x liege exakt in B_i und in keiner ihrer echten Teilmengen' aufsummiert. Hierbei wird gesammelt, was nicht gegen '$x \in A$' spricht.
  Das Plausibilitätsmaß zur Maßbasis m Pl_m wird wie folgt definiert:

$$Pl_m : \mathcal{P}(\mathcal{U}_x) \to [0,1] \; mit \; A \mapsto G_m(A) := \sum_{B:B\cap A\neq\emptyset} m(B) \tag{3.11}$$

Zu einer Maßbasis $m : \mathcal{P}(\mathcal{U}_x) \to [0,1]$ bezeichnet man die Teilmengen $A \subseteq \mathcal{U}_x$ mit $m(a) > 0$ als fokale Elemente. Solch eine Maßbasis m heißt genau dann konsonant, wenn ihre fokalen Elemente $\{A_i \subseteq \mathcal{U}_x | m(A_i) > 0\}$ eine monotone Mengenfamilie $A_1 \supset A_2 \supset \ldots \supset A_n$ bilden. Ein unscharfes Maß, das eine konsonante Maßbasis besitzt, heißt konsonantes unscharfes Maß. Das Gegenteil von konsonant wird mit dissonant bezeichnet.
Ein konsonantes Plausibilitätsmaß heißt Possibilitätsmaß Π und ein konsonantes Glaubwürdigkeitsmaß wird Notwendigkeitsmaß N genannt.
Soll beispielsweise das Alter eines Menschen repräsentiert werden, den man auf zwischen 25 und 30 Jahre schätzt, und man verteilt das Zutrauen auf die Mengen $\{27\}, \{25,\ldots,30\}$ und $\{21,\ldots,40\}$, so handelt es sich um ein konsonantes Maß, da $\{27\} \subset \{25,\ldots,30\} \subset \{21,\ldots,40\}$.
Konsonante Maße eignen sich zur Repräsentation konfliktfreien Wissens, wie man es bei subjektiven Einschätzungen wie der obigen bekommt, dissonante Maße dagegen eignen sich für widersprüchliches Wissen. Bei Inferenzmechanismen für unsichere Information muss zwischen unvollständiger und widersprüchlicher Information unterschieden werden.

3.4.7 Unvollständige Informationen

Für unvollständige Informationen einer zuverlässigen Quelle beschreibt [Wei96] possibilistisches Schließen aufbauend auf Possibilitätsverteilungen (auch Möglichkeitsverteilungen genannt). Bei der possibilistischen Informationsverarbeitung werden alle Werte $u \in \mathcal{U}_x$ als mögliche Werte für x angesehen, die nicht durch die vorhandenen Informationen ausgeschlossen werden können. Die Possibilitätsverteilung der kanonischen Variablen x ist definiert als eine Abbildung $\pi_x : \mathcal{U}_x \to [0,1]$, bei der $\pi_x(u)$ für $u \in \mathcal{U}_x$ den Grad der Möglichkeit angibt, dass $x = u$. Je mehr Information über x zur Verfügung steht, desto spezifischer ist die Possibilitätsverteilung π_x, desto weniger mögliche Werte bleiben für x. Die Menge der potentiellen Lösungen wird mit Informationsgewinn immer kleiner. Exaktes Wissen $x = u'$ bedeutet $\pi_x(u') = 1$ und $\pi_x(u) = 0 \; \forall u \neq u'$. Vollständiges Unwissen dagegen wäre $\pi_x(u) = 1 \; \forall u \in \mathcal{U}_x$.

Sei x eine kanonische Variable auf $\mathcal{U}_x$. Die Algebra $\mathcal{P}_x = (T_x, \Sigma, Q)$ der Possibilitätsverteilungen der unscharfen Variablen x ist wie folgt definiert:

$$
\begin{aligned}
&(1) \quad T_x = \{\pi_x | \pi_x : \mathcal{U}_x \to [0,1]\} \\
&(2) \quad \Sigma = \{\diamond : T_x \times T_x \to T_x\} \\
&(3) \quad Q = \{\pi_x, \pi'_x \in T_x \Rightarrow \forall u \in \mathcal{U}_x : (\pi_x \diamond \pi'_x(u) = min\{\pi_x(u), \pi'_x(u)\}\}
\end{aligned}
$$

Die Kombination $\pi_{x,y,z}$ zweier Possibilitätsverteilungen $\pi_{x,y}$ und $\pi_{y,z}$ auf $\mathcal{U}_x \times \mathcal{U}_y$ bzw. $\mathcal{U}_y \times \mathcal{U}_z$ ist definiert als:

$$
\begin{aligned}
\pi_{x,y,z} \Leftrightarrow \forall (u,v,w) \in \mathcal{U}_x \times \mathcal{U}_y \times \mathcal{U}_z : \pi_{x,y,z}(u,v,w) \\
mit\ \pi_{x,y,z} := min(\pi_{x,y}, \pi_{y,z}) \\
und\ \pi_{x,y,z}(u,v,w) := min\{\pi_{x,y}(u,v), \pi_{y,z}(v,w)\}.
\end{aligned} \tag{3.12}
$$

Zur Possibilitätsverteilung $\pi_{x,y}$ auf $\mathcal{U}_x \times \mathcal{U}_y$ ist die Projektion auf $\mathcal{U}_x$ $[\pi_{x,y} \downarrow \mathcal{U}_x]$ eine Possibilitätsverteilung mit:

$$
\forall u \in \mathcal{U}_x : [\pi_{x,y} \downarrow \mathcal{U}_x](u) := \underset{v \in \mathcal{U}_y}{max}\{\pi_{x,y}(u,v)\}. \tag{3.13}
$$

Komposition erhält man durch Kombination aller zu berücksichtigenden Verteilungen, wie in Definition 3.12 gezeigt, und Projektion des Ergebnisses auf das interessierende Universum nach Definition 3.13.
Die Komposition $\pi_x \circ \pi_{x,y} : \mathcal{U}_y \to [0,1]$ der Possibilitätsverteilungen π_x auf $\mathcal{U}_x$ und $\pi_{x,y}$ auf $\mathcal{U}_x \times \mathcal{U}_y$ ist eine Possibilitätsverteilung auf $\mathcal{U}_y$ mit:

$$
\forall v \in \mathcal{U}_y : \pi_x \circ \pi_{x,y}(v) := \underset{u \in \mathcal{U}_x}{max}\{min\{\pi_x(u), \pi_{x,y}(u,v)\}\}. \tag{3.14}
$$

Linguistische Prädikate können als Possibilitätsverteilungen interpretiert werden, was zu einheitlichen Verarbeitungsmechanismen von vagen und in Form von Unvollständigkeit unsicheren Informationen führt. Die vage Information x is $\widetilde{A}$ kann duch $\pi_x := \mu_A$ in eine unsichere gewandelt werden. Die linguistische Regel IF x is $\widetilde{A}$ THEN y is $\widetilde{B}$ mit $\widetilde{A} \in \widetilde{\mathcal{P}}(\mathcal{U}_x), \widetilde{B} \in \widetilde{\mathcal{P}}(\mathcal{U}_y)$ entspricht dann der Verteilung $\pi_{x,y} : \mathcal{U}_x \times \mathcal{U}_y \to [0,1]$. Mit Verwendung des verallgemeinerten Modus Ponens des unscharfen Schließens, wie in Darstellung 3.9, und der Komposition von Possibilitätsverteilungen nach Definition 3.14 ist possibilistisches Schließen möglich. Als passende Relation hierzu lässt sich die Gödel-Relation herleiten.

$$
\begin{aligned}
\pi_{x,y} \quad & (Regel) && (3.15) \\
\pi'_x \quad & (Fakt) && (3.16) \\
\overline{\pi'_y := \pi'_x \circ \pi_{x,y}} \quad & (Schlu\beta) && (3.17)
\end{aligned}
$$

Die Gödel-Relation μ_G zur Regel $[\widetilde{A} \Rightarrow \widetilde{B}]$ der unscharfen Mengen $\widetilde{A} \in \widetilde{\mathcal{P}}(\mathcal{U}_x)$ und $\widetilde{B} \in \widetilde{\mathcal{P}}(\mathcal{U}_y)$ ist die unscharfe Relation $\widetilde{G} \in \widetilde{\mathcal{P}}(\mathcal{U}_x \times \mathcal{U}_y)$ mit

$$\forall (u,v) \in \mathcal{U}_x \times \mathcal{U}_y : \mu_G(u,v) := \mu_A(u) \propto \mu_B(v) \; mit \; a \propto b := \begin{cases} 1, & \text{falls } a \leq b \\ b & \text{sonst} \end{cases} \tag{3.18}$$

Für eine (possibilistische) Regelbasis muss man noch die Auswertung mehrerer Regeln betrachten: Die Anwendung von n Regeln der Form $[\pi_x^i \Rightarrow \pi_y^i]$ mit $i \in \{1, \ldots, n\}$ auf das Faktum π_x' ist festgelegt mittels:

$$\pi_y' := \pi_x' \circ \min_i(\hat{\pi}_{x,y}^i) \; mit \; \hat{\pi}_{x,y}^i := \pi_x^i \propto \pi_y^i. \tag{3.19}$$

Man kann zeigen, dass sich als gemeinsame Possibilitätsverteilung $\hat{\pi}_{x,y}^*$ zur Repräsentation dieser Regelbasis die größte anbietet, die die Ungleichung $\pi_x^i \circ \hat{\pi}_{x,y}^* \leq \pi_y^i$ erfüllt:

$$\hat{\pi}_{x,y}^*(u,v) := \min_{i \in \{1,\ldots,n\}} \{\pi_x^i(u) \propto \pi_y^i(v)\} \tag{3.20}$$

Zwischen der in diesem Abschnitt vorgeführten Possibilitätsverteilung π_x und dem Possibilitätsmaß Π_x (konsonanten Plausibilitätsmaß nach Definition 3.11) aus dem vorherigen Abschnitt existiert ein bijektiver Zusammenhang.

$$\forall u \in \mathcal{U}_x : \pi_x(u) := \Pi_x(\{u\}) \tag{3.21}$$

$$\forall u \in \mathcal{U}_x : \Pi_x(A) := \max_{u \in A}\{\pi_x(u)\} \tag{3.22}$$

Dies bedeutet, dass man mit dem possibilistischen Schließen ein geeignetes Mittel zur Behandlung unvollständiger Informationen hat.

3.4.8 Widersprüchliche Informationen

Bei unsicheren Informationen, die Widersprüche enthalten können, kann man das possibilistische Schließen nicht anwenden, da dieser Ansatz empfindlich auf inkonsistentes Wissen reagiert. Hierfür stellt [Wei96] das dazu komplementäre Verfahren des evidenzgestützten Schließens vor. Hierbei werden nicht ausschließende Informationen für mögliche Werte gesammelt, sondern solche, die die Annahme von Ereignissen unterstützen, auch wenn es sich dabei um sich widersprechende Informationen handelt.
Die Evidenzverteilung der kanonischen Variable x ist definiert als eine Abbildung $\sigma_x : \mathcal{U}_x \to [0,1]$ bei der $\sigma_x(u)$ für $u \in \mathcal{U}_x$ den Grad angibt, zu dem die Annahme $x = u$ unterstützt wird. Je mehr Information über x zur Verfügung steht, desto kompletter ist die Evidenzverteilung σ_x, desto mehr mögliche Werte gibt es für x. Die Menge der potentiellen Lösungen wird demnach mit Informationsgewinn immer größer. Exaktes Wissen $x = u'$ bedeutet $\sigma_x(u') = 1$ und $\sigma_x(u) = 0 \; \forall u \neq u'$. Vollständiges Unwissen dagegen wäre $\sigma_x(u) = 0 \; \forall u \in \mathcal{U}_x$.

Sei x eine kanonische Variable auf $\mathcal{U}_x$. Die Algebra $\mathcal{S}_x = (T_x, \Sigma, Q)$ der Evidenzverteilungen der unscharfen Variablen x ist wie folgt definiert:

$$
\begin{aligned}
&(1) \quad T_x = \{\sigma_x | \sigma_x : \mathcal{U}_x \to [0,1]\} \\
&(2) \quad \Sigma = \{\diamond : T_x \times T_x \to T_x\} \\
&(3) \quad Q = \{\sigma_x, \sigma'_x \in T_x \Rightarrow \forall u \in \mathcal{U}_x : (\sigma_x \diamond \sigma'_x(u) = max\{\sigma_x(u), \sigma'_x(u)\}\}
\end{aligned}
$$

Die Kombination $\sigma_{x,y,z}$ zweier Evidenzverteilungen $\sigma_{x,y}$ und $\sigma_{y,z}$ auf $\mathcal{U}_x \times \mathcal{U}_y$ bzw. $\mathcal{U}_y \times \mathcal{U}_z$ ist definiert als:

$$
\begin{aligned}
\sigma_{x,y,z} \Leftrightarrow \forall (u,v,w) \in \mathcal{U}_x \times \mathcal{U}_y \times \mathcal{U}_z : \sigma_{x,y,z}(u,v,w) \\
mit\ \sigma_{x,y,z} := min(\sigma_{x,y}, \sigma_{y,z}) \\
und\ \sigma_{x,y,z}(u,v,w) := min\{\sigma_{x,y}(u,v), \sigma_{y,z}(v,w)\}.
\end{aligned} \tag{3.23}
$$

Die Projektion einer Evidenzverteilung $[\sigma_{x,y} \downarrow \mathcal{U}_x](u)$ ist im Allgemeinen gleich Null. Deshalb spielt diese kaum eine Rolle. Die Komposition erhält man daher hier nicht auf gleichem Weg, wie bei Possibilitätsverteilungen. Man betrachtet hier zu einem festen u den Grad der Unterstützung $\sigma_x(u)$ von $x = u$ und den Grad $\sigma_{x,y}(u,v)$ zu dem das gemeinsame Auftreten von $x = u$ und $y = v$ unterstützt wird. Sei dazu τ_u der abgeleitete Grad zu dem $y = v$ unterstützt wird. τ_u hängt von der Wechselwirkung zwischen $\sigma_x(u)$ und $\sigma_{x,y}(u,v)$ ab. Daraus folgt $\tau_u \geq \sigma_x(u)$ oder $\tau_u \geq \sigma_{x,y}(u,v)$. Mit dem Prinzip minimaler Komplettierung ergibt sich: $\tau_u = min\{\sigma_x(u), \sigma_{x,y}(u,v)\}$. Betrachtet man nun alle $u \in \mathcal{U}_x$ und bestimmt die zugehörigen τ_u, so bekommt man: $\sigma_y(v) := \underset{u \in \mathcal{U}_x}{max}\{\tau_u\}$.
Trotz unterschiedlicher Herleitung bekommt man das selbe Ergebnis wie bei Possibilitätsverteilungen:
Die Komposition $\sigma_x \circ \sigma_{x,y} : \mathcal{U}_y \to [0,1]$ der Evidenzverteilungen σ_x auf $\mathcal{U}_x$ und $\sigma_{x,y}$ auf $\mathcal{U}_x \times \mathcal{U}_y$ ist eine Evidenzverteilung auf $\mathcal{U}_y$ mit:

$$
\forall v \in \mathcal{U}_y : \sigma_x \circ \sigma_{x,y}(v) := \underset{u \in \mathcal{U}_x}{max}\{min\{\sigma_x(u), \sigma_{x,y}(u,v)\}\}. \tag{3.24}
$$

Linguistische Prädikate können ebenfalls als Evidenzverteilungen interpretiert werden, was zu einheitlichen Verarbeitungsmechanismen von vagen und widersprüchlichen Informationen führt. Die vage Information x is $\widetilde{A}$ kann duch $\sigma_x := \mu_A$ in eine unsichere gewandelt werden. Die linguistische Regel IF x is $\widetilde{A}$ THEN y is $\widetilde{B}$ mit $\widetilde{A} \in \widetilde{\mathcal{P}}(\mathcal{U}_x), \widetilde{B} \in \widetilde{\mathcal{P}}(\mathcal{U}_y)$ entspricht dann der Verteilung $\sigma_{x,y} : \mathcal{U}_x \times \mathcal{U}_y \to [0,1]$.
Auch hier ist mit Verwendung des verallgemeinerten Modus Ponens des unscharfen Schließens, wie in Darstellung 3.9, und der Komposition von Evidenzverteilungen nach Definition 3.24 possibilistisches Schließen möglich.
Die zur Gödel-Relation duale Relation $\sigma_{x,y*}(u,v) := \begin{cases} 0, & \text{falls } \sigma_x(u) \leq \sigma_y(v) \\ \sigma_y(v) & \text{sonst} \end{cases}$
ist dabei nicht die perfekte Wahl, denn die MAMDANI-Relation $\breve{\sigma}_{x,y}$ nach Definition 3.28 ist eine im Allgemeinen echt komplettere Implikationsrelation.

$$\breve{\sigma}_{x,y} \quad (Regel) \tag{3.25}$$
$$\sigma'_x \quad (Fakt) \tag{3.26}$$
$$\overline{\sigma'_y := \sigma'_x \circ \breve{\sigma}_{x,y}} \quad (Schlu\beta) \tag{3.27}$$

Die MAMDANI-Relation μ_R zur Regel $[\widetilde{A} \Rightarrow \widetilde{B}]$ der unscharfen Mengen $\widetilde{A} \in \widetilde{\mathcal{P}}(\mathcal{U}_x)$ und $\widetilde{B} \in \widetilde{\mathcal{P}}(\mathcal{U}_y)$ ist die unscharfe Relation $\widetilde{R} \in \widetilde{\mathcal{P}}(\mathcal{U}_x \times \mathcal{U}_y)$ mit:

$$\forall (u,v) \in \mathcal{U}_x \times \mathcal{U}_y : \mu_R(u,v) := min\{\mu_A(u), \mu_B(v)\} \tag{3.28}$$

Die Auswertung von $n > 1$ Regeln $[\sigma_x^i \Rightarrow \sigma_y^i]$ erhält man durch Auswerten des Maximums über alle i Regeln (vergleiche Darstellung 3.27) unter Verwendung der Mamdani-Relation nach Definition 3.28 und das ganze komponentenweise mit Hilfe der Komposition von Evidenzverteilungen, wie in Definition 3.24:

$$\sigma'_y = \sigma'_x \circ \underset{i}{max}(min(\sigma_x^i, \sigma_y^i)) \tag{3.29}$$
$$\forall v \in \mathcal{U}_y : \sigma_y(v) = \underset{u \in \mathcal{U}_x}{max}\{min\{\sigma'_x(u), \underset{i}{max}\{min\{\sigma_x^i(u,v), \sigma_y^i(u,v)\}\}\}\} \tag{3.30}$$

Zur Evidenzverteilung gibt [Wei96] eine korrespondierende Maßbasis an:

$$\forall A \subset \mathcal{U}_x : m(A) := \begin{cases} \frac{H(\sigma_x)}{\sum_{u \in \mathcal{U}_x} \sigma_x(u)} * \sigma_x(u), & \text{falls } A = \{u\},\ u \in \mathcal{U}_x, \\ 1 - H(\sigma_x), & \text{falls } A = \mathcal{U}_x, \\ 0, & \text{sonst.} \end{cases} \tag{3.31}$$

$$\text{Wobei die sogenannte Höhe definiert ist mit: } H(\sigma_x) := \underset{u \in \mathcal{U}_x}{max}\{\sigma_x(u)\}. \tag{3.32}$$

Bei der in Definition 3.31 vorgestellten Maßbasis handelt es sich um eine dissonante Maßbasis, deren Wertebereich der zugehörigen unscharfen Variablen selbst fokales Element sein kann.

3.4.9 Gleichzeitige Behandlung von Unsicherheit und Vagheit

Gleichzeitig vage und unsichere Information kann also je nach Art der Unsicherheit mittels possibilistischen Ansatzes, wenn man es mit Unvollständigkeit zu tun hat, oder evidenzgestütztem Ansatz, wenn man es mit Inkonsistenz zu tun hat, behandelt werden. In der Praxis lassen sich Informationsquellen meist nicht eindeutig der einen oder anderen Art zuordnen. Zugleich sind beide Ansätze sehr empfindlich gegenüber der jeweils anderen Art der Unsicherheit.
Dies kann man gut anhand des von [Wei96] gegebenen Beispiels sehen:
Er betrachtet zwei Possibilitätsverteilungen π_x^1 und π_x^2, die inkonsistente Informationen repräsentieren:

- π_x^1 besagt beispielsweise, dass x nicht größer als u' sein kann:
  $\forall u \in \mathcal{U}_x : u > u' \Rightarrow \pi_u^1 = 0$
- π_x^2 schließt beispielsweise für x alle Werte kleiner als u' völlig aus:
  $\forall u \in \mathcal{U}_x : u \leq u' \Rightarrow \pi_u^2 = 0$

Kombiniert man diese beiden Wissensquellen, so erhält man:
$\pi_x = \pi_x^1 \diamond \pi_x^2 = min(\pi_x^1, \pi_x^2) \equiv 0$.
Die Annahme $x = u$ wird also für alle Werte $u \in \mathcal{U}_x$ als vollkommen unmöglich betrachtet. Die so abgeleitete Possibilitätsverteilung π_x lässt auch keine sinnvollen Schlussfolgerungen mehr zu, denn mit $\pi_x \equiv 0$ wird aus $\pi_y := \pi_x \circ \pi_{x,y}$ (siehe Darstellung 3.17) bei komponentenweiser Auswertung der Komposition (siehe Definition 3.14): $\pi_y(v) = \underset{u \in \mathcal{U}_x}{max}\{min\{0, \pi_{x,y}(u,v)\}\} = 0$.
Man kann allerdings den Grad der Tauglichkeit des gewählten Ansatzes bestimmen. Der Grad ϵ der Konsistenz einer Possibilitätsverteilung ist gerade ihre 'Höhe':
$\epsilon := H(\pi_x) = \underset{u \in \mathcal{U}_x}{max}\{\pi_x(u)\}$.
Eine normalisierte Possibilitätsverteilung hat Konsistenzgrad 1. Die Possibilitätsverteilung $\pi_x \equiv 0$ aus obigem Beispiel hat Konsistenzgrad 0.
Zu einer possibilistischen Regelbasis $[\pi_x^i \Rightarrow \pi_y^i]$, die sich, wie in Formel 3.20 gezeigt, durch die gemeinsame Possibiliätsverteilung $\hat{\pi}_{x,y}^*$ repräsentieren lässt, kann man den ϵ-Konsistenzgrad bestimmen mittels:

$$\epsilon := \underset{u \in \mathcal{U}_x}{min}\{[\hat{\pi}_{x,y}^* \downarrow \mathcal{U}_x](u)\} = \underset{u \in \mathcal{U}_x}{min}\{\underset{v \in \mathcal{U}_y}{max}\{\hat{\pi_{x,y}}^*(u,v)\}\}. \tag{3.33}$$

Analog dazu kann man für Evidenzverteilungen den ϵ-Vollständigkeitsgrad angeben, der wiederum der 'Höhe' wie in Formel 3.32 entspricht: $\epsilon := H(\sigma_x)$
Der ϵ-Vollständigkeitsgrad einer evidenzgestützten Regelbasis $[\sigma_x^i \Rightarrow \sigma_y^i]$ mit $\check{\sigma}_{x,y}^*$ ergibt sich als:

$$\epsilon := \underset{u \in \mathcal{U}_x}{min}\{\underset{v \in \mathcal{U}_y}{max}\{\check{\sigma}_{x,y}^*(u,v)\}\} \tag{3.34}$$

$$mit\ \check{\sigma}_{x,y}^*(u,v) := \underset{i \in \{1,\ldots,n\}}{max}\{min\{\sigma_x^i(u), \sigma_y^i(v)\}\} \tag{3.35}$$

Hat man eine zuverlässige aber eventuell nicht vollständige (nicht mitteilsame) Informationsquelle, eignet sich der possibilistische Ansatz. Für eine mitteilsame, aber eventuell nicht zuverlässige Informationsquelle der evidenzgestützte Ansatz. Mit Hilfe von ϵ-Konsistenz und ϵ-Vollständigkeit, kann man testen in wieweit sich eine possibilistische oder evidentzestütze Regelbasis, die auf den vorhandenen Informationen aufbaut, für Schlussfolgerungen eignet.
Eine andere Methode liefert die Dempster-Shafer-Theorie [Dem68, Sha76]. Sie umgeht das Problem der beiden sich ausschließenden Theorien, indem statt eines Zusicherungsgrades jeweils zwei Werte, einer für Überzeugung und einer für Plausibilität betrachtet werden. Die Arbeit [MYC+10] wendet die Dempster-Shafer-Theorie auf die Situationserkennung an.

Kapitel 4

Entwicklung einer Regelbibliothek für den Diskursbereich Parkplatz und Personenüberwachung

Die Begriffliche-Primitive-Ebene (BPE), siehe Abbildung 3.1, beinhaltet eine Regelbibliothek, die das Zwischenstück zwischen visuellem Teilsystem (VS) und Verhaltens-Repräsentations-Ebene (VRE) bildet. Einfache Zusammenhänge zwischen geometrischen Lagedaten wie Koordinaten, Breite und Höhe objektumgebender Rechtecke und Informationen wie Objektklasse, Abstand, Abstandsänderung, Geschwindigkeit oder Richtung werden als FMTHL-Regeln formuliert. Diese aufbereiteten Informationen stehen dann im SGT bei der Formulierung des Verhaltenswissens als logische Prädikate zur Verfügung.
Als FMTHL-Fakten in F-LIMETTE importierte geometrische Szenenbeschreibungen aus dem visuellen Teilsystem (VS) sehen beispielsweise so aus:

```
5151 ! (has_status(obj_3,855,1240,391,57,64,1)).
5151 ! (has_status(obj_4,2206,704,657,66,153,1)).
5152 ! (has_status(obj_3,855,1240,390,58,65,1)).
5152 ! (has_status(obj_4,2206,705,657,66,153,1)).
5153 ! (has_status(obj_3,855,1240,389,58,66,1)).
5153 ! (has_status(obj_4,2206,705,657,66,153,1)).
```

Hier hat man für zwei Objekte, die mit obj_3 und obj_4 bezeichnet sind, zu den Einzelbildnummern 5151 bis 5153 die Verweildauer des Objektes (855 bzw. 2206 Einzelbilder), die Lage eines umgebenden Rechtecks in Form von x- und y-Koordinate, Breite und Höhe, sowie eine 1, die anzeigt, dass es sich um eine Person handelt. Regeln, die aus diesen Daten bestimmen, ob ein Objekt eine Person ist oder den Abstand zweier Objekte bestimmen, sehen beispielsweise so aus:

```
always (is_person(Agent) :-
        has_status(Agent,_,_,_,_,_,1)
).

always (has_geometry(Agent,X,Y,W,H) :-
        has_status(Agent,_,Xp,Yp,W,H,_)
        , projectiveTransform(Xp,Yp,X,Y)
).

always (distance_is(Agent,Patient,Distance) :-
        has_geometry(Agent,X,Y,_,_)
        , has_geometry(Patient,X0,Y0,_,_)
        , length(X-X0,Y-Y0,Distance)
).
```

Weiteres Verhaltenswissen, das in Situationsgraphenbäumen festgehalten wird, wird nach Übersetzung in F-LIMETTE-Fakten und -Regeln per Traversierung des Graphen genutzt. Die nötigen Anweisungen zur Traversierung liegen getrennt vom Situationsgraphenbaum ebenfalls als FMTHL-Regeln für F-LIMETTE vor.

4.1 Entwicklung von Modultests für die Regelbibliothek

Viele Fehler in F-LIMETTE werden zur Compilezeit noch nicht erkannt. Andererseits ist es sehr leicht sich, beispielsweise durch Schreibfehler, eine fehlerhafte Regelbasis einzuhandeln, was sowohl zu syntaktischen, als auch zu semantischen (unbemerkten) Fehlern führen kann:

```
always(quadrat1(A, Q) :- Q is A * A).
always(quadrat2(A, Q) :- Q is A * Aa).
always(quadrat3(A, Q) :- Q is A + A).
```

In obigem Beispiel ist nur die erste Zeile korrekt. Die Anfrage ‘quadrat2(3, Q).’ führt zum Fehler ‘variable _5 in arithmetic term’ und die Anfrage ‘quadrat3(3, Q).’ liefert zwar fehlerfrei ein Ergebnis, aber nicht das erwartete. Nur eine Anfrage der Form ‘quadrat1(Q).’ in der Regelbasis würde direkt beim Zufügen der Regelbasis einen Fehler (predicate quadrat1 / 1 is undefined.) melden. Es war schnell klar, dass ein automatischer Test der F-LIMETTE-Dateien sehr sinnvoll ist, nicht nur um Fehler aufzudecken und sicherzugehen keine neuen Fehler bei Änderungen einzubringen, sondern auch um Beispielanfragen zu allen Regeln zu haben. Um mehr Erfahrung mit F-LIMETTE zu

sammeln werden die Tests selbst in F-LIMETTE realisiert, statt ein externes Programm zu schreiben, das Anfragen an F-LIMETTE schickt und die Antworten verifiziert. Das Prolog-Paket 'Prolog Unit Tests' von Jan Wielemaker [WSTL10] konnte weder direkt noch mit überschaubaren Anpassungen für F-LIMETTE verwendet werden, diente allerdings als Inspiration für eine deutlich einfacher gehaltene Variante von Tests für F-LIMETTE. Der Test nutzt einerseits aus, dass in F-LIMETTE bei einer Und-Verknüpfung von Termen termweise ausgewertet wird solange die einzelnen Terme zu 'true' auswerten. Andererseits werden alle passenden Regeln durchprobiert, solange noch keine ein 'true' ergab, aber noch Möglichkeiten dazu vorhanden sind.

```
always(test(quadrat1) :-
       writeln('starte test von quadrat1')
       , quadrat1(3, 9)
       , writeln('test von quadrat1 bestanden')
       , fail
).
```

Obige Regel verdeutlich das Funktionsprinzip der Tests, die durch Hilfsfunktionen zur Vereinheitlichung und Aufwandsersparnis geringfügig anders aussehen. Es kommt nur zur Ausgabe von 'test von quadrat1 bestanden' wenn sowohl writeln('starte test von quadrat1') als auch quadrat1(3, 9) zu 'true' ausgewertet werden konnten, was man von ersterem als gegeben voraussetzen kann. Das 'fail' am Ende sorgt dafür, dass eine Anfrage 'test(quadrat1).' alle Tests mit obigem Aufbau durchgeht, statt nach dem ersten abzubrechen und mittels 'test(X).' können alle vorhandenen Tests für beliebige Regeln durchgegangen werden.

4.2 Aufteilung und Anpassung der Regelbibliothek

Da die vom visuellen Teilsystem gelieferten Zustands-Vektoren je Agent und Zeitpunkt in den enthaltenen Parametern variieren, musste jeweils die Regelbasis komplett angepasst werden. Auch Änderungen im Zeitabstand der Einzelbilder oder der Skalierung der Koordinaten zogen jeweils nötige Anpassungen mit sich. Ein möglichst klein gehaltener Satz von F-LIMETTE-Regeln, die in solchen Fällen anzupassen sind, wurde in eine Interface-Datei ausgelagert und die Regelbasis so umgestellt, dass indirekt über dieses Interface auf Fakten zugegriffen wird. Zum Beispiel wird nun in der Regelbasis das im Interface als Regel definierte $has\_image\_geometry(Agent, X, Y)$. verwendet statt $has\_status(Agent, ...X, Y, ...)$. mit X und Y an jeweils passender Stelle. Siehe auch Kapitel 6.3. Die Regelbasis orientiert sich an jener in der Arbeit [GN08] musste allerdings an den neuen Diskursbereich angepasst und grundlegend erweitert werden. Bei Personen gibt es beispielsweise kein geschwindigkeitsabhängiges Abstandsempfinden, wie bei Fahrzeugen untereinander und das Richtungsempfinden ist nicht durch Fahrspuren beeinflusst, so dass schräg rechts sinnvoll sein kann, wo zuvor eher die Zu-

ordnung der selben oder benachbarten Fahrspur interessierte. Einige Regeln sind bedingt durch den neuen Diskursbereich, aber auch das Interesse an anderen Situationen hinzugekommen, wie beispielsweise das Auftauchen oder Verschwinden von Objekten:

```
always( has_data(Agent) :-
        has_image_geometry(Agent, _, _, _, _)
).

always( appear(Agent) :-
          not (-5 ! has_data(Agent))
        , not (-4 ! has_data(Agent))
        , 0 ! has_image_geometry(Agent, _, _, _, _)
        , 1 ! has_image_geometry(Agent, _, _, _, _)
).

always( disappear(Agent) :-
         -1 ! has_image_geometry(Agent, _, _, _, _)
        , 0 ! has_image_geometry(Agent, _, _, _, _)
        , not (4 ! has_data(Agent))
        , not (5 ! has_data(Agent))
).
```

Im Zustandsschema des SGTs kann man mittels SGT-Editor mehrere Bedingungen angeben, die UND-verknüpft ausgewertet werden. Die Eingabe einer ODER-Verknüpfung ist weder als getrennte Eingabe von Bedingungen vorgesehen, noch kann ein ODER-Verknüpfter Term in Form von ‘A ; B’ als eine Bedingung eingegeben werden. Daher wurde eine Regel für ein ODER in der strong Variante (siehe 3.2) zugefügt:

```
always( or(A, B) :- A ;@S B ).
```

Für die relative Lage zweier Personen stellte sich heraus, dass der Ursprung des umgebenden Rechtecks als Referenzpunkt für Vergleiche ungünstig ist. Bedingt durch die Projektion der dreidimensionalen Szene auf zweidimensionale Bildpunkte wird die Lage zweier Objekte zueinander durch ihre unterschiedliche Höhe verfälscht. Siehe Abbildung 4.1 links. Da Personen verglichen mit ihrer Höhe eine kleine Grundfläche besitzen und da die genutzte projektive Transformation Bildpunkte auf Koordinaten der Bodenfläche in der Szene abbildet, bietet es sich an den Referenzpunkt ebenfalls in der Bodenfläche anzusiedeln und zwar in der Mitte der unteren Kante des umgebenden Rechtecks. Hierzu wurde folgende Regel ergänzt:

```
always( has_ground_geometry(Agent, Xg, Yg) :-
        has_image_geometry(Agent, Xi, Yi, W, H)
        , X is Xi + W/2
        , Y is Yi + H
        , projectiveTransform(X, Y, Xg, Yg)
).
```

Dieser Ansatz ist allerdings nicht mehr sinnvoll, wenn man es mit Objekten zu tun hat, deren Grundfläche verglichen mit ihrer Höhe groß ist, wie es bei Fahrzeugen der Fall ist. In der mittleren Abbildung von 4.1 kann man das entstehende Problem gut sehen: Die Person würde als 2,67 Meter vom Fahrzeug entfernt betrachtet werden, obwohl sie direkt daneben steht. Für den Lagevergleich von Personen und solchen großflächigeren Objekten wie Fahrzeugen wurden passende Regeln ergänzt. Zum einen eine Regel dafür inwieweit ein umgebendes Rechteck in einem anderen enthalten ist. Der berechnete Anteil der Überlappung wird mittels $sp(DoV)$ als Zusicherungsgrad (Degree of Validity) gesetzt:

```
always( box_in_box(X1, Y1, W1, H1, X2, Y2, W2, H2, DoV) :-
          X1_ is X1 + W1
        , Y1_ is Y1 + H1
        , X2_ is X2 + W2
        , Y2_ is Y2 + H2
        , max(X1, X2, XL)
        , min(X1_, X2_, XR)
        , W_ is XR - XL
        , W_ > 0
        , max(Y1, Y2, YO)
        , min(Y1_, Y2_, YU)
        , H_ is YU - YO
        , H_ > 0
        , N is 1.0 * W_ * H_
        , Z is 1.0 * W1 * H1
        , DoV is N/Z
).
always( box_in_box(Tiny, Huge) :-
        has_image_geometry(Tiny, X1, Y1, W1, H1)
        , has_image_geometry(Huge, X2, Y2, W2, H2)
        , box_in_box(X1, Y1, W1, H1, X2, Y2, W2, H2, DoV)
        , sp(DoV)
).
```

Da man hier nur Bildpunkte betrachtet und es sich um prozentuale Angaben handelt spart man im Gegensatz zu absoluten Abstandsbestimmungen die Überlegung welchen Skalierungsfaktor die projektive Transformation mit sich bringt. Für die vage Aussage, dass sich eine Person in der Nähe eines Fahrzeuges befindet, genügt diese Regel abgesehen von wenigen Ausnahmen: Hat man ein Fahrzeug, das quer zum Bild steht und eine Person am Kofferraum, so befinden sich die beiden umgebenden Rechtecke vermutlich überschneidungsfrei genau nebeneinander. Siehe Abbildung 4.1 rechts. Ähnliches kann bei Personen die vom Fahrzeug selbst verdeckt werden passieren: die beiden Rechtecke liegen überschneidungsfrei übereinander. Da, wie bereits erwähnt, absolute Abstandsangaben durch die Projektion schwierig zu handhaben sind, wurde auch hierfür eine

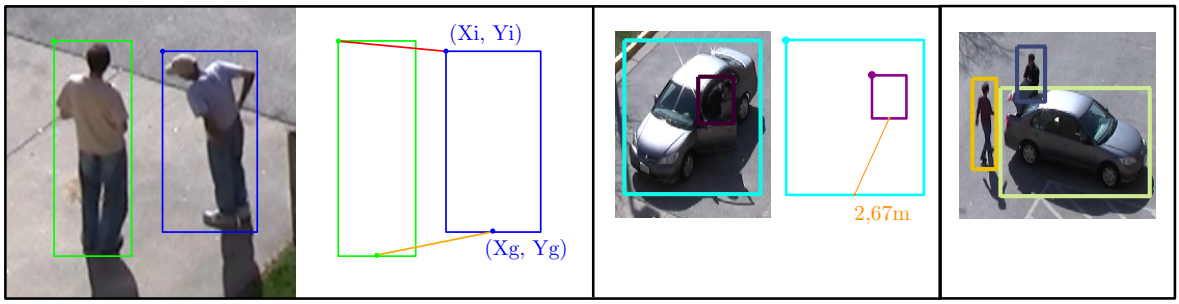

Abbildung 4.1: Links: Mögliche Referenzpunkte der Objekte bei der gegenseitigen Lagebestimmung: Der Ursprung des umgebenden Rechtecks (Xi, Yi) in der linken oberen Ecke, der sich bedingt durch die Projektion insbesondere bei der Richtungsbestimmung als ungünstig erwiesen hat, und der Mittelpunkt der unteren Kante (Xg, Yg) als Alternative. Mitte: Betrachtung der Mitte der unteren Rechteckskante als Referenzpunkt führt bei großflächigen Objekten mit geringer Höhe zu Problemen: Der aussteigende Fahrer hätte bei dieser Betrachtung einen Abstand von 2,67m zum Fahrzeug. Rechts: Bei quer stehenden Fahrzeugen kann es vorkommen, dass es mit dem umgebenden Rechteck einer Person direkt neben dem Fahrzeug zu keiner Überschneidung kommt.

Lösung gefunden, die mit relativen Werten auskommt: Hierzu wird das umgebende Rechteck der Person passend zu den Problemfällen vergrößert indem an beiden Seiten die bisherige Breite noch einmal angefügt wird und an der unteren Kante die halbe Höhe. Siehe Abbildung 4.3. Da für überscheidungsfrei direkt nebeneinanderliegende Rechtecke die Überschneidung des vergrößerten Rechtecks mit dem großen maximal ein Drittel betragen kann, wird der Grad der Überschneidung für den Zusicherungsgrad dreifach genommen.

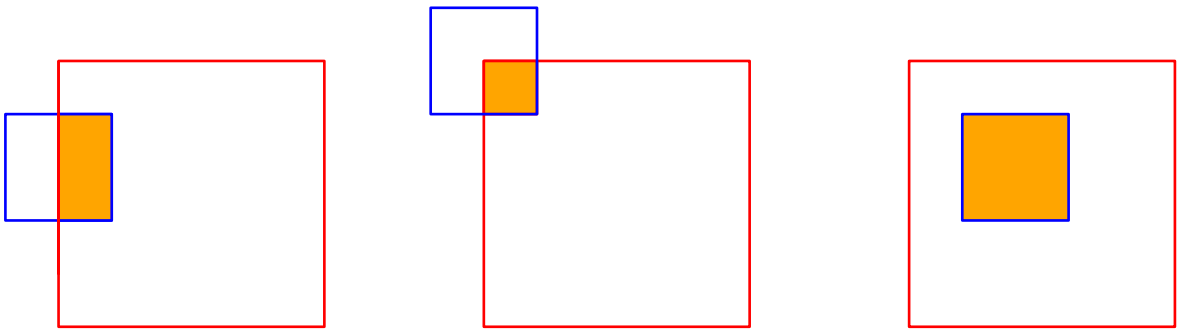

Abbildung 4.2: Für die Regel zur Bestimmung, ob eine Person in der Nähe eines Fahrzeuges ist, wird bestimmt welcher Anteil der Fläche ihres umgebenden Rechtecks im großen Rechteck des Fahrzeuges liegt (Links 0.5, Mitte 0.25, Rechts 1.0).

```
always( box_near_box(Tiny, Huge) :-
        has_image_geometry(Tiny, X1, Y1, W1, H1)
        , has_image_geometry(Huge, X2, Y2, W2, H2)
        , X1e is X1 - W1
        , Y1e is Y1
        , W1e is 3 * W1
        , H1e is 1.5 * H1
        , box_in_box(X1e, Y1e, W1e, H1e, X2, Y2, W2, H2, R_)
        , Rn is R_ * 3
        , min(R,1,D)
        , DoV is 1.0 * D
        , sp(DoV)
).
```

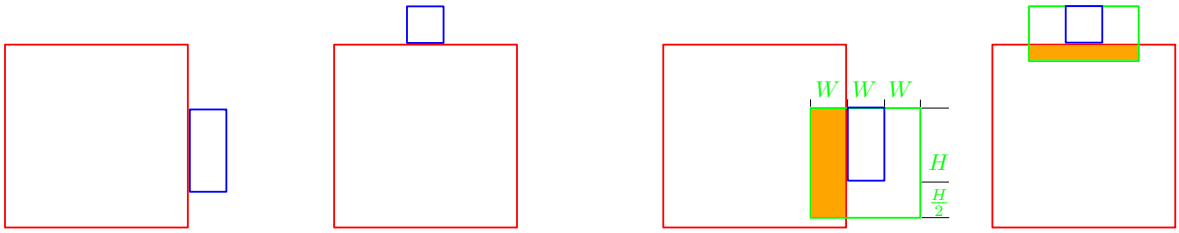

Abbildung 4.3: Linke Hälfte: Fälle in denen die Betrachtung der Überschneidung der Rechtecke nicht praktikabel ist, beispielsweise entstanden durch Person am Kofferraum eines quer zum Bild stehenden Fahrzeuges und vom Fahrzeug teilweise überdeckte Person. Für diese Fälle wird das umgebende Rechteck der Person passend vergrößert und bestimmt welcher Anteil der Fläche des vergrößerten Rechtecks im großen Rechteck des Fahrzeuges liegt.

Die komplette Regelbibliothek inklusive der darin enthaltenen Erläuterungen und Erklärungen ist im Anhang B.2 zu finden. Die Datei mit dem zugehörigen Modultest wurde unter B.3 ebenfalls angefügt.

Kapitel 5

Umsetzung im SGT-Editor und im Traversierungsalgorithmus

Der SGT-Editor soll so erweitert werden, dass er mit unvollständigen Daten zurechtkommt. Hierzu werden zwei Arten von Unvollständigkeit getrennt voneinander betrachtet: Zuerst der Datenausfall für ein Objekt zu einem Zeitpunkt, bedingt durch eher technische Probleme im visuellen Teilsystem und zum Zweiten Unwissen über ein Objekt über mehrere Zeitpunkte hinweg, wozu es beispielsweise durch Verdeckungen kommen kann.

5.1 Unvollständige Daten durch Ausfall zu einem Zeitpunkt

Für temporale Schlussfolgerungen zu einem Zeitpunkt t in F-LIMETTE müssen alle Fakten zu einem Zeitintervall $[t-\text{MIN\_INTERVAL}+1, t+\text{MIN\_INTERVAL}-1]$ um diesen Zeitpunkt t zur Verfügung stehen. Die Größe dieses Intervalls hängt davon ab, welche Zeitpunkte in temporalen Operatoren wie beispielsweise 'drei Zeitpunkte zuvor' (-3 ! has_ground_geometry(Agent, X_3, Y_3).) verwendet werden.
Bei F-LIMETTE Regeln, die Fakten zu mehreren Zeitpunkten benötigen, wie beispielsweise in Abbildung 5.1 für den betrachteten Zeitpunkt sowie ein, zwei bzw. drei Zeitpunkte zuvor und danach, ist eine Auswertung zu 'true' nur möglich, wenn diese benötigten Fakten für den betrachteten Agenten lückenlos vorliegen. Bei einer Auswertung zum Zeitpunkt 15 werden beispielsweise die Fakten für das Zeitintervall $[12, 18]$ benötigt.
Hat man nun in den Daten, die das visuelle Teilsystem liefert, eine Lücke, wie in Abbildung 5.2, so liefert die Auswertung des Prädikates 'has_ground_geometry(Agent, X, Y)' zum Zeitpunkt 16 ein 'false' und damit auch die Auswertung des Prädikates 'move_direction_is(Agent,Dir)' zu den Zeitpunkten 13 bis 19 ebenfalls nur 'false'. Um trotzdem eine Auswertung zu allen Zeitpunkten zu ermöglichen, müssen solche Lücken in den Daten vor der logischen Auswertung geschlossen werden.

```
always( move_direction_is(Agent,Dir) :-
          -3 ! has_ground_geometry(Agent,X_3,Y_3)
        , -2 ! has_ground_geometry(Agent,X_2,Y_2)
        , -1 ! has_ground_geometry(Agent,X_1,Y_1)
        ,  1 ! has_ground_geometry(Agent,X1,Y1)
        ,  2 ! has_ground_geometry(Agent,X2,Y2)
        ,  3 ! has_ground_geometry(Agent,X3,Y3)
        , Y is Y3 + Y2 + Y1 - Y_1 - Y_2 - Y_3
        , X is X3 + X2 + X1 - X_1 - X_2 - X_3
        , atan2(Y, X, Rad)
        , convert_angle(Rad, Dir)
).
```

Abbildung 5.1: F-LIMETTE Regel, die Fakten zu mehreren Zeitpunkten benötigt. Durch die UND-verknüpften Terme, die das Prädikat 'has_ground_geometry' für den betrachteten Zeitpunkt sowie ein, zwei bzw. drei Zeitpunkte zuvor und danach enthalten, kann die Regel nur ausgewertet werden, wenn die nötigen Fakten, d.h. 'has_status', für die Auswertung der Prädikate 'has_ground_geometry' für all diese Zeitpunkte vorhanden sind.

5.1.1 Angepasste Bestimmung der für den Zeitpunkt zu betrachtenden Agenten

Um die Traversierung für neu hinzugekommene Agenten anzustoßen oder für bereits vorhandene Agenten fortzuführen werden im SGT-Editor die neuen und alten Agenten in jedem Traversierungsschritt bestimmt. Damit der Umgang mit Lücken ermöglicht wird, muss bei der Bestimmung der zu einem Zeitpunkt t vorhandenen und neu hinzugekommenen Agenten beachtet werden, dass zu diesem oder dem vorherigen Zeitpunkt t_{-1} in der Zeitleiste eine Lücke für einen in Frage kommenden Agenten vorliegen kann.
Für neue Agenten galt bisher die Bedingung:
Zum Zeitpunkt t_{-1} nicht vorhanden und zum Zeitpunkt t vorhanden.
Die angepasste Bedingung für neue Agenten lautet:
Im Intervall I_1 der Länge SUBST_INTERVAL vor dem Zeitpunkt t_{-1} nicht vorhanden und im Zeitpunkt t vorhanden,
mit $I_1 := [t_{-1}-\text{SUBST\_INTERVAL}, t_{-1}]$.
Für aktuelle Agenten galt bisher die Bedingung:
Zu beiden Zeitpunkten t_{-1} und t vorhanden.
Die angepasste Bedingung für aktuelle Agenten lautet:
In beiden Intervallen I_1 und I_2 vorhanden,
mit I_1 wie oben und $I_2 := [t, t+\text{SUBST\_INTERVAL}]$.

```
3 50 12 10 29 60 170 1
3 50 13 11 28 60 170 1
3 50 14 12 27 60 170 1
3 50 15 13 26 60 170 1
<keine Daten zu Zeitpunkt 16>
3 50 17 15 24 60 170 1
3 50 18 16 23 60 170 1
```

Abbildung 5.2: Ausschnitt von Zeitpunkt 12 bis 18 der für eine Verweildauer von insgesamt 50 Zeitpunkten ermittelten geometrischen Lagedaten des Agenten mit der Identifikationsnummer 3. Jede Zeile enthält folgende Daten: Objekt-Identifikationsnummer, Verweildauer, Zeitpunkt, X-Koordinate, Y-Koordinate, Breite, Höhe und Identifikationsnummer für die Objektart. Wobei keine Daten für Zeitpunkt 16 vorliegen.

Da die betrachteten Zeitpunkte t und t_{-1} nicht Abstand 1 haben müssen, muss das Intervall zwischen ihnen, bis zu der Größe von SUBST_INTERVAL, aber nicht über den Zeitpunkt $t-1$ hinaus, bei obiger Betrachtung noch dem ersten Intervall I_1 zugedacht werden. Damit ergibt sich:
$I_1 := [t_{-1}-\text{SUBST\_INTERVAL}, min(t_{-1}+\text{SUBST\_INTERVAL}, t-1)]$.

Zur Umsetzung musste in der abstrakten Klasse AbsTraversal eine Variable angelegt werden für SUBST_INTERVAL und in der Klasse Timeline folgende Methoden angepasst werden:

```
public synchronized Vector<String>
   getNewAgentsForTimePoint(int _tp, int _last, int _interval)
public synchronized Vector<String>
   getOldAgentsForTimePoint(int _tp, int _last, int _interval)
```

5.1.2 Erkennen der unvollständigen Information in der Zeitleiste

Zum Auffinden der in dem Traversierungsschritt interessanten Stellen mit fehlender Information (Lücken in der Zeitleiste) muss zu jedem Zeitpunkt im Intervall $[t-\text{MIN\_INTERVAL}+1, t+\text{MIN\_INTERVAL}-1]$ untersucht werden, ob für alle alten Agenten Daten für mindestens ein F-LIMETTE-Fakt ‘has_status’ vorhanden sind und zu jedem Zeitpunkt im Intervall $[t, t+\text{MIN\_INTERVAL}-1]$, selbiges für alle neuen Agenten untersucht werden.

Dies geschieht in der konkreten Klasse OfflineTraversal in der Methode

```
public void run();
```

nach

```
Vector<AgentStatus> all\_states = timeLine.getStatesForTimePoint(time);
```

Dazu muss zu einem Agent und Zeitpunkt das Vorhandensein von Daten überprüft werden können.

Die Methode

```
public synchronized Vector<AgentStatus>
   getStatesForTimePointByAgent(int _tp, String _name)
```

in der Klasse TimeLine liefert im Falle des Nichtvorhandenseins der Daten einen leeren Vektor:

```
new Vector<AgentStatus>();
```

5.1.3 Daten in der Zeitleiste finden, die als Grundlage zum Vervollständigen der Information dienen können

Zum Finden passender Daten, die als Grundlage für inferierte Daten zum Vervollständigen der Daten dienen können, wird die Zeitleiste betrachtet, die die Daten aus dem visuellen Teilsystem enthält: Ausgehend vom Zeitpunkt t einer Lücke wird jeweils in der Vergangenheit und der Zukunft bis zu SUBST_INTERVAL viele Zeitschritte weit in der Zeitleiste nach den zeitlich nächstliegenden Daten für den Agenten, dessen Daten an der Stelle zu vervollständigen sind, gesucht.

Hierzu wird eine weitere Methode in der Klasse TimeLine benötigt:

```
public Vector<AgentStatus>
   getSubstitutesForTimePointByAgent(int _tp, int _interval,
                                     String _agent_name)
```

Diese liefert zwei Eingangs-Datensätze aus denen der inferierte Datensatz bestimmt werden soll, der die Datenstruktur vervollständigt und einen Wert für den Zusicherungsgrad des inferierten Datensatzes, je nachdem wie weit die beiden Eingangs-Datensätze vom Zeitpunkt t der Lücke entfernt sind. Als Eingangs-Datensätze werden auf der Zeitleiste die zeitlich nächstliegenden für den betreffenden Agenten in Vergangenheit und

Zukunft verwendet.
Zum Bestimmen des Zusicherungsgrades werden die jeweils nötigen Schrittanzahlen für das Erreichen der beiden Ersatz-Daten genommen und daraus $1 - \frac{Schrittzahl}{\text{SUBST\_INTERVAL } +1}$ berechnet. Der Zusicherungsgrad ergibt sich aus dem Minimum der beiden Werte:

$\min_{obj}(1 - \frac{Schrittzahl(obj)}{\text{SUBST\_INTERVAL } +1})$.

5.1.4 Daten zum Auffüllen der unvollständigen Information kumulieren

Aus den beiden Eingangs-Datensätzen und dem Zusicherungsgrad soll nun der inferierte Datensatz konstruiert werden. Da unklar ist, welcher Art die einzelnen Daten sind, das heißt ob es sich um nominale oder skalare Daten handelt, oder etwa um zirkulär zu betrachtende Zahlen wie Gradangaben, muss das Zusammenfügen zweier Datensätze zu einem Ersatz von der Klasse vorgenommen werden, die das Wissen über die Beschaffenheit der Daten besitzt. Dies bedeutet, es wird in allen Unterklassen der Klasse AgentStatus folgende Methode benötigt:

```
public ViratAgentStatus
   combineAgentStatesToSubstitute(Vector<AgentStatus> substitutes)
```

die aus zwei gegebenen Datensätzen komponentenweise einen Ersatz-Datensatz bestimmt.

5.1.5 Kumulieren von Einzeldaten

Zum Kumulieren der einzelnen Daten in den Datensätzen wird der Mittelwert zweier Daten für verschiedene Datentypen benötigt.

- Für metrische Werte ist dies $\frac{a+b}{2}$.

- Für Nominalzahlen, wie zum Beispiel die zahlenkodierten Objektklassen 1 = Fußgänger, 2 = Auto, 3 = Radfahrer, muß man das über eine Fallunterscheidung lösen, da der Mittelwert 2 nicht zu einem absteigenden Radfahrer passt.

- Für Gradzahlen gilt:

```
public static final Double meanOfDegrees(Double a, Double b) {
    Double x = a+b/2;
    // 'wrong side' problems if |x-a| > 90 or |x-b| > 90
    if (a-b > 180 || b-a > 180) {
        if (a < 0 || b < 0) {      // Degrees from 0 to +/-180
            if (x > 0) { return x - 180; }
            else { return x + 180; }
        }
        if (a > 180 || b > 180) {  // Degrees from 0 to 360
            if (x < 180) { return x + 180; }
            else { return x - 180; }
        }
    }
    return x;
}
```

5.2 Unvollständige Daten über mehrere Zeitpunkte hinweg

Betrifft die Unvollständigkeit der Daten, beispielsweise durch Verdeckung eines Objektes, einen ganzen Zeitraum, so führt dies dazu, dass die Traversierung des SGTs zu dem Zeitpunkt abbricht, zu dem keine Daten mehr vorhanden sind und sobald wieder Daten vorhanden sind, komplett neu von der Wurzel aus ausgewertet wird, als sei das Objekt gerade erst aufgetaucht. Temporale Abfolgen, die in einem Situationsgraph durch Prädiktionskanten modelliert wurden, würden nicht erkannt werden, wenn nur ein Teil der Abfolge durch Datenunvollständigkeit fehlt. Es muss also ein Ansatz gefunden werden, wie man die Traversierung trotz Unwissen Weiterlaufen lassen, und bei Wiedervorhandensein von Daten auf Konsistenz überprüfen kann.

5.2.1 Erweiterung der SGT-Traversierung

Es stellte sich heraus, dass dies gut im Traversierungsalgorithmus und nicht im SGT-Editor zu realisieren ist. Dort werden nun alle SGT-Abläufe (siehe Kapitel 3.2.4) mit Zusicherungsgrad 0.0 weiterverfolgt, auch wenn das nächste Situationsschema auf dem Pfad nicht ausgeprägt werden konnte. Dies ist gleichbedeutend damit im Traversierungsalgorithmus 1 in Zeile 15 folgendes einzufügen:

---

if $\neg(predSit \textit{ is start situation }) \wedge predSit \in G$ **then**
  continue instantiation with Degree of Validity 0.0;

Erreicht wird dies durch Einfügen der Zeilen zwischen den Trennlinien im Traversierungsalgorithmus:

```
always (traversal_instantiate_situation(S,Vars) :-
          (  (_X {sgt_incr_state(S,Vars)} SitPoss)
            ,(   traversal_specialize_situation(S,Vars)
              ;  (  traversal_do_actions(S,Vars)
                    ,writeln('STORED'),writeln(SitPoss),writeln(S),writeln(Vars)
                    ,fail
                 )
              ;  true
             )
          )
// ------------------------------------------------------------------------------
        ;  (  not(sgt_start_sit(S))
              ,writeln('STORED'),writeln(0.0),writeln(S),writeln(Vars)
              ,true
          )
// ------------------------------------------------------------------------------
        ;  true
).
```

5.3 Darstellung des Zusicherungsgrades in der Ausgabe

Infolge der Änderungen wurde es noch interessanter, den Zusicherungsgrad der Ergebnisse mit diesen zu visualisieren. Dieser wird in der Traversierung in F-LIMETTE beim Auswerten des Zustandsschemas bestimmt und mittels writeln in der F-LIMETTE-Antwort an den SGT-Editor übermittelt. Diese Antwort wird in folgender Methode in der Klasse AbsVehicleTraversalModul geparst:

```
public Vector<SituationInstantiation>
   getSituationInstantiationsFromAnswer(Vector<String> given_answer,
                                        double previous_validity,
                                        SituationInstantiation _parent)
```

Zur Visualisierung wurde in der Klasse OfflineTraversal:

```
writeToResultFile(agent, act_out_inst.getCommands(), true);
```

Ersetzen durch:

```
Vector<String> act_out_inst_sitposs_string = new Vector<String>();
for(String act_out_inst_string : act_out_inst.getCommands())
{
 act_out_inst_sitposs_string.add(Double.toString(act_out_inst.getPoss())
                                 + " | " + act_out_inst_string);
}
writeToResultFile(agent, act_out_inst_sitposs_string, true);
```

Kapitel 6

Experimente

Die erstellte Regelbibliothek und die Erweiterungen im SGT-Editor und bei der Situationsgraphenbaumtraversierung müssen an vergleichbaren Daten ausgewertet werden. Im Folgenden wird der verwendete Datensatz, die Methoden der Auswertungen und die Ergebnisse dargestellt.

6.1 VIRAT Datensatz

Anfang 2011 wurde der Datensatz „VIRAT Video Dataset Release 1.0“ [OHP+11] veröffentlicht. Dieser Datensatz enthält Videos von sechs verschiedenen Orten. Aufgenommen wurde mit stationären HD-Kameras mit 1080p oder 720p. Von drei dieser Orte gibt es Trainingsvideos zu denen Annotationen vorliegen, die einzelbildweise die Lage von Objekten sowie vorkommende Situationen enthalten. Des Weiteren gib es zu allen sechs Orten zusätzliche Testvideos. Zum Objektort wurde in je einer Zeile pro Objekt und Einzelbild die Koordinaten der objektumgebenden Rechtecke annotiert, im Detail:

- Die eindeutige Identifikationsnummer des Objektes (nicht unbedingt fortlaufend vergeben).
- Verweildauer des Objektes in Anzahl Einzelbilder.
- Nummer des betrachteten Einzelbildes im Video (ab 0 gezählt).
- X- und Y-Koordinate der linken oberen Ecke des umgebenden Rechtecks.
- Breite und Höhe des umgebenden Rechtecks.
- Kodierung der Objektart: 0=Unbekannt, 1=Person, 2=Auto, 3=anderes Fahrzeug, 4=anderer Gegenstand, 5=Fahrrad.

Folgende Situationen liegen zu den Trainingsvideos annotiert vor:

0 = Unbekannt.

1 = Person lädt einen Gegenstand in ein Fahrzeug ein.

2 = Person lädt einen Gegenstand aus einem Fahrzeug aus.

3 = Person öffnet den Kofferraum eines Fahrzeuges.

4 = Person schließt den Kofferraum eines Fahrzeuges.

5 = Person steigt in ein Fahrzeug ein.

6 = Person steigt aus einem Fahrzeug aus.

Mit jeweils einer Zeile pro auftretendes Ereignis wurde folgendes annotiert:

- Identifikationsnummer des Ereignisses (getrennt von Identifikationsnummern der Objekte).
- Kodierung der Art des Ereignisses (Nummer 0 bis 6 der Situationen siehe oben).
- Dauer des Ereignisses in Anzahl Einzelbilder.
- Nummer des Startbildes.
- Nummer des letzten an diesem Ereignis beteiligten Bildes.
- X- und Y-Koordinate der linken oberen Ecke des Rechtecks um das Ereignis.
- Breite und Höhe des umgebenden Rechtecks.
- Anzahl der am Ereignis beteiligten Objekte.
- Zu jeder im Video vergebenen Objekt-Identifikationsnummer in aufsteigend sortierter Reihenfolge je eine 0 für nicht beteiligt bzw. eine 1 für beteiligt.

Die Experimente wurden mit Daten aus dem Virat-Datensatz durchgeführt. Stellvertretend für die Ausgabe eines sehr zuverlässigen Trackingsystems wurden die geometrischen Lagedaten aus den annotierten Trainingsdatensätzen verwendet. Zur Simulation lückenhafter Daten wurden beim Einlesen der Dateien zufallsgesteuert einzelne Zeilen, das heißt die kompletten Lagedaten eines Objektes zu einem Zeitpunkt, ausgelassen. Verwendet wurden folgende Daten zu Videos der beiden mit 0000 und 0002 bezeichneten Orte:

- VIRAT_S_000002.viratdata.objects.txt
- VIRAT_S_000003.viratdata.objects.txt

- VIRAT_S_000004.viratdata.objects.txt
- VIRAT_S_000006.viratdata.objects.txt
- VIRAT_S_000200_06_001693_001824.viratdata.objects.txt
- VIRAT_S_000202_00_000000_000977.viratdata.objects.txt

Für die beiden Orte wurde jeweils ein F-LIMETTE Regel-Interface angelegt, das die im Datensatz angegebene projektive Transformation der Bildpunkte auf die Bodenfläche der Szene in Abbildung 6.1 umsetzt:

```
always (projectiveTransform(X, Y, X_neu, Y_neu) :-
        XX is (X * -0.00025796104) + (Y * 0.00093565491) - 0.49343613
        , YY is (X * 0.00057183404) + (Y * 0.00082778391) - 0.86963374
        , ZZ is (X * 0.0000015464549) + (Y * 0.000040565615) + 0.016012838
        , X_neu is (XX / ZZ)
        , Y_neu is (YY / ZZ)
).
```

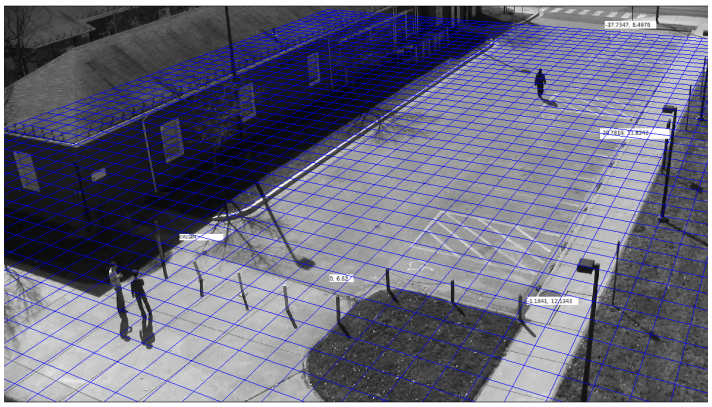

Abbildung 6.1: Bildliche Darstellung der projektiven Transformation der Bildpunkte auf die Bodenfläche der Szene, passend zum Ort 0000 aus dem Datensatz [OHP+11].

Der restliche Teil der Interface-Dateien hängt nur vom Datensatz selbst ab und ist daher für beide Orte gleich. Er enthält die nötigen Schnittstellen zum Format der geometrischen Lagedaten, siehe Abbildung 6.3 und F-LIMETTE-Regeln zu einer Ausgabe,

die möglichst nah an der im Datensatz [OHP+11] verwendeten liegt, siehe Abbildung 6.2.
Es wurde mit dem SGT-Editor ein passender Situationsgraph, siehe Abbildungen A.1 und A.2, zum Erkennen folgender Situationen erstellt:

0 = Zwei Personen mit kleinem Abstand kommen sich noch näher.

1 = Person lädt einen Gegenstand in ein Fahrzeug ein.

2 = Person lädt einen Gegenstand aus einem Fahrzeug aus.

3 = Zwei Personen stehen beisammen.

4 = Zwei Person laufen gemeinsam.

5 = Person steigt in ein Fahrzeug ein.

6 = Person steigt aus einem Fahrzeug aus.

```
0.795658 | OUTPUT: 1 1 15182 15182 1131 392 303 229 3 obj\_8 obj\_6 obj\_10
0.801166 | OUTPUT: 1 1 15183 15183 1131 392 303 229 3 obj\_8 obj\_6 obj\_10
0.801166 | OUTPUT: 1 1 15184 15184 1131 392 303 229 3 obj\_8 obj\_6 obj\_10

7 1 85 15147 15231 1052 368 189 194 3 0 0 0 0 1 1 0 0 1
```

Abbildung 6.2: Ausgabeformat einer erkannten Situation wie es der SGT-Editor mit dem verwendeten Interface produziert: Zusicherungsgrad, Art der Situation (1 für Einladen), Dauer, Startzeitpunkt, Endzeitpunkt, X-Koordinate, Y-Koordinate, Breite und Höhe des umgebenden Rechtecks der Situation, Anzahl und Aufzählung beteiligter Objekte (oben) und Ausgabeformat im Datensatz: Identifikationsnummer der Situation, Art der Situation, Dauer, Startzeitpunkt, Endzeitpunkt, X-Koordinate, Y-Koordinate, Breite und Höhe des umgebenden Rechtecks der Situation, Anzahl und Kodierung der beteiligten Objekte (unten).

Das Video mit der Kennziffer 02 des Ortes 0000 ist 9074 Einzelbilder lang und enthält 10 Objekte, davon 7 Agenten. Es liegen dazu 47466 Datensätze zur geometrischen Lage der Objekte zu verschiedenen Einzelbildern vor. Bei Video 03 sind dies 14641 Einzelbilder, 12 Objekte, 6 Agenten und 20597 Datensätze, bei Video 04 14838 Einzelbilder, 9 Objekte, 4 Agenten, 24797 Datensätze und bei Video 06 17166 Einzelbilder, 22 Objekte, 11 Agenten und 44182 Datensätze. Von den beiden verwendeten Videos zu Ort 0002 ist das Video mit der Nummer 00 das längste mit 29307 Einzelbildern, 21 Objekte, 12 Agenten und 206161 Datensätzen und im Gegensatz dazu das mit der Nummer 06 insgesamt das kürzeste mit 3904 Einzelbildern, 5 Objekten, 2 Agenten und 8996 Datensätzen, siehe auch Tabelle 6.1.

```
// x * Co = 1 cm
always (get_spatial_coefficient(Co) :-
        Co is 100.0
).

// Co = frame/sec
always (get_time_coefficient(Co) :-
        Co is 10.0
).

always (has_image_geometry(Agent,X,Y,W,H) :-
        has_status(Agent,_,X,Y,W,H,_)
).

always (is_person(Agent) :-
        has_status(Agent,_,_,_,_,_,1)
).
```

Abbildung 6.3: Weitere F-LIMETTE Regeln im Interface zur Anpassung an den Datensatz beinhalten Skalierungskoeffizienten für Zeit und Längeneinheiten sowie Schnittstellen zu den Lagedaten und der Klassifizierung von Objekten.

6.2 Versuchsdurchführung

Um Vergleichsdaten zu sammeln und einen Überblick zu bekommen wurden die ersten Durchläufe mit den vollständigen annotierten Daten durchgeführt. Dazu wurden Server mit jeweils zwei Quad-Core Xeon E5450 Prozessoren, also 8 logischen Kernen, verwendet. Die Dauer der Durchläufe für die Videos lag zwischen 27 Minuten und 11 Stunden 15 Minuten. Die Zeiten für alle Videos kann man Tabelle 6.1 entnehmen.

Als nächstes wurde Durchläufe mit der Simulation unvollständiger Daten durchgeführt. Dabei lag der Anteil der beim Einlesen ausgelassenen Daten zwischen 15,5% und 16%. Die Laufzeiten der Durchläufe mit unvollständigen Daten wichen nicht merklich von denen der Vergleichsdurchläufe mit vollständigen Daten ab.

Es stellte sich heraus, dass eine Methode sinnvoll ist, die Ergebnisse hinsichtlich ihrer Qualität auf einen Blick einschätzen zu können. Hierzu hat es sich als zweckdienlich erwiesen je Agent und erkannter Situation den Zusicherungsgrad auf einer Zeitachse aufzutragen. Alle Plots in einem Schaubild hat sich dabei als zu unübersichtlich gezeigt, daher wurde die gestaffelte Darstellung gewählt. Schaubilder der gewählten Videos in dieser Form kann man im Anhang unter A.4, A.5, A.6, A.7, A.8, A.9 und A.10 finden.

Video Kenn-ziffer	Laufzeit	Einzel-bilder	Anzahl Objekte	Anzahl Agenten	Anzahl Daten-sätze	Einzel-bilder pro Sekunde	Daten-sätze pro Sekunde
0000 02	7h 22min	9074	10	7	47466	0.342	1.790
0000 03	3h 9min	14641	12	6	20597	1.291	1.816
0000 04	4h 35min	14838	9	4	24797	0.899	1.503
0000 06	5h 6min	17166	22	11	44182	0.935	2.406
0002 00	11h 15min	29307	21	12	206161	0.724	5.090
0002 06	27min	3904	5	2	8996	2.410	5.553

Tabelle 6.1: Einige Eckdaten zu den jeweils ersten Durchläufen mit perfekten Daten der verwendeten Videos.

Die ersten Ergebnisse enthielten einen sehr hohen Anteil an Falschalarm, wie man in Abbildung 6.4 B sehen kann. Die erste Vermutung, dass dies alleine an einem zu klein gewählten Wert für SUBST_INTERVAL liegt, das heißt der Größe des Intervalls in dem vorhandene Daten für die inferierten Daten zum Vervollständigen der Datenstruktur verwendet werden, war falsch. Das Problem liegt nicht nur an Zeitpunkten für die keine Daten inferiert werden können, sondern vielmehr auch in den inferierten Daten selbst.

Das Einsteigen in ein Fahrzeug sowie das Einladen eines Gegenstandes in ein Fahrzeug wird über das Verschwinden eines passenden Objektes neben einem Fahrzeug erkannt, was wiederum als Wechsel von Vorhandensein zu Nichtvorhandensein von geometrischen Lagedaten modelliert wird. Die inferierten Daten zum Auffüllen der unvollständigen Daten haben einen Zusicherungsgrad kleiner 1.0, was bedeutet, dass an diesen Stellen zu geringem Zusicherungsgrad immer das Nichtvorhandensein geometrischer Lagedaten geschlossen werden kann. Analoges gilt für Aussteigen, Ausladen und der Wechsel von Nichtvorhandensein zu Vorhandensein geometrischer Lagedaten. Dies führte bei den ersten Versuchen mit unvollständigen Daten, die durch inferierte Daten aufgefüllt wurden, zu einer hohen Falschalarmrate. Das Erhöhen des Wertes für SUBST_INTERVAL von 4 auf 6 und als weitere stabilisierende Maßnahme das Verwenden von Nichtvorhandensein der geometrischen Lagedaten zu zwei statt einem Zeitpunkt in den Prädikaten appear und disappear (siehe 4.2) verbesserten das Ergebnis merklich. Und es konnte eine weitere Verbesserung durch Nutzen von UND und NOT in den Prädikaten appear und disappear in der *strong* Variante erzielt werden.

Einen Vergleich von annotierten Situationen, Ergebnis des Vergleichsdurchlaufs anhand annotierter Daten und Ergebnis eines Durchlaufs mit simuliert unvollständigen Daten für Video 0002 06 findet man im Schaubild A.3.

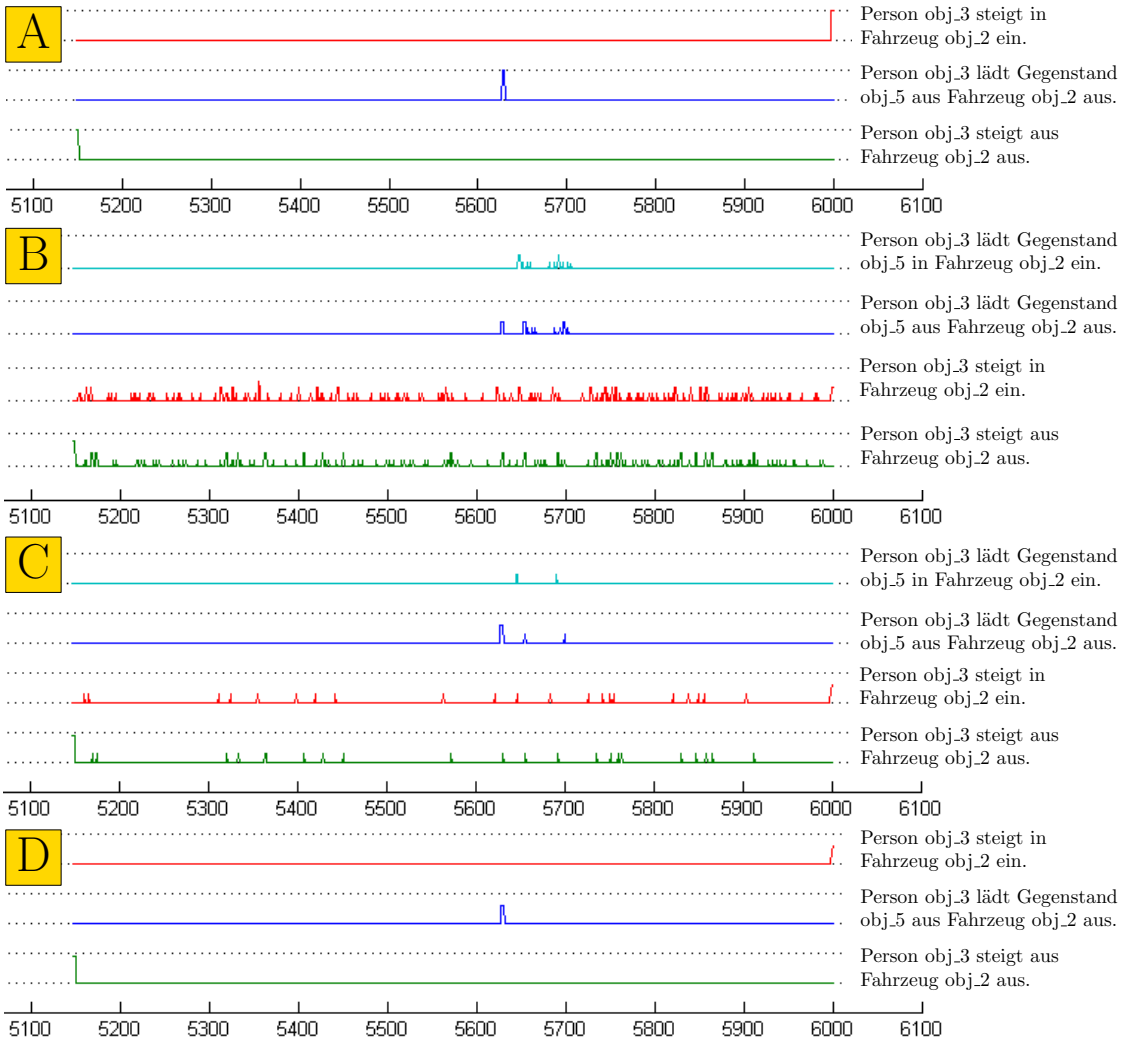

Abbildung 6.4: Versuchsreihe eines Zeitabschnitts aus Video 0000 03: A mit perfekten Daten. B bis D mit unvollständigen Daten, wobei von B auf C SUBST_INTERVAL von 4 auf 6 erhöht wurde und der Nichtvorhandenseinstest über zwei Zeitpunkte statt einem ermittelt wurde. Ergebnis D wurde durch Umstellung der UND und NOT in den Prädikaten appear und disapper auf die *strong* Variante erreicht.

Der Datensatz [OHP+11] enthält Matlab-Skripte zum Erzeugen von Ergebnisbildern mit eingezeichneten Rechtecken der Lage einer detektierten Situation. Dies war sehr hilfreich beim Einordnen der Ergebnisse. Einen Vergleich der drei ersten erkannten Situationen in Video 0000 03, dieselben wie in der Übersicht 6.4, kann man in der Abbildung 6.5 sehen. In der ersten Zeile das Ergebnis mit den annotierten Situationen des Trainigsdatensatzes, in der zweiten Zeile das Ergebnis des Vergleichsdurchlaufs mit vollständigen annotierten Lagedaten des Trainigsdatensatzes und in der dritten Zeile

das Ergebnis eines Durchlaufs mit simuliert unvollständigen Daten und der aktuellsten Version von Regelbibliothek und SGT-Editor.

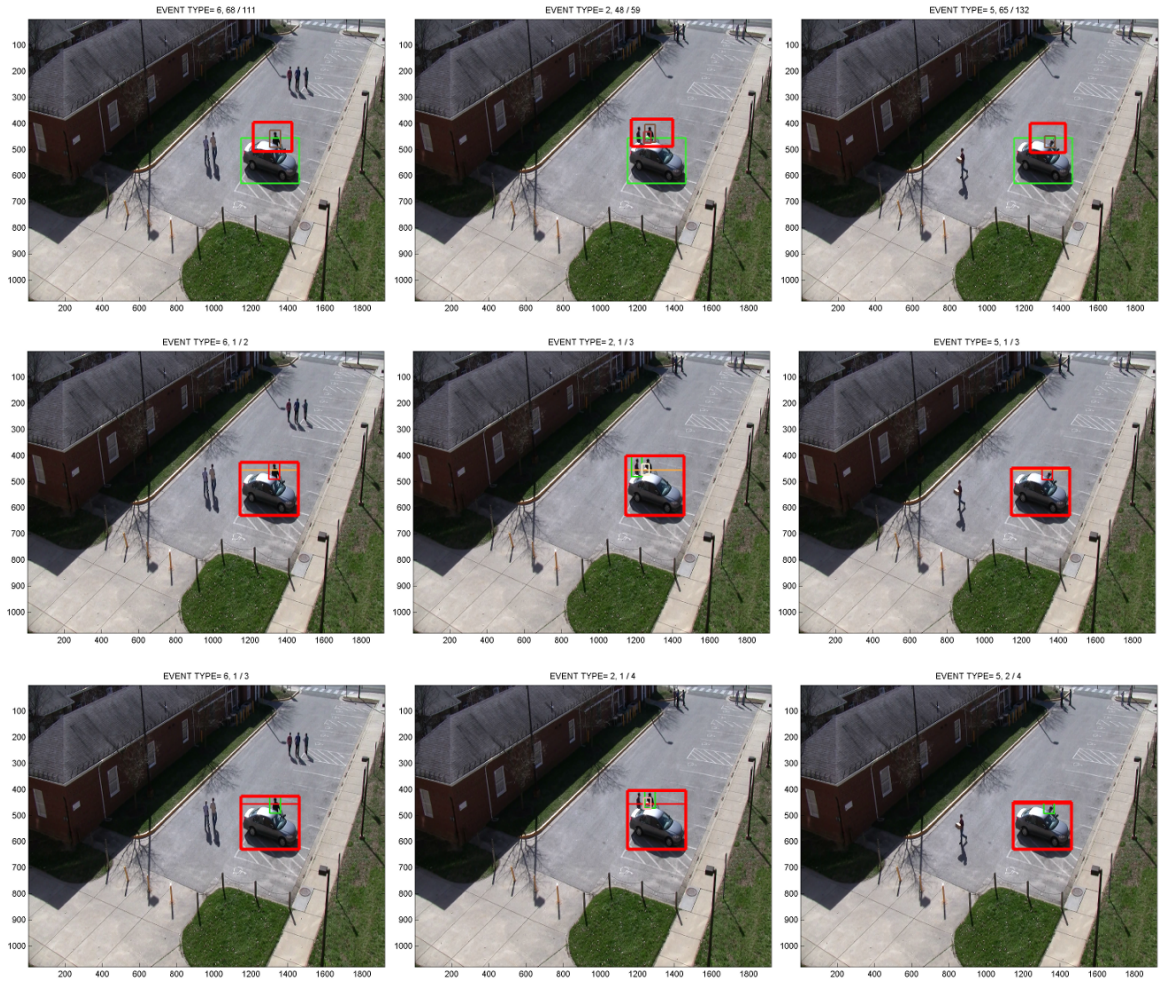

Abbildung 6.5: Mit den in [OHP+11] mitgelieferten Matlab-Skripten erzeugte Ergebnisbilder der ersten drei Situationen aus dem Video 0000 03: Person steigt aus einem Fahrzeug aus, Person lädt einen Gegenstand aus einem Fahrzeug aus und Person steigt in ein Fahrzeug ein. In der oberen Zeile erzeugt anhand der mitgelieferten annotierten Situationen in den Trainingsdaten, in der mittleren Zeile mit den Ergebnissen des Vergleichsdurchlaufs auf den annotierten Daten erzeugt und in der unteren Zeile unter Verwendung des Ergebnisses eines Durchlaufs mit simuliert unvollständigen Daten und der aktuellsten Version der Regelbasis und des SGT-Editors erzeugt.

Verglichen mit den annotierten Ergebnissen des Virat-Trainingsdatensatzes ist die Zeitdauer der erkannten Ereignisse kürzer und das umgebende Rechteck etwas größer. Für

das Öffnen und Schließen des Kofferraums konnte bisher keine Möglichkeit gefunden werden, diese Handlungen anhand der geometrischen Lagedaten der beteiligten Objekte zu erkennen.
Der Datensatz liefert ebenfalls eine Software zur automatischen Auswertung der Güte eigener Ergebnisse.
Hierbei werden folgende Werte betrachtet:

- Exaktheit (Precision): $\frac{TP}{D}$
  TP = Anzahl korrekter Detektionen.
  D = Anzahl aller Detektionen (richtig oder falsch).
- Teststärke (Probability of Detection): TP / T
  T = Anzahl Ereignisse in der Grundwahrheit.
- Falschalarmrate (FAR): FP / NORM
  FP = Anzahl falsch positiv detektierter Ereignisse.
  NORM = Normalisierungsfaktor um auf Ereignisse pro Minute zu kommen.
- F-score: Harmonisches Mittel aus Teststärke und Exaktheit.

$$F = \frac{1}{\frac{1}{2}(\frac{1}{Teststärke} + \frac{1}{Exaktheit})} \tag{6.1}$$

- Gewichteter aggregierter F-Score.
  Erfasst die allgemeine Güte der Detektoren.

$$\frac{1}{w_i * \sum_{i=1}^{n} \frac{1}{F_i}} \tag{6.2}$$

  Die Formel 6.2 wird über einer Menge positiver F-Score Ergebnisse F_i berechnet. Gewichte w_i proportional zur Anzahl der Ereignisse ergeben in der Summe 1.

Die automatische Auswertung mit der Scoring Software, die in dem VIRAT Datensatz [OHP+11] mitgeliefert wird, war mit den Ergebnissen nicht möglich, da die Zeitdauer der erkannten Situationen verglichen mit den Zeitspannen, die die annotierten Situationen andauern, deutlich kürzer ist.

```
Number of ground truth events
 0 0 1 1 1 1 1
Number of total detection events
 0 1 2 0 0 1 1
Number of correct detection events
 0 0 0 0 0 0 0
Number of false detection events
 0 1 2 0 0 1 1
Precision (positive detected / total detected)
```

```
 0 0 0 0 0 0 0
Probability of detection  (positive detected / ground truth)
 0 0 0 0 0 0 0
Total number of minutes
5.03333
False alarm rate
 0 0.198675 0.397351 0 0 0.198675 0.198675
F-score
 -1 -1 0 0 0 0 0
Weighted aggregate F-score
0
Inputs
VIRAT_S_000002
```

Dieses Problem besteht unabhängig davon ob mit unvollständigen oder perfekten Daten gearbeitet wurde. Vergrößert man die Dauer, indem man zu jeder erkannte Situation jeweils vor dem ersten erkannten Zeitpunkt 30 Einzelbilder mit den Daten zum ersten Einzelbild der erkannten Situation und nach dem letzten Zeitpunkt 30 Einzelbilder mit den Daten im letzten Einzelbild der erkannten Situation ergänzt, so sieht es schon besser aus, was darauf schließen lässt, dass eine Modellierung der Situationen mit längerer Dauer hier Abhilfe schaffen kann.

```
Number of ground truth events
 0 0 1 1 1 1 1
Number of total detection events
 0 1 2 0 0 1 1
Number of correct detection events
 0 0 1 0 0 1 1
Number of false detection events
 0 1 1 0 0 0 0
Precision (positive detected / total detected)
 0 0 0.5 0 0 1 1
Probability of detection  (positive detected / ground truth)
 0 0 1 0 0 1 1
Total number of minutes
5.03333
False alarm rate
 0 0.198675 0.198675 0 0 0 0
F-score
 -1 -1 0.666667 0 0 1 1
Weighted aggregate F-score
0.857143
Inputs
VIRAT_S_000002
```

Durch die in dieser Arbeit umgesetzten Verfahren ist die Situationserkennung robust bei Ausfall bis zu einer Anzahl von SUBST_INTERVAL (siehe 5.1.2) temporal aufein-

anderfolgender Daten für ein Objekt. Da mit steigendem Wert von SUBST_INTERVAL mehr inferierte Daten mit geringerem Zusicherungsgrad verwendet werden, kann der Wert jedoch nicht beliebig groß gewählt werden. In den Experimenten wurde mit bis zu sechs Zeitpunkte langen Datenlücken das Potential noch nicht ausgeschöpft.

Kapitel 7

Zusammenfassung

7.1 Erreichte Ziele

Zur Vorbereitung der Experimente wurde die F-LIMETTE Regelbibliothek passend für den Diskursbereich Parkplatz und Personenüberwachung und passend zu dem VIRAT-Datensatz ergänzt und um einen Modultest erweitert. Es wurden zwei Arten von unvollständigen Informationen betrachtet, einerseits Datenausfall für ein Objekt zu einem Zeitpunkt, bedingt durch eher technische Probleme im visuellen Teilsystem, was dazu führt, dass keine Auswertung in dem Zeitintervall um den Ausfallzeitpunkt möglich ist. Hierfür wurde eine Lösung in Form von inferierten Daten zum Komplettieren der unvollständigen Daten gefunden und im SGT-Editor implementiert. Nach einer Anpassung in der Regelbibliothek konnten mit dieser Lösung Situationen auf unvollständigen Daten annähernd genauso gut erkannt werden, wie auf perfekten Daten. Für Unwissen über ein Objekt, das mehrere Zeitpunkte andauert, was beispielsweise durch Verdeckungen passieren kann, wurde der Traversierungsalgorithmus ergänzt, so dass er mit Zusicherungsgrad 0.0 in solchen Fällen weiterläuft.

7.2 Ausblick

Die Änderung im Traversierungsalgorithmus gilt es noch intensiver zu testen. Hierfür waren der in dieser Arbeit verwendete SGT sowie die betrachteten Situationen und Daten nicht ausreichend geeignet, da zu wenig komplexe zeitliche Abfolgen modelliert wurden.
Das Erkennen von komplexen Gruppenaktionen mit vielen Objekten konnte bisher nicht erreicht werden, aufgrund mangelnder Daten. Hierzu kann man verschiedene Ansätze verfolgen. Einerseits kann man versuchen, diese passend in einem SGT zu modellieren, was bei variabler Anzahl beteiligter Objekte nicht ganz einfach ist. Es kam auch die Idee auf, im SGT-Editor das Sichern von Zwischenergebnissen zu ermöglichen. Beispielsweise durch eine zusätzlich eingebundene dynamische F-LIMETTE-Regeldatei, die zu Anfang leer ist und im Handlungsschema durch Fakten erweitert werden kann.

Eine andere Möglichkeit wäre es die Ausgabedatei in Form von F-LIMETTE Regeln und Fakten zu halten und mit dieser eine finale Auswertung in F-LIMETTE durchzuführen.
Um die automatische Auswertung mit dem Scoring-Programm, das dem VIRAT-Datensatz beiliegt, zu ermöglichen sollte man versuchen den SGT so zu gestalten, dass die Dauer der erkannten Situationen länger ist. Ebenfalls ist zu überlegen die umgebenden Rechtecke der Ergebnisse kleiner zu gestalten als das alle beteiligten Objekte einschließende Rechteck, insbesondere wenn ein Fahrzeug an der Situation beteiligt ist.
Beim Umgang mit unvollständigen Daten kann man noch einige Varianten ausprobieren: Beispielsweise statt Kumulieren zweier Datenvektoren benachbarter Zeitpunkte zum Ausfüllen einer Lücke das Verwenden beider Datenvektoren mit unveränderten Werten für den zu ergänzenden Zeitpunkt. Auch wäre es denkbar statt Mittelung zweier umgebender Rechtecke das Rechteck, das beide enthält oder den Schnitt der beiden für die inferierten Daten zu verwenden.
Bei einem System, bei dem so viele Komponenten zusammenspielen, wären automatische Tests für das Gesamtsystem, die Unit-Tests für alle Einzelkomponenten enthalten, sinnvoll.
In dieser Arbeit wurde ausschließlich die Offline-Anwendung auf gespeicherten Daten des SGT-Editors verwendet. Es steht noch aus, das Erreichte auf das ebenfalls vorgesehene Onlinesystem zu übertragen und gegebenenfalls passend zu ergänzen. Damit kann die Situationserkennung in ein echtzeitfähiges Gesamtsystem integriert werden.

Abbildungsverzeichnis

Glossar

Ablauf
Alle zeitabhängigen Zustandsänderungen. 3

Agent
Mobile Komponente mit internen Freiheitsgraden, das heißt mehr als einer Bewegungsmöglichkeit zur Auswahl. 3

Aktion
Ablauf von Geschehen; Verkettung von Handlungen eines Agenten zum Erreichen eines von ihm angestrebten Zieles. 3

Bildauswertung
Algorithmische Transformation eines Bildes in eine Aussage. 61

Bildbereich
Problembereich der Bildauswertung, der sich damit beschäftigt ohne zusätzliches Hintergrundwissen Bildbereichshinweise, beispielsweise Hinweise auf bestimmten lokalen Grauwertverlauf, und Bildmerkmale aus der Bildinformation zu erlangen. 4

Diskursbereich
Ausschnitt aus der realen Welt, Anwendungsbereich für den ein erörtertes System Gültigkeit besitzt. 3, 14, 15, 33, 34

F-LIMETTE
F-LIMETTE steht für Fuzzy Logic Programming Integrating Metric Temporal Extensions. 5, 12, 15, 38, 40

Geschehen
Elementare Abläufe für die es in der deutschen Sprache ein Verb gibt. 3, 4

Handlung
Zeitweise Ausführung einer Tätigkeit durch ein Individuum. 61

Situationsgraphenbaum
Ein Situationsgraphenbaum ist eine strukturierte Anordnung einzelner Situationsschemata, zum Zweck Hintergrundwissen über das erwartete Verhalten von Agenten zu repräsentieren. Eine ausführliche Beschreibung siehe Kapitel 3.2. 15, 63

Szene
Betrachteter geometrischer Ausschnitt der realen Welt mit dreidimensionaler Lage, Beschaffenheit und Zustand aller darin befindlichen Objekte. 3

Szenenbereich
Die dreidimensionale Szene, die auf die zweidimensionale Bildebene der Kamera abgebildet wird. 4

Trajektorie
Lage des Abbildes eines Objektes als Funktion der Bildnummer. 4

Verhalten
Folge von Bewegungen eines Agenten zum Verfolgen einer Absicht. 3, 5

Abkürzungsverzeichnis

CHMM	Coupled Hidden Markov Model 7
FHL	unscharfe Hornlogik 12, 14
FL1	unscharfe Logik erster Ordnung 12, 14
FMTHL	unscharfe metrisch-temporale Logik eingeschränkt auf das Hornfragment 5, 8, 12, 14, 15
HHMM	Hierarchisches Hidden Markov Model 7
HMM	Hidden Markov Model 6, 7
MTHL	metrisch-temporale Hornlogik 12, 14
MTL	metrisch-temporale Logik 12, 14
PL1	Prädikatenlogik erster Ordnung 12, 14, 15
SCFG	stochastisch Kontextfreie Grammatik 7
SGT	Situationsgraphenbaum 5, 11, 12, 15–19, 31, 34, 43
VLMM	Variable Length Hidden Markov Model 7

Symbolverzeichnis

$\widetilde{\forall}\mathcal{F}$	Allabschluss $\widetilde{\forall}\mathcal{F} \equiv \forall v_1 \ldots \forall v_n \mathcal{F}$ mit $\{v_i \vert 1 \leq i \leq n\}$ Menge aller freien Variablen in $\mathcal{F}$. 15
$\widetilde{A}$	unscharfe Menge A. 22
$\widetilde{\exists}\mathcal{F}$	Existenzabschluss $\widetilde{\exists}\mathcal{F} \equiv \exists v_1 \ldots \exists v_n \mathcal{F}$ mit $\{v_i \vert 1 \leq i \leq n\}$ Menge aller freien Variablen in $\mathcal{F}$. 15
σ_x	Evidenzverteilung (englisch support distribution) der unscharfen Variablen x. 27, 28
$\mathcal{F}_\Sigma(\mathcal{V})$	Die Menge aller Formeln über der Signatur Σ zur Menge $\mathcal{V}$ von Variablen. 14
G_m	Glaubwürdigkeitsmaß zur Maßbasis m. 24
$[\widetilde{A} \Rightarrow \widetilde{B}]$	Kurzform der linguistischen Regel 'IF x is $\widetilde{A}$ THEN y is $\widetilde{B}$'. 26, 28
m	Maßbasis. 24
μ_A	Zugehörigkeitsfunktion der unscharfen Menge $\widetilde{A}$. 22, 26, 28
N	Notwendigkeitsmaß. 25
Pl_m	Plausibilitätsmaß zur Maßbasis m. 25
Π_x	Possibilitätsmaß (oder auch Möglichkeitsmaß genannt) der unscharfen Variablen x. 25
π_x	Possibilitätsverteilung (oder auch Möglichkeitsverteilung genannt) der unscharfen Variablen x. 25, 26
$\widetilde{\mathcal{P}}(\mathcal{U})$	Unscharfe Potenzmenge von $\mathcal{U}$: Menge aller unscharfen Mengen über $\mathcal{U}$. 23
$[\sigma_{x,y} \downarrow \mathcal{U}_x]$	Projektion der Evidenzverteilung $\sigma_{x,y}$ auf $\mathcal{U}_x$. 28
$[\pi_{x,y} \downarrow \mathcal{U}_x]$	Projektion der Possibilitätsverteilung $\pi_{x,y}$ auf $\mathcal{U}_x$. 26
$\widetilde{R}$	unscharfe Relation. 23
$\mathcal{U}$	endliches Universum als Grundmenge von Elementen $u \in \mathcal{U}$. 22
U	unscharfes Maß. 24
$\mathcal{U}_x$	Universum der Variablen x. 24

Literaturverzeichnis

[AGN08] Arens, M. ; Gerber, R. ; Nagel, H.-H.: Conceptual representations between video signals and natural language descriptions. In: *Image and Vision Computing* 26 (2008), Nr. 1, S. 53 – 66. – ISSN 0262–8856. – Cognitive Vision-Special Issue 5, 8

[All83] Allen, James F.: Maintaining knowledge about temporal intervals. In: *Commun. ACM* 26 (1983), November, S. 832–843. – ISSN 0001–0782 8, 9, 58

[AR11] Aggarwal, J.K. ; Ryoo, M.S.: Human activity analysis: A review. In: *ACM Comput. Surv.* 43 (2011), April, S. 16:1–16:43. – ISSN 0360–0300 6

[Are04] Arens, Michael: *Dissertationen zur Künstlichen Intelligenz (DISKI)*. Bd. 287: *Repräsentation und Nutzung von Verhaltenswissen in der Bildfolgenauswertung*. Berlin : Akad. Verl.-Ges. Aka, 2004. – ISBN 3898382877. – Zugl.: Karlsruhe, Univ., Diss., 2004 2, 5, 8, 13, 14, 16, 18, 125

[Bay63] Bayes, Thomas: *An Essay towards solving a problem in the doctrine of chances*. Philosophical Transactions Royal Society, 1763. – 370–418 S. – Reprinted in Biometrica 45: 293-319 1958 11

[BCN95] Beth, Thomas ; Calmet, Jacques ; Nagel, Hans-Hellmut: *Forschung und Lehre am IAKS: ein Tätigkeitsbericht anläßlich des zehnjährigen Bestehens 1985-1995*. Univ., Inst. für Algorithmen und Kognitive Systeme, 1995 3

[BI98] Bobick, A.F. ; Ivanov, Y.A.: Action recognition using probabilistic parsing. In: *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, 1998. – ISSN 1063–6919, S. 196 –202 7

[BL04] Brachman, Ronald ; Levesque, Hector: *Knowledge Representation and Reasoning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2004. – ISBN 1558609326 15

[BOP97] Brand, M. ; Oliver, N. ; Pentland, A.: Coupled hidden Markov models for complex action recognition. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 0 (1997), S. 994. – ISSN 1063–6919 7

[Bra97] Brand, Matthew: The ´Inverse Hollywood Problem': From video to scripts and storyboards via causal analysis. In: *In Proceedings, AAAI97*, 1997, S. 12–96 7

[Brz96] Brzoska, Christoph: Temporales logisches Programmieren. In: *Informatik - Forschung und Entwicklung* 11 (1996), S. 61–68. – ISSN 0178–3564 15

[Cah88] Cahn von Seelen, Ulf: *Ein Formalismus zur Beschreibung von Bewegungsverben mit Hilfe von Trajektorien*, Universität Karlsruhe, Diplomarbeit, 1988 4

[Coo90] Cooper, Gregory F.: The computational complexity of probabilistic inference using bayesian belief networks. In: *Artificial Intelligence* 42 (1990), Nr. 2-3, S. 393 – 405. – ISSN 0004–3702 11

[Dem68] Dempster, A. P.: A Generalization of Bayesian Inference. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 30 (1968), Nr. 2, S. 205–247 11, 30

[DL93] Dagum, Paul ; Luby, Michael: Approximating probabilistic inference in Bayesian belief networks is NP-hard. In: *Artificial Intelligence* 60 (1993), Nr. 1, S. 141 – 153. – ISSN 0004–3702 11

[DPY93] Dubois, Didier ; Prade, Henri ; Yager, Ronald R. ; Dubois, Didier [. (Hrsg.): *Readings in fuzzy sets for intelligent systems*. San Mateo, CA : Morgan Kaufman Publishers, 1993. – ISBN 1–55860–257–7 11

[GCMH02] Galata, Aphrodite ; Cohn, Anthony ; Magee, Derek ; Hogg, David: Modeling Interaction Using Learnt Qualitative Spatio-Temporal Relations and Variable Length Markov Models. In: *In Proceedings of the European Conference on Artificial Intelligence*, 2002, S. 741–745 7

[GFA07] Guerra-Filho, Gutemberg ; Aloimonos, Yiannis: A Language for Human Action. In: *Computer* 40 (2007), Mai, S. 42–51. – ISSN 0018–9162 7

[GN08] Gerber, R. ; Nagel, H.-H.: Representation of occurrences for road vehicle traffic. In: *Artificial Intelligence* 172 (2008), Nr. 4-5, S. 351 – 391. – ISSN 0004–3702 5, 33

[GRVR09] Gonzàlez, Jordi ; Rowe, Daniel ; Varona, Javier ; Roca, F. X.: Understanding dynamic scenes based on human sequence evaluation. In: *Image and Vision Computing* 27 (2009), Nr. 10, S. 1433 – 1444. – ISSN 0262–8856. – Special Section: Computer Vision Methods for Ambient Intelligence 11

[Hal03] Halpern, Joseph Y.: *Reasoning about uncertainty*. Cambridge, Mass. [u.a.] : MIT Press, 2003. – ISBN 0–262–08320–5 11

[HTG09] Heller, Katherine ; Teh, Yee W. ; Görür, Dilan: Infinite Hierarchical Hidden Markov Models. In: *Journal of Machine Learning Research - Proceedings Track* (2009), S. 224–231 7

[IB99a] Intille, S.S. ; Bobick, A.F.: Visual recognition of multi-agent action using binary temporal relations. In: *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* Bd. 1, 1999 9

[IB99b] Intille, Stephen S. ; Bobick, Aaron F.: A framework for recognizing multi-agent action from visual evidence. In: *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1999 (AAAI '99/IAAI '99). – ISBN 0–262–51106–1, S. 518–525 9

[IB00] Ivanov, Yuri A. ; Bobick, Aaron F.: Recognition of visual activities and interactions by stochastic parsing. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2000), Nr. 8, S. 852–872 7, 8, 58

[IB01] Intille, Stephen S. ; Bobick, Aaron F.: Recognizing planned multiperson action. In: *Comput. Vis. Image Underst.* 81 (2001), März, S. 414–445. – ISSN 1077–3142 9

[JB03] Jaynes, E.T. ; Bretthorst, G.L.: *Probability theory: the logic of science*. Cambridge University Press, 2003. – ISBN 9780521592710 11

[Jef65] Jeffrey, R. C.: *The Logic of Decision*. McGraw-Hill, New York, 1965 11

[KHM11] Kaloskampis, Ioannis ; Hicks, Yulia A. ; Marshall, David: Analysing Engineering Tasks Using a Hybrid Machine Vision and Knowledge Based System Application. In: *Proceedings of the 12th IAPR Conference on Machine Vision Applications (MVA 2011)*, 2011 7

[Kol50] Kolmogorov, A. N.: *Foundations of the Theory of Probability*. Oxford, England: Chelsea Publishing Co., 1950. – 71 S. 11

[Kol92] Koller, Dieter: *Detektion, Verfolgung und Klassifikation bewegter Objekte in monokularen Bildfolgen am Beispiel von Straßenverkehrsszenen*. Sankt Augustin, Universität Karlsruhe, Diss., 1992 3

[Kol95] Kollnig, Henner: *Ermittlung von Verkehrsgeschehen durch Bildfolgenauswertung*. Sankt Augustin, Universität Karlsruhe, Diss., 1995 3

[KSS08] Kitani, Kris M. ; Sato, Yoichi ; Sugimoto, Akihiro: Recovering the Basic Structure of Human Activities from noisy Video-Based Symbol Strings. In: *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)* 22 (2008), S. 1621–1646 7

[LMSR08] Laptev, Ivan ; Marszalek, Marcin ; Schmid, Cordelia ; Rozenfeld, Benjamin: Learning realistic human actions from movies. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 0 (2008), S. 1–8. ISBN 978–1–4244–2242–5 6

[LN99] Leuck, Holger ; Nagel, Hans-Hellmut: Automatic Differentiation Facilitates OF-Integration into Steering-Angle-Based Road Vehicle Tracking. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 2 (1999), S. 2360. – ISSN 1063–6919 3

[ME01] Moore, Darnell ; Essa, Irfan: Recognizing Multitasked Activities using Stochastic Context-Free Grammar. In: *In Proceedings of AAAI Conference*, 2001 7

[ME02] Moore, Darnell ; Essa, Irfan: Recognizing Multitasked Activities from Video Using Stochastic Context-Free Grammar. In: *In Proc. AAAI National Conf. on AI*, AAI, 2002, S. 770–776 7

[MIAS11] Münch, David ; IJsselmuiden, Joris ; Arens, Michael ; Stiefelhagen, Rainer: High-Level Situation Recognition Using Fuzzy Metric Temporal Logic, Case Studies in Surveillance and Smart Environments. (2011) 5, 8, 13

[Mid04] Middendorf, Markus: *Zur Auswertung lokaler Grauwertstrukturen*. Norderstedt, Universität Karlsruhe, Diss., 2004 3

[MJA11] Münch, David ; Jüngling, Kai ; Arens, Michael: Towards a Multipurpose Monocular Vision-based High-Level Situation Awareness System. In: *International Workshop on Behaviour Analysis and Video Understanding (ICVS 2011)*. Sophia Antipolis, France, September 2011 2, 5, 8, 11, 19

[Müc00] Mück, Klaus: *Rechnergestützte Erkennung und Beschreibung innerstädtischer Straßenkreuzungen*, Universität Karlsruhe, Diss., 2000 3

[MYC+10] McKeever, Susan ; Ye, Juan ; Coyle, Lorcan ; Bleakley, Chris ; Dobson, Simon: Activity recognition using temporal evidence theory. In: *Journal of Ambient Intelligence and Smart Environments.* 2 (2010), Nr. 3, S. 253–269 30

[Nag96] Nagel, H.-H.: Zur Strukturierung eines Bildfolgen-Auswertungssystems. In: *Informatik - Forschung und Entwicklung* 11 (1996), S. 3–11. – ISSN 0178–3564 4, 5

[Nag00] Nagel, H.-H.: Image sequence evaluation: 30 years and still going strong. In: *Pattern Recognition, 2000. Proceedings. 15th International Conference on* Bd. 1, 2000, S. 149 –158 vol.1 12, 13

[Nag04] Nagel, H.-H.: Steps Towards a Cognitive Vision System. In: *AI Magazine* 25 (2004), S. 31–50 3, 5

[NPVB05] Nguyen, Nam T. ; Phung, Dinh Q. ; Venkatesh, Svetha ; Bui, Hung: Learning and Detecting Activities from Movement Trajectories Using the Hierarchical Hidden Markov Models. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02.* Washington, DC, USA : IEEE Computer Society, 2005 (CVPR '05). – ISBN 0–7695–2372–2, S. 955–960 7

[NZH03] Nevatia, Ram ; Zhao, Tao ; Hongeng, Somboon: Hierarchical Language-based Representation of Events in Video Streams. In: *Computer Vision and Pattern Recognition Workshop* 4 (2003), S. 39. – ISSN 1063–6919 8, 10

[OHP+11] Oh, Sangmin ; Hoogs, Anthony ; Perera, Amitha ; Cuntoor, Naresh ; Chen, Chia-Chih ; Lee, Jong T. ; Mukherjee, Saurajit ; Aggarwal, J. K. ; Lee, Hyungtae ; Davis, Larry ; Swears, Eran ; Wang, Xiaoyang ; Ji, Qiang ; Reddy, Kishore ; Shah, Mubarak ; Vondrick, Carl ; Pirsiavash, Hamed ; Ramanan, Deva ; Yuen, Jenny ; Torralba, Antonio ; Song, Bi ; Fong, Anesco ; Roy-Chowdhury, Amit ; Desai, Mita: A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video. In: *2011 8th IEEE International Conference on Advanced Video and Signal-Based Surveillancei (AVSS)*, 2011, S. 2 45, 47, 48, 51, 52, 53

[ON03] Ottlik, Artur ; Nagel, Hans-Hellmut: On Consistent Discrimination between Directed and Diffuse Outdoor Illumination. In: *Pattern Recognition* Bd. 2781. Springer Berlin / Heidelberg, 2003. – ISBN 978–3–540–40861–1, S. 418–425 3

[ON08] Ottlik, A. ; Nagel, H.-H.: Initialization of Model-Based Vehicle Tracking in Video Sequences of Inner-City Intersections. In: *International Journal of Computer Vision* 80 (2008), S. 211–225. – ISSN 0920–5691 3

[PB98] Pinhanez, Claudio ; Bobick, Aaron: Human Action Detection Using PNF Propagation of Temporal Constraints. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 0 (1998), S. 898. – ISSN 1063–6919 8, 9, 58

[PBC+08] Patino, Luis ; Benhadda, Hamid ; Corvee, Etienne ; Brémond, François ; Thonnat, Monique: Extraction of Activity Patterns on large Video Recordings. In: *IET Computer Vision* 2 (2008), Juni, Nr. 2, S. 108–128 10

[Pea88] Pearl, Judea: *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1988. – ISBN 0–934613–73–7 11

[Pea94] Pearl, Judea: Belief networks revisited. In: *Artificial intelligence in perspective* (1994), S. 49–56. ISBN 0–262–52186–5 11

[Pól54] Pólya, George: *Mathematics and Plausible Reasoning: Patterns of plausible inference*. Princeton University Press, 1954 (Mathematics and Plausible Reasoning) 11

[ROP99] Rosario, Barbara ; Oliver, Nuria ; Pentland, Alex: A synthetic agent system for Bayesian modeling of human interactions. In: *Proceedings of the third annual conference on Autonomous Agents*. New York, NY, USA : ACM, 1999 (AGENTS '99). – ISBN 1–58113–066–X, S. 342–343 6, 7

[Sch96] Schäfer, Karl H.: *Dissertationen zur Künstlichen Intelligenz (DISKI)*. Bd. 135: *Unscharfe zeitlogische Modellierung von Situationen und Handlungen in Bildfolgenauswertung und Robotik*. Sankt Augustin : infix, 1996. – ISBN 3–89601–135–9 2, 5, 8, 12, 15, 22

[Sha76] Shafer, Glenn: *A mathematical theory of evidence*. Princeton University Press (Princeton, N.J.), 1976. – 297 S. 11, 30

[SHM+04] Shi, Yifan ; Huang, Yan ; Minnen, David ; Bobick, Aaron ; Essa, Irfan: Propagation Networks for Recognition of Partially Ordered Sequential Action. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 2 (2004), S. 862–869. – ISSN 1063–6919 7

[VBT03] Vu, Van T. ; Brémond, François ; Thonnat, Monique: Automatic Video Interpretation: A Recognition Algorithm for Temporal Scenarios Based on Pre-compiled Scenario Models. In: *Computer Vision Systems* Bd. 2626.

Springer Berlin / Heidelberg, 2003. – ISBN 978–3–540–00921–4, S. 523–533 10, 58

[Wei96] Weisbrod, Joachim: *DISKI, Dissertationen zur künstlichen Intelligenz*. Bd. 117: *Unscharfes Schließen*. Sankt Augustin : Infix, 1996. – ISBN 3896011170. – Zugl.: Karlsruhe, Univ., Diss., 1995 2, 14, 20, 22, 23, 24, 25, 27, 29

[WSTL10] Wielemaker, Jan ; Schrijvers, Tom ; Triska, Markus ; Lager, Torbjörn: SWI-Prolog. In: *CoRR* abs/1011.5332 (2010) 33

[YKF94] Yager, Ronald R. (Hrsg.) ; Kacprzyk, Janusz (Hrsg.) ; Fedrizzi, Mario (Hrsg.): *Advances in the Dempster-Shafer theory of evidence*. New York, NY, USA : John Wiley & Sons, Inc., 1994. – ISBN 0–471–55248–8 11

[Zad65] Zadeh, Lotfi A.: Fuzzy sets. In: *Information and Control* 8 (1965), Nr. 3, S. 338 – 353. – ISSN 0019–9958 11

[Zad75] Zadeh, Lotfi A.: Fuzzy logic and approximate reasoning. In: *Synthese* 30 (1975), S. 407–428. – ISSN 0039–7857 11

[Zim01] Zimmermann, H.J.: *Fuzzy set theory-and its applications*. Kluwer Academic Publishers, 2001. – ISBN 9780792374350 2, 20, 22, 125

Anhang A

Schaubilder

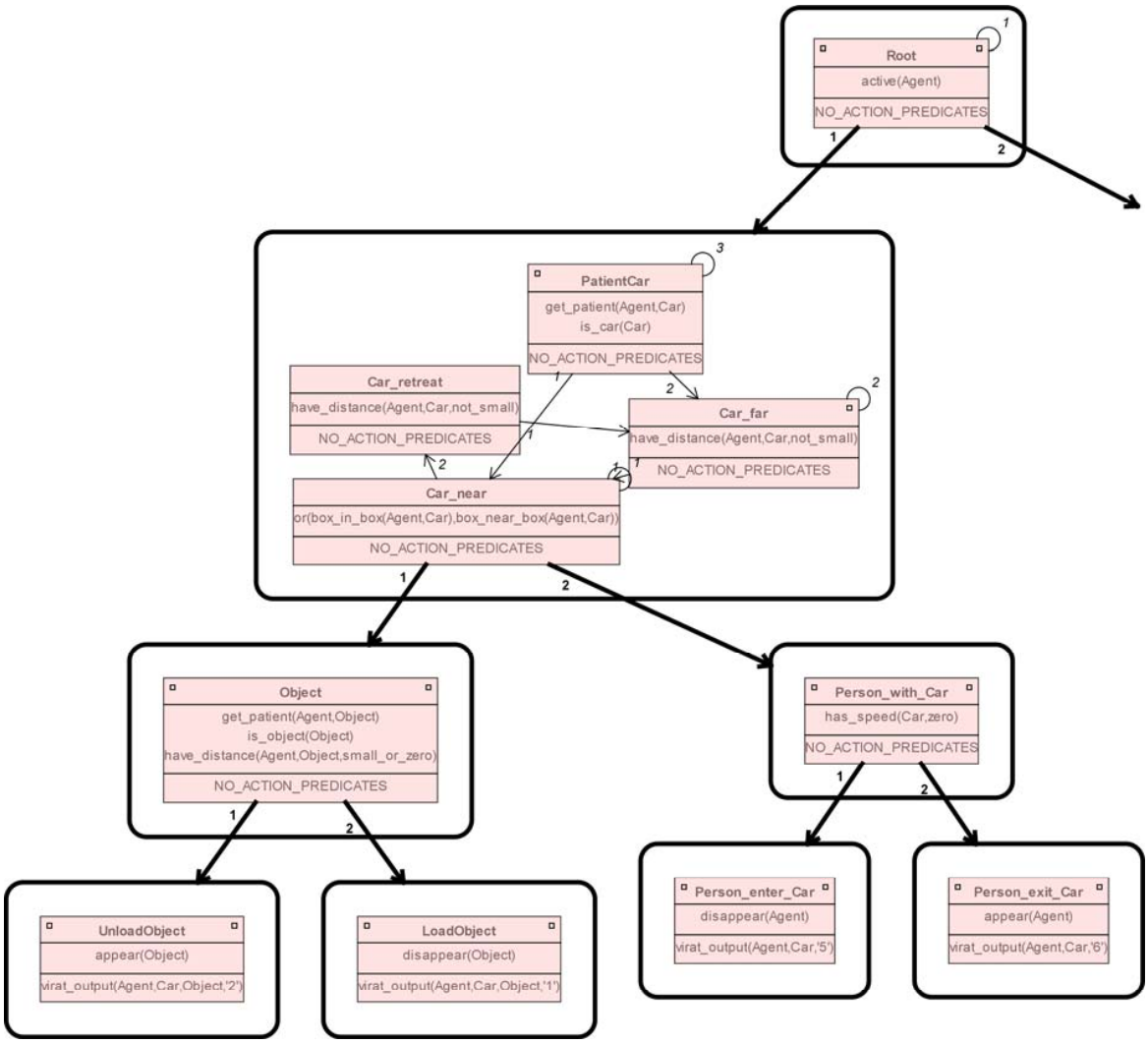

Abbildung A.1: In den Experimenten verwendeter SGT. Linke Hälfte, die Interaktion des Agenten mit einem Fahrzeug behandelt.

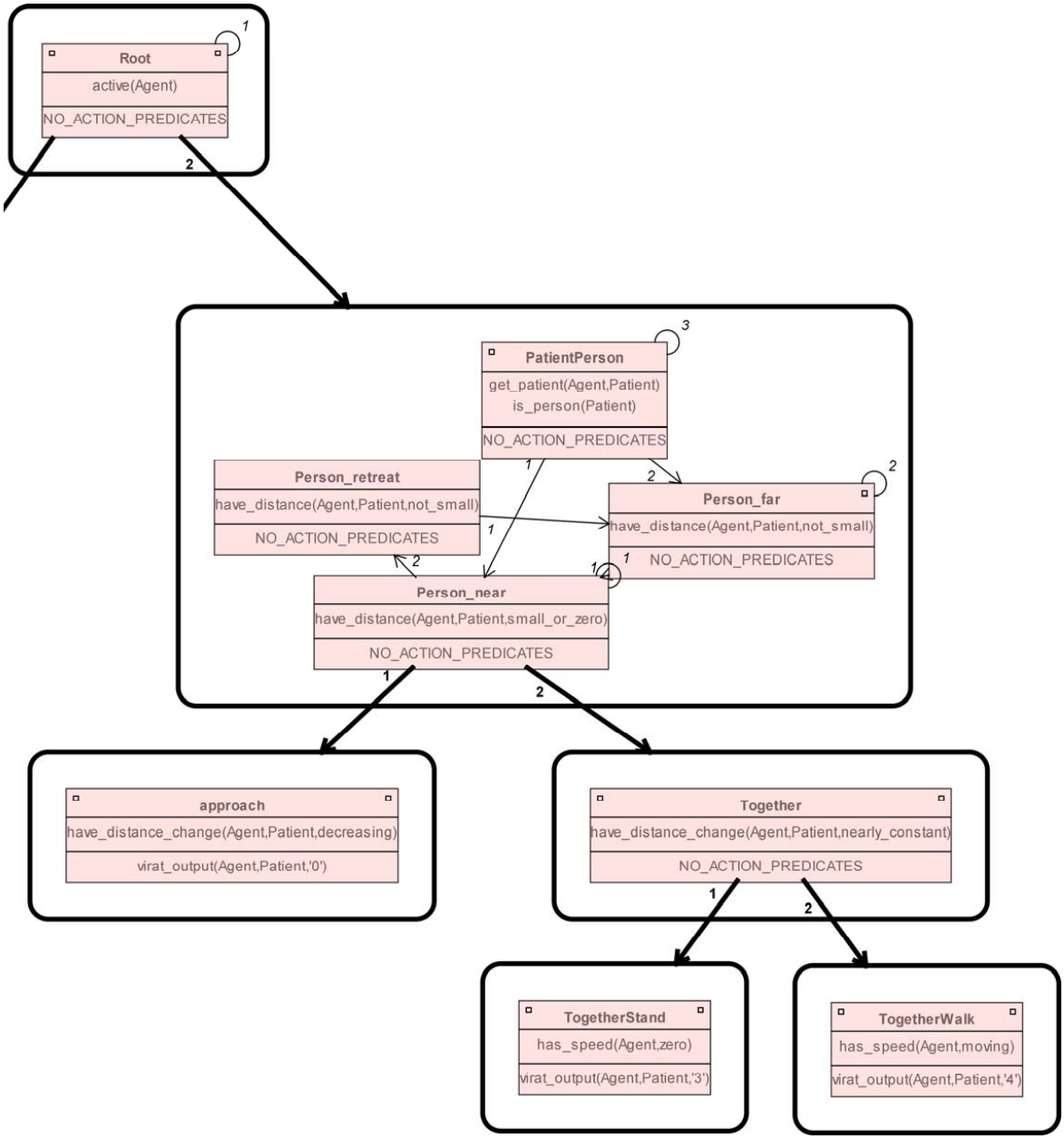

Abbildung A.2: In den Experimenten verwendeter SGT. Rechte Hälfte, die Interaktion des Agenten mit einer anderen Person auswertet.

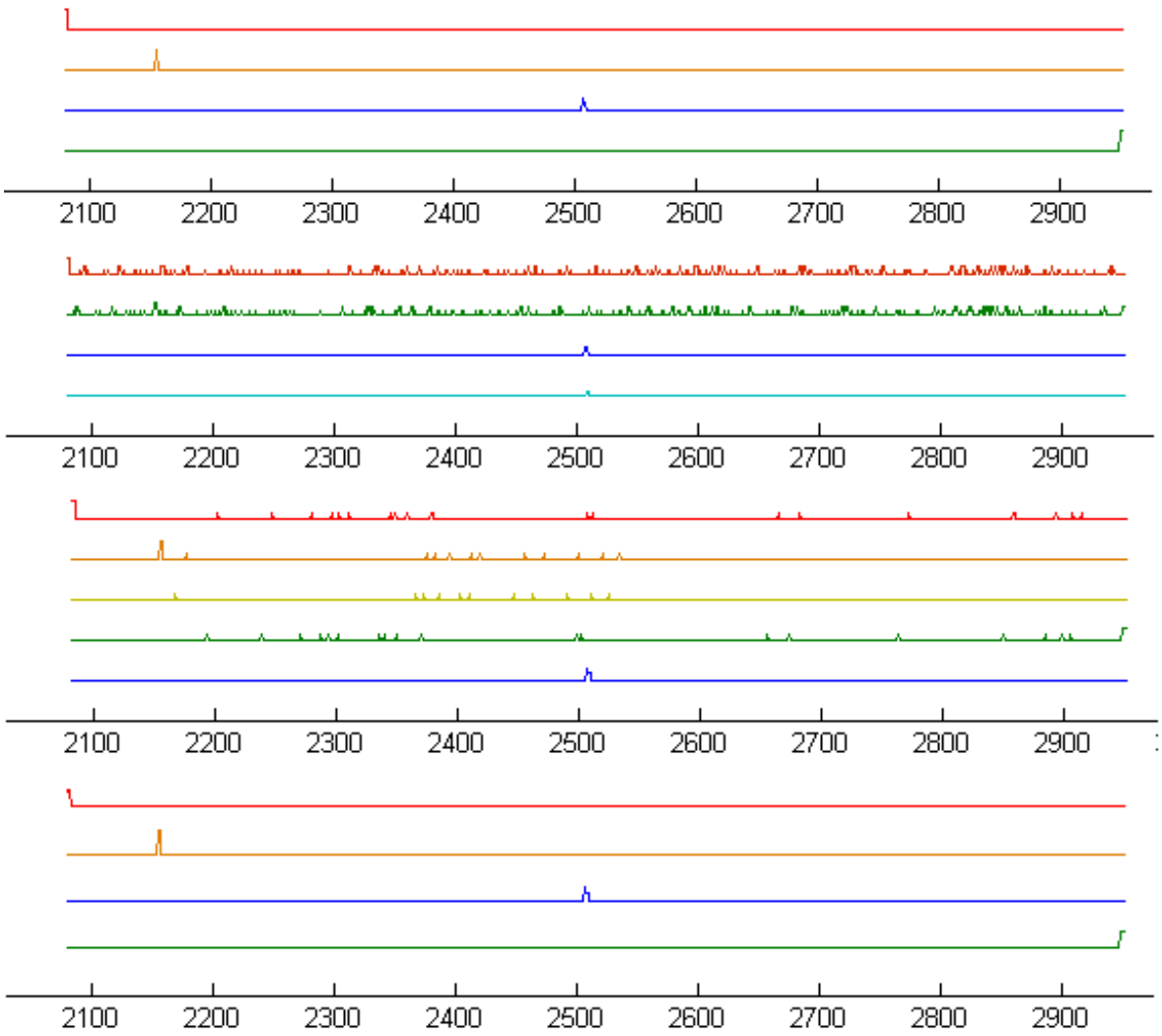

Abbildung A.3: Übersicht zu Video 06 des Ortes 0002: Ausgabe mit perfekten Daten, Ausgabe der ersten Version, Ausgabe mit verlängertem SUBST_INTERVAL und zwei Zeitpunkte langem Test auf Nichtvorhandensein und Ausgabe mit zusätzlich strong UND und NOT in den Prädikaten die Auftauchen und Verschwinden behandeln.

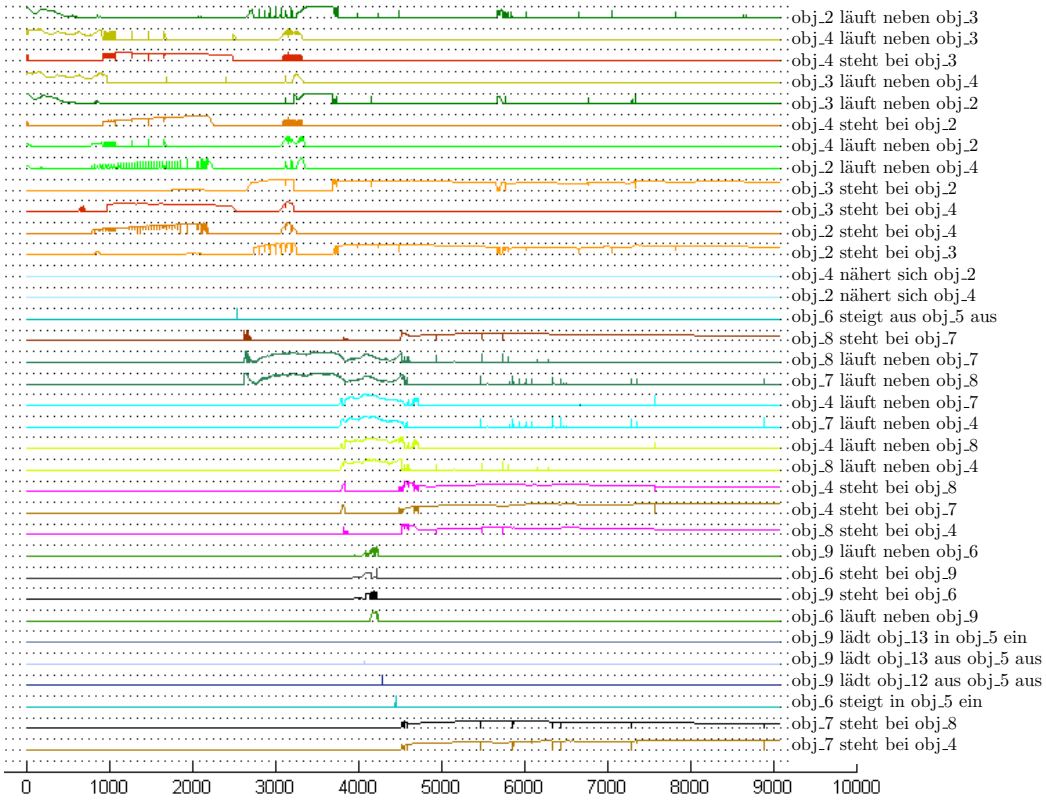

Abbildung A.4: Schaubild zu Video 02 des Ortes 0000: Auswertung von perfekten Daten.

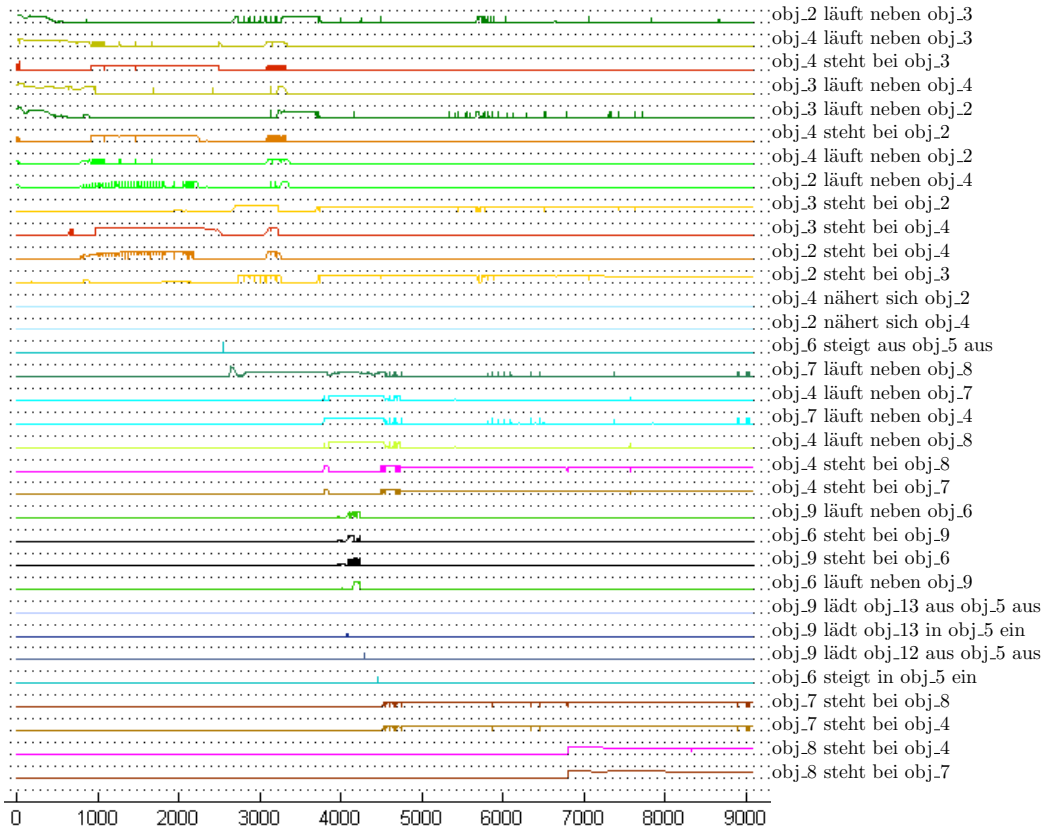

Abbildung A.5: Schaubild eines Ergebnisses zu Video 02 des Ortes 0000: Auswertung von unvollständigen Daten mit der aktuellsten Version.

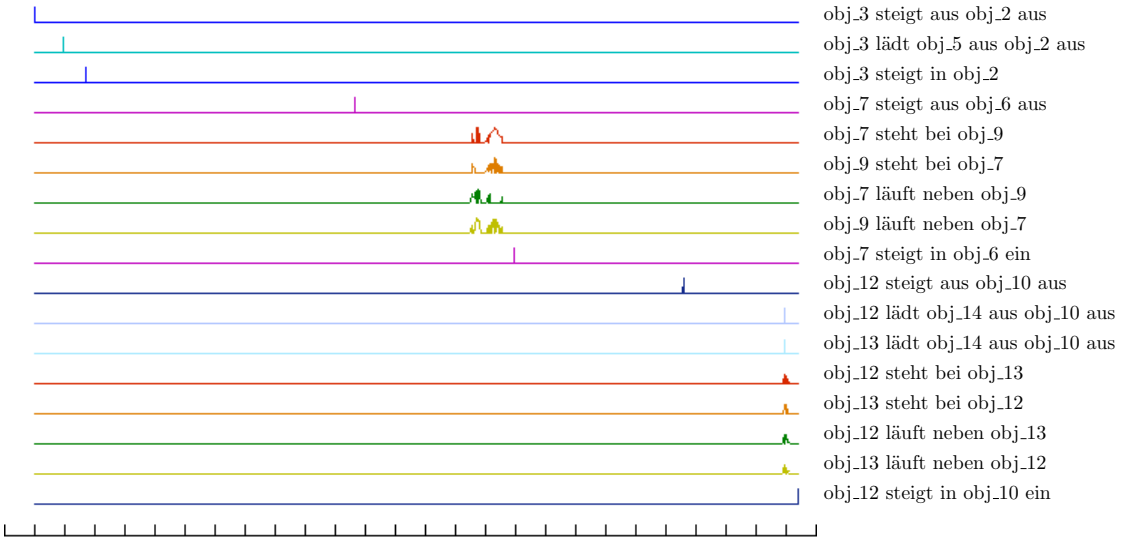

Abbildung A.6: Schaubild zu Video 03 des Ortes 0000: Auswertung von perfekten Daten.

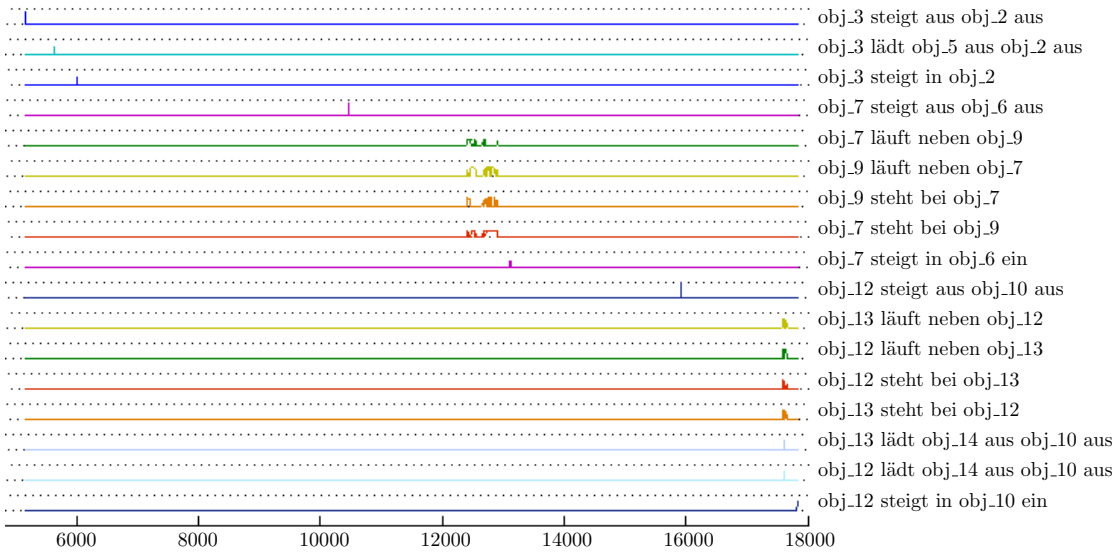

Abbildung A.7: Schaubild eines Ergebnisses zu Video 03 des Ortes 0000: Auswertung von unvollständigen Daten mit der aktuellsten Version.

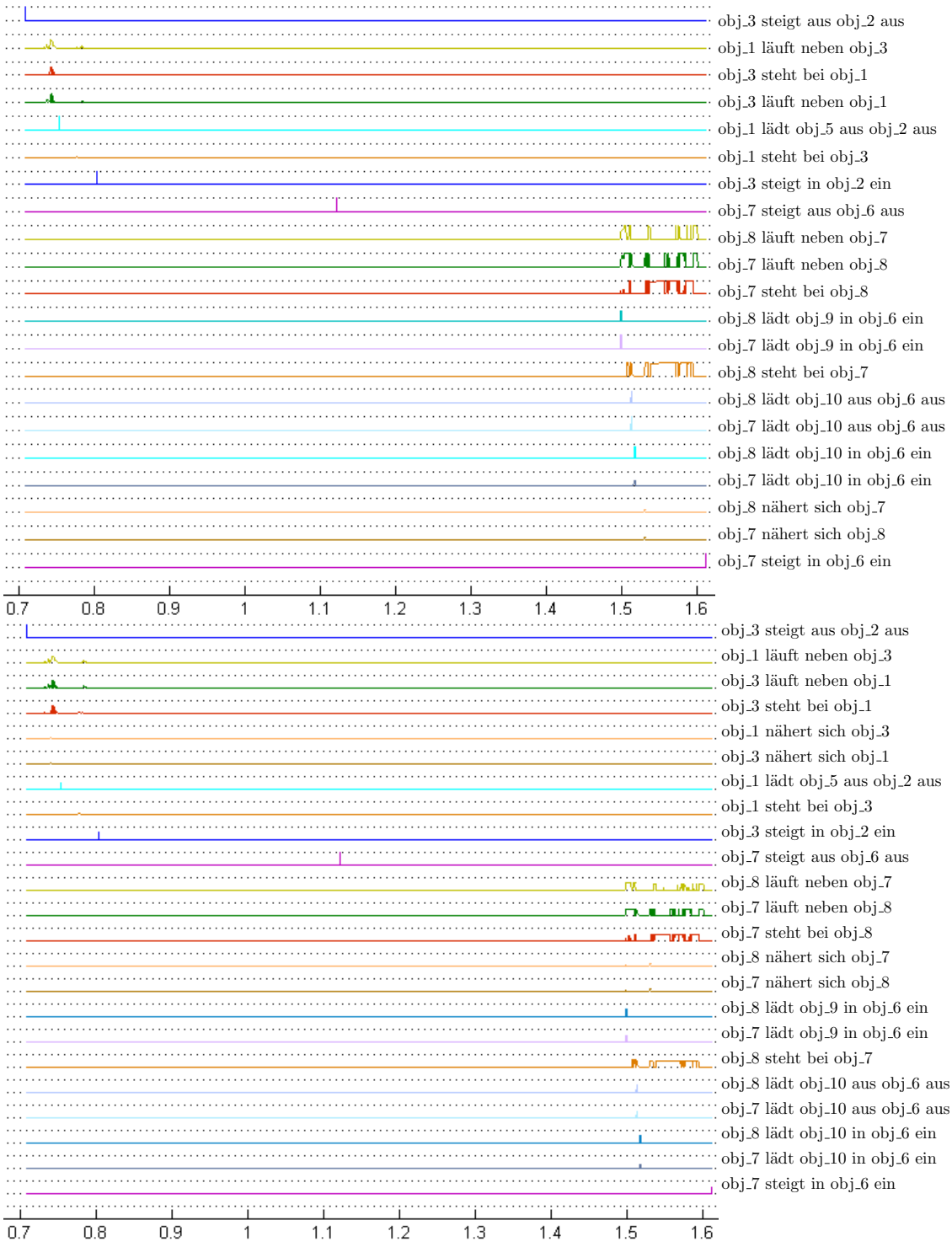

Abbildung A.8: Schaubild zu Video 04 des Ortes 0000: Oben: Auswertung von perfekten Daten. Unten: Auswertung von unvollständigen Daten mit der aktuellsten Version.

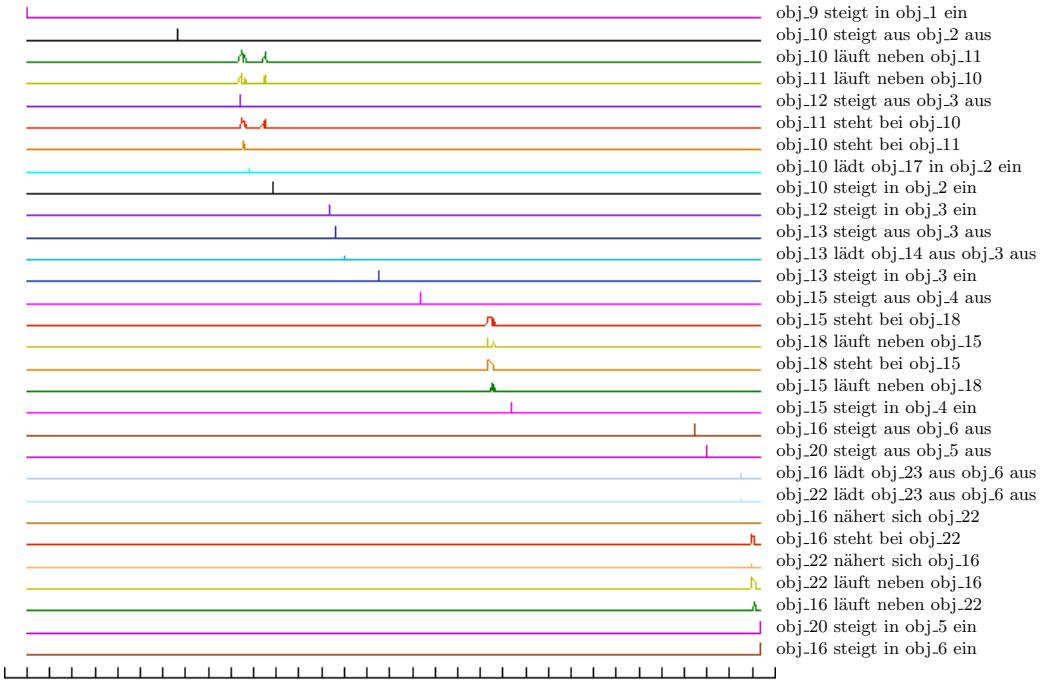

Abbildung A.9: Schaubild zu Video 06 des Ortes 0000: Auswertung von perfekten Daten.

Abbildung A.10: Schaubild eines Ergebnisses zu Video 06 des Ortes 0000: Auswertung von unvollständigen Daten mit der aktuellsten Version.

Anhang B

F-LIMETTE Regeldateien

B.1 Testregeln

```
always(nloutput(Var1,Var2,Var3) :-
       append('log_test.lim')
       , nl
       , write(Var1)
       , write(Var2)
       , write(Var3)
       , nl
       , told
       , nl
       , write(Var1)
       , write(Var2)
       , write(Var3)
       , nl
).

always(output(Var1,Var2,Var3) :-
       append('log_test.lim')
       , write(Var1)
       , write(Var2)
       , write(Var3)
       , nl
       , told
       , write(Var1)
       , write(Var2)
       , write(Var3)
       , nl
).

always(test_one(Name) :-
       test(Name)
       , output(Name,': OK','')
       , fail
).

always(test_all :-
       test_one(X)
).

always(pre_test(Name,Desc) :-
       nloutput(Name,': test ',Desc)
).

always(is_testing).
```

B.2 Basisregeln

```
//------------------------------------------------------------------------
// Tag  Meaning
// +    argument is expected to be instantiated
//      (i.e. have a value rather than be a variable)
// -    argument is expected to be a variable to be bound
// ?    argument may be either instantiated, or be an unbound variable,
// when the procedure is called as a goal.
//------------------------------------------------------------------------

// or(+A, +B)
// positive_derivative(+A, +B, +C, +D, +E)
// derivative(+A, +B, +C, +D, +E, ?Deriv)
// degreeOfValidity(+X, +P1, +P2, +P3, +P4)
// length(+A, +B, ?L)
// max(+A, +B, ?M)
// min(+A, +B, ?M)
// abs(+Number, ?Abs)
// epsilon_neighborhood(+Epsilon, +Val1, +Val2)

// box_geom(+X1, +Y1, +W1, +H1, +X2, +Y2, +W2, +H2, ?X, ?Y, ?W, ?H)
// box_obj(+Agent, +Patient, ?X, ?Y, ?W, ?H)
// box_in_box(+X1, +Y1, +W1, +H1, +X2, +Y2, +W2, +H2, ?DoV)
// box_in_box(+Tiny, +Huge)
// box_near_box(+Tiny, +Huge)
// has_ground_geometry(+Agent, ?X, ?Y)
// has_geometry(+Agent, ?X, ?Y, ?W, ?H)

// appear(+Agent)
// disappear(+Agent)

// distance_is(+Agent, +Patient, ?Distance)
// have_distance(+Agent, +Patient, ?Value)
// associate_distance(+Distance, ?Value)
// have_distance_change(+Agent, +Patient, ?Value)
// associate_distance_change(+Deriv, ?Value)

// has_speed(+Agent, ?Value)
// associate_speed(+V, ?Value)

// atan2(+Y, +X, ?Radians)
// convert_angle(+Radians, ?Degrees)
// convert_angle(?Radians, +Degrees)
// ang_direction(+X, +Y, ?Angle)
// ang_norm(+Angle, ?Norm)

// move_direction_is(+Agent, ?Dir)
// distance_direction_is(+Agent, +Patient, ?Dir)

// associate_angle(+Dir1, +Dir2, ?Value)
// associate_angle(+Angle, ?Value)

// relative_position_is(+Agent, +Patient, ?Angle)
// have_relative_position(+Agent, +Patient, ?Value)
// associate_relative_position(+Angle, ?Value)
// have_view_angle(+Agent, +Patient, ?Value)
// have_configuration(+Agent, +Patient, ?Value)
// associate_configuration(+Angle, ?Value)

// have_move_angle(+Agent, +Patient, ?Value)
```

```
// have_course(+Agent, +Patient, ?Value)
// course_is(+Agent, +Patient, ?Course)
// associate_course(+Course, ?Value)
// have_course_change(+Agent, +Patient, ?Value)
// associate_course_change(+Deriv, ?Value)

// have_difference_in_orientation(+Agent, +Patient, ?Value)
// difference_in_orientation_is(+Agent, +Patient, ?Diff)
// associate_difference_in_orientation(+Diff, ?Value)
// have_difference_in_orientation_change(+Agent, +Patient, ?Value)
// associate_difference_in_orientation_change(+Deriv, ?Value)

// has_direction(+Agent, ?Value)
// associate_direction(+A, ?Value)

// ============================================================================
// Supplementary Rules
// ============================================================================

// strong or for call from SGTEditor
always( or(A, B) :-
        A
        ;@S B
).

//5-ary predicate symbol. True, if the derivation of 5 points given is positive.
always( positive_derivative(A, B, C, D, E) :-
        Derivative is (-2.0*A)-B+D+(2.0*E)
        , Derivative > 0
).

always( derivative(A, B, C, D, E, Deriv) :-
        Deriv is ( (-2.0*A) - B + D + (2.0*E) )/3
).

always( degreeOfValidity(X, P1, P2, P3, P4) :-
        X >= P1 , X < P2 , Wert is (X - P1) / (P2 - P1) , sp(Wert)
        ; X >= P2 , X < P3 , sp(1.0)
        ; X >= P3 , X < P4 , Wert is (P4 - X) / (P4 - P3) , sp(Wert)
).

always( length(A, B, L) :-
        L is sqrt((A * A) + (B * B))
).

always( max(A, B, M) :-
        A > B
          , M is A
          , !
        ; M is B
).

always( min(A, B, M) :-
        A < B
          , M is A
          , !
        ; M is B
).

always( abs(Number, Abs) :-
        Number < 0
          , Abs is -1.0*Number
          , !
        ; Abs is Number
```

```
).

always( epsilon_neighborhood(Epsilon, Val1, Val2) :-
          Val2 >= Val1
          , Val2 - Val1 < Epsilon
        ; Val1 > Val2
          , Val1 - Val2 < Epsilon
).

always( box_geom(X1, Y1, W1, H1, X2, Y2, W2, H2, X, Y, W, H) :-
        min(X1, X2, X)
        , min(Y1, Y2, Y)
        , max(X1 + W1, X2 + W2, X_)
        , max(Y1 + H1, Y2 + H2, Y_)
        , W is X_ - X
        , H is Y_ - Y
).

always( box_obj(Agent, Patient, X, Y, W, H) :-
        has_image_geometry(Agent, X1, Y1, W1, H1)
        , has_image_geometry(Patient, X2, Y2, W2, H2)
        , box_geom(X1, Y1, W1, H1, X2, Y2, W2, H2, X, Y, W, H)
).

// +---------+
// |      +-+|
// |      +.+|
// |      /  |
// |     /   |
// |____/____|
// has_ground_geometry not usefull for small distance between
// huge and tiny object
// no need to use projectiveTransform since this only concern %-values

always( box_in_box(X1, Y1, W1, H1, X2, Y2, W2, H2, DoV) :-
          X1_ is X1 + W1
        , Y1_ is Y1 + H1
        , X2_ is X2 + W2
        , Y2_ is Y2 + H2
        , max(X1, X2, XL)
        , min(X1_, X2_, XR)
        , W_ is XR - XL
        , W_ > 0
        , max(Y1, Y2, YO)
        , min(Y1_, Y2_, YU)
        , H_ is YU - YO
        , H_ > 0
        , N is 1.0 * W_ * H_
        , Z is 1.0 * W1 * H1
        , DoV is N/Z
).
always( box_in_box(Tiny, Huge) :-
        has_image_geometry(Tiny, X1, Y1, W1, H1)
        , has_image_geometry(Huge, X2, Y2, W2, H2)
        , box_in_box(X1, Y1, W1, H1, X2, Y2, W2, H2, DoV)
        //, note(box_in_box+Tiny+Huge+DoV)
        , sp(DoV)
).

// problem tiny object next to huge object
// +---------+
// |         |
// |         |+-+
// |         || |
// |         |+-+
```

```
// +---------+
// => use box_in_box with bloated tiny object:
//   w   w   w
// N---O---+---+ --         O: X1, Y1
// |   |   |   |            N: X1e, Y1e
// |   |   |   | H
// |   +---+   | --
// |           | H/2
// +-----------+ --
always( box_near_box(Tiny, Huge) :-
        has_image_geometry(Tiny, X1, Y1, W1, H1)
        , has_image_geometry(Huge, X2, Y2, W2, H2)
        , X1e is X1 - W1
        , Y1e is Y1
        , W1e is 3 * W1
        , H1e is 1.5 * H1
        , box_in_box(X1e, Y1e, W1e, H1e, X2, Y2, W2, H2, R_)
        , box_in_box(X1, Y1, W1, H1, X2, Y2, W2, H2, Ri)
        , Rn is R_ * 3
        , max(Rn,Ri,R)
        , min(R,1,D)
        , DoV is 1.0 * D
        //, note(box_near_box+Tiny+Huge+R_+Ri+DoV)
        , sp(DoV)
).

// to determine e.g. speed or direction
// use bottom center of bounding box in world geometry
// +-> X
// | +---+ -
// v |_O_| |
// Y | | | H
//   |/ \| |
//   +-*-+ -   *= (X+W/2, Y+H)
//   |-W-|

always( has_ground_geometry(Agent, Xp, Yp) :-
        has_image_geometry(Agent, Xi, Yi, W, H)
        , X is Xi + W/2
        , Y is Yi + H
        , projectiveTransform(X, Y, Xp, Yp)
).

always( has_geometry(Agent, X, Y, W, H) :-
        has_image_geometry(Agent, Xi, Yi, Wi, Hi)
        , projectiveTransform(Xi, Yi, X, Y)
        , projectiveTransform(Xi+Wi, Yi, Xw, Yw)
        , projectiveTransform(Xi, Yi+Hi, Xh, Yh)
        , length(Xw-Xi, Yw-Yi, W)
        , length(Xh-Xi, Yh-Yi, H)
).

// ============================================================================
// Presence
// ============================================================================

// problem with: not ( -1 ! has_image_geometry(Agent, _, _, _, _))
// => extra predicate without unknown variables
always( has_data(Agent) :-
        has_image_geometry(Agent, _, _, _, _)
).

// Strong AND and NOT: ,@S and not@S
always( appear(Agent) :-
        not@S (-5 ! has_data(Agent))
```

```
        ,@S not@S (-4 ! has_data(Agent))
        ,@S 0 ! has_image_geometry(Agent, _, _, _, _)
        ,@S 1 ! has_image_geometry(Agent, _, _, _, _)
        //, note('appear'+Agent)
).

// Strong AND and NOT: ,@S and not@S
always( disappear(Agent) :-
         -1 ! has_image_geometry(Agent, _, _, _, _)
        ,@S 0 ! has_image_geometry(Agent, _, _, _, _)
        ,@S not@S (4 ! has_data(Agent))
        ,@S not@S (5 ! has_data(Agent))
        //, note('disappear'+Agent)
).

// ========================================================================
// Distance
// ========================================================================

always( distance_is(Agent,Patient,Distance) :-
        has_ground_geometry(Agent,X,Y)
        , has_ground_geometry(Patient,X0,Y0)
        //, note('has_ground_geometry'+Agent+X+Y)
        //, note('has_ground_geometry'+Patient+X0+Y0)
        , length(X-X0,Y-Y0,Distance)
).

always( have_distance(Agent,Patient,Value) :-
        distance_is(Agent,Patient,Distance)
        , get_spatial_coefficient(Co)
        , Distance_ is (Co * Distance)
        //, note('have_distance'+Agent+Patient+Distance_)
        , associate_distance(Distance_,Value)
        //, (A1{associate_distance(Distance_,Value)}B1)
        //, header_dov(B1, Value, not_small, dist)
).

always( associate_distance(Distance,Value) :-
        degreeOfValidity(Distance, -5.0, -5.0, 1.0, 5.0)
          , Value = zero//, note(zero)
        ; degreeOfValidity(Distance, 1.0, 5.0, 70.0, 170.0)
          , Value = small//, note(small)
        ; degreeOfValidity(Distance, -5.0, -5.0, 70.0, 170.0)
          , Value = small_or_zero//, note(small_or_zero)
        ; degreeOfValidity(Distance, 70.0, 170.0, 262.5, 367.5)
          , Value = normal//, note(normal)
        ; degreeOfValidity(Distance, 262.5, 367.5, 10000.0, 10000.0)
          , Value = high//, note(high)
        ; degreeOfValidity(Distance, 1.5, 3.5, 10000.0, 10000.0)
          , Value = not_zero//, note(not_zero)
        ; degreeOfValidity(Distance, 105.0, 157.5, 10000.0, 10000.0)
          , Value = not_small//, note(not_small)
).

always( have_distance_change(Agent,Patient,Value) :-
          -2 ! distance_is(Agent,Patient,Distance_2)
        , -1 ! distance_is(Agent,Patient,Distance_1)
        ,  0 ! distance_is(Agent,Patient,Distance)
        ,  1 ! distance_is(Agent,Patient,Distance1)
        ,  2 ! distance_is(Agent,Patient,Distance2)
        , derivative(Distance_2,Distance_1,Distance,Distance1,Distance2,Deriv)
        , get_spatial_coefficient(Co)
        , Deriv_ is (Co * Deriv)
        //, note('have_distance_change'+Agent+Patient+Deriv_)
        , associate_distance_change(Deriv_,Value)
```

```
        //, (A1{associate_distance_change(Deriv_,Value)}B1)
        //, header_dov(B1, Value, increasing, dist_ch)
).

always( associate_distance_change(Deriv,Value) :-
        degreeOfValidity(Deriv, -7.0, -3.0, 3.0, 7.0)
          , Value = constant//, note(constant)
        ; degreeOfValidity(Deriv, -35.0, -14.0, -7.0, -2.0)
          , Value = slightly_decreasing//, note(slightly_decreasing)
        ; degreeOfValidity(Deriv, -10000.0, -10000.0, -33.0, -17.0)
          , Value = significant_decreasing//, note(significant_decreasing)
        ; degreeOfValidity(Deriv, -10000.0, -10000.0, -60.0, -15.0)
          , Value = decreasing//, note(decreasing)
        ; degreeOfValidity(Deriv, 2.0, 7.0, 14.0, 35.0)
          , Value = slightly_increasing//, note(slightly_increasing)
        ; degreeOfValidity(Deriv, 17.0, 33.0, 10000.0, 10000.0)
          , Value = significant_increasing//, note(significant_increasing)
        ; degreeOfValidity(Deriv, 15.0, 60.0, 10000.0, 10000.0)
          , Value = increasing//, note(increasing)
        ; degreeOfValidity(Deriv, -150.0, -60.0, 60.0, 150.0)
          , Value = nearly_constant//, note(nearly_constant)
).

// ============================================================================
// Speed
// ============================================================================
// -3\
// -2-* +
// -1/  |  Distance_210 (2 frames) \
//  0   ++                          > + / 4 = Distance between * per frame
//  1\   | Distance012 (2 frames)  /
//  2-*  +
//  3/

always( has_speed(Agent,Value) :-
          -3 ! has_ground_geometry(Agent, X_3, Y_3)
        , -2 ! has_ground_geometry(Agent, X_2, Y_2)
        , -1 ! has_ground_geometry(Agent, X_1, Y_1)
        ,  0 ! has_ground_geometry(Agent, X0, Y0)
        ,  1 ! has_ground_geometry(Agent, X1, Y1)
        ,  2 ! has_ground_geometry(Agent, X2, Y2)
        ,  3 ! has_ground_geometry(Agent, X3, Y3)
        , length(((1.0*X_3)+X_2+X_1)/3-X0, ((1.0*Y_3)+Y_2+Y_1)/3-Y0,
                 Distance_210)
        , length(((1.0*X3)+X2+X1)/3-X0, ((1.0*Y3)+Y2+Y1)/3-Y0, Distance012)
        , get_spatial_coefficient(Sco)
        , Distance is Sco * ((Distance_210+Distance012)/4)
        , get_time_coefficient(Tco)
        // Speed in cm/sec
        , Speed is Tco * Distance
        //, note(has_speed+Agent+Speed)
        , associate_speed(Speed,Value)
).

// cm per sec
// average pedestrian speed: Trier: 138 cm/sec Dresden: 149 cm/sec
// 5km Stadtwerke walking 50. women 173 cm/sec

// 5km Stadtwerke run    100. men   234 cm/sec
// 5km Stadtwerke walking  1. women 239 cm/sec
// 5km Stadtwerke walking  1. men   253 cm/sec

// 5km Stadtwerke run     50. men   319 cm/sec
// german Record women 20km walking:379 cm/sec
```

```
always( associate_speed(V,Value) :-
        degreeOfValidity(V, -10.0, -5.0, 5.0, 10.0)
          , Value = zero//, note(zero)
        ; degreeOfValidity(V, 5.0, 30.0, 50.0, 80.0)
          , Value = very_small//, note(very_small)
        ; degreeOfValidity(V, 70.0, 80.0, 100.0, 120.0)
          , Value = small//, note(small)
        ; degreeOfValidity(V, 115.0, 135.0, 155.0, 175.0)
          , Value = normal//, note(normal)
        ; degreeOfValidity(V, 150.0, 170.0, 290.0, 310.0)
          , Value = high//, note(high)
        ; degreeOfValidity(V, 300.0, 320.0, 10000.0, 10000.0)
          , Value = very_high//, note(very_high)
        ; degreeOfValidity(V, 5.0, 10.0, 10000.0, 10000.0)
          , Value = moving//, note(moving)
        ; degreeOfValidity(V, -10000.0, -10000.0, -10.0, -5.0)
          , Value = backwards//, note(backwards)
).

// ============================================================================
// Direction
// ============================================================================

// -1/2 * Pi       -90    +---> x
//     |            |     |
// Pi -*- 0    180 -*- 0  |
//     |            |     V y
//  1/2 * Pi        90


always( atan2(Y, X, R) :-
          X >  0,           Q is (1.0*Y)/(1.0*X), R is atan(Q)
        ; Y >= 0, X <   0,  Q is (1.0*Y)/(1.0*X), R is atan(Q) + 3.1415926
        ; Y <  0, X <   0,  Q is (1.0*Y)/(1.0*X), R is atan(Q) - 3.1415926
        ; Y >  0, X =:= 0,  R is 3.1415926/2
        ; Y <  0, X =:= 0,  R is -3.1415926/2
).

always( convert_angle(Radians, Degrees) :-
        nonvar(Radians)
          , Degrees is (Radians*360/6.2831853)
        ; nonvar(Degrees)
          , Radians is (Degrees*6.2831853/360)
).

always( ang_direction(X, Y, Ang) :-
        atan2(Y, X, Rad)
        , convert_angle(Rad, Ang)
).

// Difference between two angles Ang has Values between -360 and 360
// Returnvalue Norm should be between -180 and 180

always( ang_norm(Ang, Norm) :-
          Ang > 180
          , Norm is Ang - 360
          , !
        ; Ang < -180
          , Norm is Ang + 360
          , !
        ; Norm is Ang
).

always( move_direction_is(Agent,Dir) :-
          -3 ! has_ground_geometry(Agent,X_3,Y_3)
```

```
        , -2 ! has_ground_geometry(Agent,X_2,Y_2)
        , -1 ! has_ground_geometry(Agent,X_1,Y_1)
        ,  1 ! has_ground_geometry(Agent,X1,Y1)
        ,  2 ! has_ground_geometry(Agent,X2,Y2)
        ,  3 ! has_ground_geometry(Agent,X3,Y3)
        , Y is Y3 + Y2 + Y1 - Y_1 - Y_2 - Y_3
        , X is X3 + X2 + X1 - X_1 - X_2 - X_3
        , atan2(Y, X, Rad)
        , convert_angle(Rad, Dir)
).

always( distance_direction_is(Agent,Patient,Dir) :-
        has_ground_geometry(Agent,X1,Y1)
        , has_ground_geometry(Patient,X2,Y2)
        , Y is (Y2 - Y1)
        , X is (X2 - X1)
        , atan2(Y, X, Rad)
        , convert_angle(Rad, Dir)
).

// ------------------------------------------------------------------------
// Give direction of Dir2 relative to Dir1

always( associate_angle(Dir1, Dir2, Value) :-
        A is Dir2 - Dir1
        , ang_norm(A, Norm)
        , associate_angle(Norm, Value)
).

always( associate_angle(A, Value) :-
        degreeOfValidity(A, -30.0, -20.0,  20.0,  30.0) , Value = ahead ;
        degreeOfValidity(A,  15.0,  30.0,  60.0,  75.0) , Value = right_ahead ;
        degreeOfValidity(A, -75.0, -60.0, -30.0, -15.0) , Value = left_ahead ;
        degreeOfValidity(A,  40.0,  70.0, 110.0, 140.0) , Value = right ;
        degreeOfValidity(A,-140.0,-110.0, -70.0, -40.0) , Value = left ;
        degreeOfValidity(A, 105.0, 120.0, 150.0, 165.0) , Value = right_behind ;
        degreeOfValidity(A,-165.0,-150.0,-120.0,-105.0) , Value = left_behind ;
        degreeOfValidity(A,-180.1,-180.1,-160.0,-150.0) , Value = behind ;
        degreeOfValidity(A, 150.0, 160.0, 180.1, 180.1) , Value = behind
).

// ------------------------------------------------------------------------
// VDA: view direction of Agent
// MDA: move direction of Agent
// DDP: distance direction of Patient
// VDP: view direction of Patient
//
//     | DDP                    | VDP                            | VDA
// ----+------------------------+--------------------------------+-------------
// VDA | have_relative_position | have_difference_in_orientation |
//     | have_view_angle        |                                |
//     | have_configuration     |                                |
// ----+------------------------+--------------------------------+-------------
// MDA | have_move_angle        |                                | has_direction
//     | have_course            |                                |
//     |                        |                                |

// ------------------------------------------------------------------------
// Give direction of Patient relative to view direction of Agent
// Note: this is not symmetric for two Objects * and O
//                      _*_     _O_
// bird's eye view of    |  and  |
//                      / \     / \
//         |
// -*- <-> O   Patient O is right to Agent *
```

```
//         |   Patient * is ahead of Agent O


always( relative_position_is(Agent, Patient, Angle) :-
        has_view_direction(Agent, Dir1)
        , distance_direction_is(Agent, Patient, Dir2)
        , A is Dir2 - Dir1
        , ang_norm(A, Angle)
).

always( have_relative_position(Agent, Patient, Value) :-
        relative_position_is(Agent, Patient, A)
        , associate_relative_position(A, Value)
).

always( associate_relative_position(A, Value) :-
        degreeOfValidity(A, -20.0,  -5.0,   5.0,  20.0) , Value = straight ;
        degreeOfValidity(A,-180.1,-180.1,-175.0,-160.0) , Value = straight ;
        degreeOfValidity(A, 160.0, 175.0, 180.1, 180.1) , Value = straight ;

        degreeOfValidity(A,  10.0,  30.0,  60.0,  80.0) , Value = half_right ;
        degreeOfValidity(A, 100.0, 120.0, 150.0, 170.0) , Value = half_right ;

        degreeOfValidity(A, -80.0, -60.0, -30.0, -10.0) , Value = half_left ;
        degreeOfValidity(A,-170.0,-150.0,-120.0,-100.0) , Value = half_left ;

        degreeOfValidity(A,  40.0,  70.0, 110.0, 140.0) , Value = right ;
        degreeOfValidity(A,-140.0,-110.0, -70.0, -40.0) , Value = left
).

always( have_view_angle(Agent, Patient, Value) :-
        relative_position_is(Agent, Patient, A)
        , associate_angle(A, Value)
).

always( have_configuration(Agent, Patient, Value) :-
        relative_position_is(Agent, Patient, A)
        , associate_configuration(A, Value)
).

always( associate_configuration(A, Value) :-
        degreeOfValidity(A, -90.0, -50.0,  50.0,  90.0) , Value = front ;

        degreeOfValidity(A,-180.1,-180.1,-130.0, -90.0) , Value = behind ;
        degreeOfValidity(A,  90.0, 130.0, 180.1, 180.1) , Value = behind ;

        degreeOfValidity(A,  20.0,  50.0, 130.0, 160.0) , Value = beside ;
        degreeOfValidity(A,-160.0,-130.0, -50.0, -20.0) , Value = beside
).

// ---------------------------------------------------------------------------
// Give direction of Patient relative to move direction of Agent

always( have_move_angle(Agent, Patient, Value) :-
        move_direction_is(Agent, Dir1)
        , distance_direction_is(Agent, Patient, Dir2)
        , associate_angle(Dir1, Dir2, Value)
        //, (A1{associate_angle(Dir1, Dir2, Value)}B1)
        //, header_dov(B1, angle)
).

// ---------------------------------------------------------------------------
// Give direction of standing Patient relative to move direction of Agent

always( have_course(Agent, Patient, Value) :-
```

```
        course_is(Agent, Patient, Course)
        , associate_course(Course, Value)
).

always( course_is(Agent, Patient, Course) :-
        has_speed(Agent, moving)
        , has_speed(Patient, zero)
        , move_direction_is(Agent, Dir1)
        , distance_direction_is(Agent, Patient, Dir2)
        , A is Dir2 - Dir1
        , ang_norm(A, Course)
).

always( associate_course(A, Value) :-
        degreeOfValidity(A, -50.0, -30.0,  30.0,  50.0) , Value = approaching ;
        //degreeOfValidity(A,  30.0,  50.0, 130.0, 150.0) , Value = passing ;
        //degreeOfValidity(A,-150.0,-130.0, -50.0, -30.0) , Value = passing ;
        degreeOfValidity(A,  30.0,  50.0, 130.0, 150.0) , Value = passing_left ;
        degreeOfValidity(A,-150.0,-130.0, -50.0, -30.0) , Value = passing_right ;
        degreeOfValidity(A, 130.0, 150.0, 180.1, 180.1) , Value = leaving ;
        degreeOfValidity(A,-180.1,-180.1,-150.0,-130.0) , Value = leaving
).

// NOTE:
// this function needs -5 to 5 in time (+-2 and +-3 in has_speed/move_direction_is)!

always( have_course_change(Agent, Patient, Value) :-
          -2 ! course_is(Agent, Patient, Course_2)
        , -1 ! course_is(Agent, Patient, Course_1)
        ,  0 ! course_is(Agent, Patient, Course)
        ,  1 ! course_is(Agent, Patient, Course1)
        ,  2 ! course_is(Agent, Patient, Course2)
        , derivative(Course_2, Course_1, Course, Course1, Course2, Deriv)
        , ang_norm(Deriv, Norm)
        , associate_course_change(Norm, Value)
).

always( associate_course_change(Deriv, Value) :-
        degreeOfValidity(Deriv, -13.0,  -5.0,   5.0,  13.0) , Value = constant ;
        degreeOfValidity(Deriv,   5.0,  13.0, 180.1, 180.1) , Value = changing ;
        degreeOfValidity(Deriv,-180.1,-180.1, -13.0,  -5.0) , Value = changing
).

// ------------------------------------------------------------------------
// Difference between view direction of Agent and Patient

always( have_difference_in_orientation(Agent, Patient, Value) :-
        difference_in_orientation_is(Agent, Patient, Diff)
        , associate_difference_in_orientation(Diff, Value)
).

always( difference_in_orientation_is(Agent, Patient, Diff) :-
        has_view_direction(Agent, Dir1)
        , has_view_direction(Patient, Dir2)
        , A is Dir2 - Dir1
        , ang_norm(A, Norm)
        , abs(Norm, Diff)
).

always( associate_difference_in_orientation(Diff, Value) :-
        degreeOfValidity(Diff,  -0.1,  -0.1,  30.0,  50.0) , Value = equal ;
        degreeOfValidity(Diff,  30.0,  50.0, 130.0, 150.0) , Value = crossing ;
        degreeOfValidity(Diff, 130.0, 150.0, 180.1, 180.1) , Value = opposite
).
```

```
// NOTE:
// this function needs -5 to 5 in time if has_view_direction(Patient, Dir2)
// is replaced by move_direction_is!

always( have_difference_in_orientation_change(Agent, Patient, Value) :-
          -2 ! difference_in_orientation_is(Agent, Patient, Diff_2)
        , -1 ! difference_in_orientation_is(Agent, Patient, Diff_1)
        ,  0 ! difference_in_orientation_is(Agent, Patient, Diff)
        ,  1 ! difference_in_orientation_is(Agent, Patient, Diff1)
        ,  2 ! difference_in_orientation_is(Agent, Patient, Diff2)
        , derivative(Diff_2, Diff_1, Diff, Diff1, Diff2, Deriv)
        , associate_difference_in_orientation_change(Deriv, Value)
).

always( associate_difference_in_orientation_change(Deriv, Value) :-
        degreeOfValidity(Deriv, -15.0,  -5.0,   5.0,  15.0) , Value = constant ;
        degreeOfValidity(Deriv,   5.0,  15.0, 180.1, 180.1) , Value = changing_away ;
        degreeOfValidity(Deriv,-180.1,-180.1, -15.0,  -5.0) , Value = changing_towards
).

// ------------------------------------------------------------------------
// Relation between has_view_direction and move_direction_is of Agent

always( has_direction(Agent, Value) :-
        has_speed(Agent, moving)
        , has_view_direction(Agent, Dir1)
        , move_direction_is(Agent, Dir2)
        , A is Dir2 - Dir1
        , ang_norm(A, Norm)
        , associate_direction(Norm, Value)
).

always( associate_direction(A, Value) :-
        degreeOfValidity(A, -30.0, -20.0,  20.0,  30.0) , Value = straight ;
        degreeOfValidity(A,  15.0,  30.0, 110.0, 140.0) , Value = right ;
        degreeOfValidity(A,-140.0,-110.0, -30.0, -15.0) , Value = left ;
        degreeOfValidity(A,-180.1,-180.1,-120.0,-105.0) , Value = back ;
        degreeOfValidity(A, 105.0, 120.0, 180.1, 180.1) , Value = back
).
```

B.3 Test für Basisregeln

```
//========================================================================
// replace interface for testcase:

always( get_spatial_coefficient(Co) :-
        Co is 1.0
).


always( get_time_coefficient(Co) :-
        Co is 1.0
).

always( projectiveTransform(X, Y, X_neu, Y_neu) :-
        X_neu is X
        , Y_neu is Y
).

// data needed:
// has_image_geometry(Agent, X, Y, W, H)
// has_view_direction(Agent, Dir)
```

```
//============================================================================

//============================================================================
//    derivative(+A, +B, +C, +D, +E, ?Deriv)

always( test(derivative) :-
        pre_test(derivative, 'derivative(3, 3, 3, 3, 3, Val) => Val=0')
        , derivative(3, 3, 3, 3, 3, Val)
        , epsilon_neighborhood(0.001, Val, 0)
).

always( test(derivative) :-
        pre_test(derivative, 'derivative(1, 2, 3, 4, 5, Val) => Val>0')
        , derivative(1, 2, 3, 4, 5, Val)
        , Val > 0
).

always( test(derivative) :-
        pre_test(derivative, 'derivative(5, 4, 3, 2, 1, Val) => Val<0')
        , derivative(5, 4, 3, 2, 1, Val)
        , Val < 0
).

//============================================================================
//    degreeOfValidity(+X, +P1, +P2, +P3, +P4)

//    rate X according to 'degree of validity diagram' given by
//    floating point values P1, P2, P3, P4

always( test(degreeOfValidity) :-
        pre_test(degreeOfValidity,
                 'degreeOfValidity(3, 1.0, 2.0, 4.0, 5.0) => DoV 1')
        , (A1{degreeOfValidity(3, 1.0, 2.0, 4.0, 5.0)}B1)
        , epsilon_neighborhood(0.001, B1, 1.0)
).

always( test(degreeOfValidity) :-
        pre_test(degreeOfValidity,
                 'degreeOfValidity(2, 1.0, 3.0, 4.0, 5.0) => DoV 0.5')
        , (A1{degreeOfValidity(2, 1.0, 3.0, 4.0, 5.0)}B1)
        , epsilon_neighborhood(0.001, B1, 0.5)
).

always( test(degreeOfValidity) :-
        pre_test(degreeOfValidity,
                 'degreeOfValidity(4, 1.0, 2.0, 3.0, 5.0) => DoV 0.5')
        , (A1{degreeOfValidity(4, 1.0, 2.0, 3.0, 5.0)}B1)
        , epsilon_neighborhood(0.001, B1, 0.5)
).

always( test(degreeOfValidity) :-
        pre_test(degreeOfValidity,
                 'NOT degreeOfValidity(0, 1.0, 2.0, 4.0, 5.0)')
        , not(degreeOfValidity(0, 1.0, 2.0, 4.0, 5.0))
).

//============================================================================
//    length(+A, +B, ?L)

always( test(length) :-
        pre_test(length, 'length(0, 5, L) L=5')
        , length(0, 5, L)
        , L =:= 5
).
```

```
always( test(length) :-
        pre_test(length, 'length(4, 0, L) L=4')
        , length(4, 0, L)
        , L =:= 4
).

always( test(length) :-
        pre_test(length, 'length(3, 4, L) L=5')
        , length(3, 4, L)
        , epsilon_neighborhood(0.001, L, 5)
).

//============================================================================
//     max(+A, +B, ?M)

always( test(max) :-
        pre_test(max,'max(1,2,2)')
        , max(1,2,2)
).

always( test(max) :-
        pre_test(max,'max(1,1,1)')
        , max(1,1,1)
).

always( test(max) :-
        pre_test(max,'max(3,1,M) M=3')
        , max(3,1,M)
        , M =:= 3
).

//============================================================================
//     min(+A, +B, ?M)

always( test(min) :-
        pre_test(min,'min(1,2,1)')
        , min(1,2,1)
).

always( test(min) :-
        pre_test(min,'min(1,1,1)')
        , min(1,1,1)
).

always( test(min) :-
        pre_test(min,'min(3,1,M) M=1')
        , min(3,1,M)
        , M =:= 1
).

//============================================================================
//     abs(+Number, ?Abs)

always( test(abs) :-
        pre_test(abs,'abs(-3, Val) => Val = 3')
        , abs(-3, Val)
        , Val =:= 3
).

always( test(abs) :-
        pre_test(abs,'abs(3, Val) => Val = 3')
        , abs(3, Val)
        , Val =:= 3
).
```

```
always( test(abs) :-
        pre_test(abs,'abs(0, Val) => Val = 0')
        , abs(0, Val)
        , Val =:= 0
).

//=============================================================================
//     epsilon_neighborhood(+Epsilon, +Val1, +Val2)

always( test(epsilon_neighborhood) :-
        pre_test(epsilon_neighborhood, 'epsilon_neighborhood(0.1, 1.99, 2)')
        , epsilon_neighborhood(0.1, 1.99, 2)
).

always( test(epsilon_neighborhood) :-
        pre_test(epsilon_neighborhood, 'epsilon_neighborhood(0.1, 2.01, 2)')
        , epsilon_neighborhood(0.1, 2.01, 2)
).

always( test(epsilon_neighborhood) :-
        pre_test(epsilon_neighborhood,
                 'NOT epsilon_neighborhood(0.01, 2.1, 2)')
        , not(epsilon_neighborhood(0.01, 2.1, 2))
).

//=============================================================================
//     box_in_box(+Tiny, +Huge)

// -----------------------------------------------------------------------------
// some data needed for testing
always(has_image_geometry(obj_t1,    0,     0,   10,  10)).
always(has_image_geometry(obj_h1,    5,     5,   40,  40)).
always(has_image_geometry(obj_t2,    0,     0,  200, 100)).
always(has_image_geometry(obj_h2,    201,  50,   10,  20)).
// -----------------------------------------------------------------------------

always( test(box_in_box) :-
        pre_test(box_in_box, 'box_in_box(obj_t1, obj_h1) => DoV 0.25')
        , (A1{box_in_box(obj_t1, obj_h1)}B1)
        , epsilon_neighborhood(0.001, B1, 0.25)
).

always( test(box_in_box) :-
        pre_test(box_in_box, 'NOT box_in_box(obj_t2, obj_h2)')
        , not(box_in_box(obj_t2, obj_h2))
).

//=============================================================================
//     box_near_box

always( test(box_near_box) :-
        pre_test(box_near_box, 'box_near_box(obj_t2, obj_h2)')
        , (A1{box_near_box(obj_t2, obj_h2)}B1)
        , epsilon_neighborhood(0.1, B1, 1.0)
).


//=============================================================================
//    box_geom(+X1, +Y1, +W1, +H1, +X2, +Y2, +W2, +H2, ?X, ?Y, ?W, ?H)
//    combine two bounding boxes to one including both

always( test(box_geom) :-
        pre_test(box_geom,
                 'box_geom(0,1,10,20,0,1,10,20,X,Y,W,H) -> 0,1,10,20')
```

```
        , box_geom(0,1,10,20,0,1,10,20,X,Y,W,H)
        , X =:= 0
        , Y =:= 1
        , W =:= 10
        , H =:= 20
).

// +-------+          +-------+---+
// |       |          |           |
// |   +---+---+      |   +       +
// |   |   |   |  ->  |           |
// |   +---+---+      |   +       +
// |       |          |           |
// +-------+          +-------+---+

always( test(box_geom) :-
        pre_test(box_geom, 'box_geom(0,10,20,30, 10,20,20,10, 0,10,30,30)')
        , box_geom(0,10,20,30, 10,20,20,10, 0,10,30,30)
).

//     +-------+      +---+-------+
//     |       |      |           |
// +---+---+   |      +       +   |
// |   |   |   |  ->  |           |
// |   +---+---+      |   +       +
// |       |          |           |
// +-------+          +-------+---+

always( test(box_geom) :-
        pre_test(box_geom, 'box_geom(10,10,20,20, 0,0,20,20, 0,0,30,30)')
        , box_geom(10,10,20,20, 0,0,20,20, 0,0,30,30)
).

//============================================================================
//    box_obj(+Agent, +Patient, ?X, ?Y, ?W, ?H)
// combined bounding box of two objects Agent and Patient

// ----------------------------------------------------------------------------
// some data needed for testing
always(has_image_geometry(obj_box_1,    5,  10,  15, 25)).
always(has_image_geometry(obj_box_2,    0,  20,  40, 35)).
// ----------------------------------------------------------------------------

always( test(box_obj) :-
        pre_test(box_obj,
                 'box_obj(obj_box_1, obj_box_2, X, Y, W, H) -> 0,10,40,45')
        , box_obj(obj_box_1, obj_box_2, X, Y, W, H)
        , X =:= 0
        , Y =:= 10
        , W =:= 40
        , H =:= 45
).

//============================================================================
//      has_ground_geometry(+Agent, ?Xp, ?Yp)

// ----------------------------------------------------------------------------
// some data needed for testing
always(has_image_geometry(obj_g,   1,   2, 40, 170)).
// ----------------------------------------------------------------------------

always( test(has_ground_geometry) :-
        pre_test(has_ground_geometry,
                 'has_ground_geometry(obj_g, X, Y) => X=21, Y=172')
        , has_ground_geometry(obj_g, X, Y)
```

```
        , X =:= 21
        , Y =:= 172
).

//=============================================================================
//      has_geometry(+Agent, ?X, ?Y, ?W, ?H)

always( test(has_geometry) :-
        pre_test(has_geometry,
                 'has_geometry(obj_g, X, Y, W, H) => 1,2,40,170')
        , has_geometry(obj_g, X, Y, W, H)
        , X =:= 1
        , Y =:= 2
        , W =:= 40
        , H =:= 170
).

// ============================================================================
//      appear(+Agent)

// ----------------------------------------------------------------------------
// some data needed for testing
3 ! has_image_geometry(obj_app,   0,   0, 50, 170).
4 ! has_image_geometry(obj_app,   0,   0, 50, 170).
5 ! has_image_geometry(obj_app,   0,   0, 50, 170).
6 ! has_image_geometry(obj_app,   0,   0, 50, 170).
// ----------------------------------------------------------------------------

always( test(appear) :-
        pre_test(appear,'3 ? appear(obj_app)')
        , 3 ? appear(obj_app)
).

always( test(appear) :-
        pre_test(appear,'NOT 2 ? appear(obj_app)')
        , not(2 ? appear(obj_app))
).

always( test(appear) :-
        pre_test(appear,'NOT 4 ? appear(obj_app)')
        , not(4 ? appear(obj_app))
).

// ============================================================================
//      disappear(+Agent)

// ----------------------------------------------------------------------------
// some data needed for testing
0 ! has_image_geometry(obj_dapp,   0,   0, 50, 170).
1 ! has_image_geometry(obj_dapp,   0,   0, 50, 170).
2 ! has_image_geometry(obj_dapp,   0,   0, 50, 170).
3 ! has_image_geometry(obj_dapp,   0,   0, 50, 170).
// ----------------------------------------------------------------------------

always( test(disappear) :-
        pre_test(disappear,'3 ? disappear(obj_dapp)')
        , 3 ? disappear(obj_dapp)
).

always( test(disappear) :-
        pre_test(disappear,'NOT 2 ? disappear(obj_dapp)')
        , not (2 ? disappear(obj_dapp))
).

always( test(disappear) :-
```

```
        pre_test(disappear,'NOT 4 ? disappear(obj_dapp)')
        , not (4 ? disappear(obj_dapp))
).

//=============================================================================
//      distance_is(+Agent, +Patient, ?Distance)

// -----------------------------------------------------------------------------
// some data needed for testing
always(has_image_geometry(obj_dist_1,   0,   0, 50, 170)).
always(has_image_geometry(obj_dist_2,   0,   0, 50, 170)). //   0
always(has_image_geometry(obj_dist_3, 0.3, 0.4, 50, 170)). //   0.5
always(has_image_geometry(obj_dist_4,  24,  32, 50, 170)). //  40
always(has_image_geometry(obj_dist_5, 160, 120, 50, 170)). // 200
always(has_image_geometry(obj_dist_6, 500, 200, 50, 170)).
always(has_image_geometry(obj_dist_7, 100, 200, 50, 170)). // 400
// -----------------------------------------------------------------------------

always( test(distance_is) :-
        pre_test(distance_is,
                 'distance_is(obj_dist_1, obj_dist_2, Dist), Dist=0')
        , distance_is(obj_dist_1, obj_dist_2, Dist)
        , epsilon_neighborhood(0.001, Dist, 0)
).

always( test(distance_is) :-
        pre_test(distance_is,
                 'distance_is(obj_dist_1, obj_dist_3, Dist), Dist=0.5')
        , distance_is(obj_dist_1, obj_dist_3, Dist)
        , epsilon_neighborhood(0.001, Dist, 0.5)
).

always( test(distance_is) :-
        pre_test(distance_is,
                 'distance_is(obj_dist_1, obj_dist_4, Dist), Dist=40')
        , distance_is(obj_dist_1, obj_dist_4, Dist)
        , epsilon_neighborhood(0.001, Dist, 40)
).

always( test(distance_is) :-
        pre_test(distance_is,
                 'distance_is(obj_dist_1, obj_dist_5, Dist), Dist=200')
        , distance_is(obj_dist_1, obj_dist_5, Dist)
        , epsilon_neighborhood(0.001, Dist, 200)
).

always( test(distance_is) :-
        pre_test(distance_is,
                 'distance_is(obj_dist_6, obj_dist_7, Dist), Dist=400')
        , distance_is(obj_dist_6, obj_dist_7, Dist)
        , epsilon_neighborhood(0.001, Dist, 400)
).

//=============================================================================
//      have_distance(+Agent, +Patient, ?Value)

// -----------------------------------------------------------------------------
// data see distance_is
// -----------------------------------------------------------------------------

always( test(have_distance) :-
        pre_test(have_distance,
                 'have_distance(obj_dist_1, obj_dist_2, zero)')
        , have_distance(obj_dist_1, obj_dist_2, zero)
).
```

```
always( test(have_distance) :-
        pre_test(have_distance,
                 'have_distance(obj_dist_1, obj_dist_2, Val), Val=zero')
        , have_distance(obj_dist_1, obj_dist_2, Val)
        , Val == zero
).

always( test(have_distance) :-
        pre_test(have_distance,
                 'have_distance(obj_dist_1, obj_dist_3, Val), Val=zero')
        , have_distance(obj_dist_1, obj_dist_3, Val)
        , Val == zero
).

always( test(have_distance) :-
        pre_test(have_distance,
                 'have_distance(obj_dist_1, obj_dist_4, Val), Val=small')
        , have_distance(obj_dist_1, obj_dist_4, Val)
        , Val == small
).

always( test(have_distance) :-
        pre_test(have_distance,
                 'have_distance(obj_dist_1, obj_dist_5, Val), Val=normal')
        , have_distance(obj_dist_1, obj_dist_5, Val)
        , Val == normal
).

always( test(have_distance) :-
        pre_test(have_distance,
                 'have_distance(obj_dist_6, obj_dist_7, Val), Val=high')
        , have_distance(obj_dist_6, obj_dist_7, Val)
        , Val == high
).

//============================================================================
//    associate_distance(+Distance, ?Value)

always( test(associate_distance) :-
        pre_test(associate_distance, 'associate_distance(0, zero)')
        , associate_distance(0, zero)
).

always( test(associate_distance) :-
        pre_test(associate_distance, 'associate_distance(50, small)')
        , associate_distance(50, small)
).

always( test(associate_distance) :-
        pre_test(associate_distance, 'associate_distance(160, normal)')
        , associate_distance(160, normal)
).

always( test(associate_distance) :-
        pre_test(associate_distance, 'associate_distance(400, high)')
        , associate_distance(400, high)
).

always( test(associate_distance) :-
        pre_test(associate_distance, 'associate_distance(200, not_small)')
        , associate_distance(200, not_small)
).

always( test(associate_distance) :-
```

```
        pre_test(associate_distance, 'associate_distance(10, not_zero)')
        , associate_distance(10, not_zero)
).

//=============================================================================
//      have_distance_change(+Agent, +Patient, ?Value)

// -----------------------------------------------------------------------------
// some data needed for testing
  0 ! has_image_geometry(obj_dc_1,   0,   0, 50, 170).
  0 ! has_image_geometry(obj_dc_2,   4,   3, 50, 170). //  5
  1 ! has_image_geometry(obj_dc_1,   1,   2, 50, 170).
  1 ! has_image_geometry(obj_dc_2,   1,  11, 50, 170). //  9
  2 ! has_image_geometry(obj_dc_1,   2,   3, 50, 170).
  2 ! has_image_geometry(obj_dc_2,  14,  12, 50, 170). // 15
  3 ! has_image_geometry(obj_dc_1,   2,   3, 50, 170).
  3 ! has_image_geometry(obj_dc_2,  19,   3, 50, 170). // 17
  4 ! has_image_geometry(obj_dc_1,   5,   4, 50, 170).
  4 ! has_image_geometry(obj_dc_2,  17,  20, 50, 170). // 20

  0 ! has_image_geometry(obj_dc_3,   0,   0, 50, 170).
  0 ! has_image_geometry(obj_dc_4,   0,   5, 50, 170). //  5
  1 ! has_image_geometry(obj_dc_3,  -1,   0, 50, 170).
  1 ! has_image_geometry(obj_dc_4,   1,   2, 50, 170). //  2*sqrt(2)
  2 ! has_image_geometry(obj_dc_3,   0,  -1, 50, 170).
  2 ! has_image_geometry(obj_dc_4,  -1,   0, 50, 170). //  sqrt(2)
  3 ! has_image_geometry(obj_dc_3,   0,   0, 50, 170).
  3 ! has_image_geometry(obj_dc_4,   0,   2, 50, 170). //  2
  4 ! has_image_geometry(obj_dc_3,   5,   0, 50, 170).
  4 ! has_image_geometry(obj_dc_4,   0,   0, 50, 170). //  5
// -----------------------------------------------------------------------------

always( test(have_distance_change) :-
        pre_test(have_distance_change,
                 '2 ? have_distance_change(obj_dc_1, obj_dc_2, Val)'+
                 ' => Val=slightly_increasing')
        , 2 ? have_distance_change(obj_dc_1, obj_dc_2, Val)
        , Val == slightly_increasing
).

always( test(have_distance_change) :-
        pre_test(have_distance_change,
           '2 ? have_distance_change(obj_dc_3, obj_dc_4, Val) => Val=constant')
        , 2 ? have_distance_change(obj_dc_3, obj_dc_4, Val)
        , Val == constant
).

//=============================================================================
//      associate_distance_change(+Deriv, ?Value)

always( test(associate_distance_change) :-
        pre_test(associate_distance_change,
                 'associate_distance_change(0, constant)')
        , associate_distance_change(0, constant)
).

always( test(associate_distance_change) :-
        pre_test(associate_distance_change,
                 'associate_distance_change(-10, slightly_decreasing)')
        , associate_distance_change(-10, slightly_decreasing)
).

always( test(associate_distance_change) :-
        pre_test(associate_distance_change,
                 'associate_distance_change(10, slightly_increasing)')
```

```
        , associate_distance_change(10, slightly_increasing)
).

always( test(associate_distance_change) :-
        pre_test(associate_distance_change,
                 'associate_distance_change(-30, nearly_constant)')
        , associate_distance_change(-30, nearly_constant)
).

always( test(associate_distance_change) :-
        pre_test(associate_distance_change,
                 'associate_distance_change(30, nearly_constant)')
        , associate_distance_change(30, nearly_constant)
).

// ============================================================================
//      has_speed(+Agent, ?Value)

// ----------------------------------------------------------------------------
// some data needed for testing
  0 ! has_image_geometry(obj_speed,   0,   0,  50, 170).
  1 ! has_image_geometry(obj_speed, 103, 103,  50, 170).
  2 ! has_image_geometry(obj_speed, 206, 206,  50, 170).
  3 ! has_image_geometry(obj_speed, 309, 309,  50, 170).
  4 ! has_image_geometry(obj_speed, 412, 412,  50, 170).
  5 ! has_image_geometry(obj_speed, 515, 515,  50, 170).
  6 ! has_image_geometry(obj_speed, 618, 618,  50, 170).
// ----------------------------------------------------------------------------

always( test(has_speed) :-
        pre_test(has_speed, '3 ? has_speed(obj_speed, Val) => Val=normal')
        , 3 ? has_speed(obj_speed, Val)
        , Val == normal
).

// ============================================================================
//      associate_speed(+Speed, ?Value)

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(0, zero)')
        , associate_speed(0, zero)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(40, very_small)')
        , associate_speed(40, very_small)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(90, small)')
        , associate_speed(90, small)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(150, normal)')
        , associate_speed(150, normal)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(180, high)')
        , associate_speed(180, high)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(300, very_high)')
```

```
        , associate_speed(300, very_high)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(30, moving)')
        , associate_speed(30, moving)
).

always( test(associate_speed) :-
        pre_test(associate_speed, 'associate_speed(-20, backwards)')
        , associate_speed(-20, backwards)
).

// ============================================================================
//      atan2(+Y, +X, ?R)

always( test(atan2) :-
        pre_test(atan2, 'atan2(0, 1, R) => R=0')
        , atan2(0, 1, R)
        , epsilon_neighborhood(0.001, R, 0.0)
).

always( test(atan2) :-
        pre_test(atan2, 'atan2(-1, 1, R) => R=-Pi/4')
        , atan2(-1, 1, R)
        , epsilon_neighborhood(0.001, R, -3.1415926/4)
).

always( test(atan2) :-
        pre_test(atan2, 'atan2(1, 1, R) => R=Pi/4')
        , atan2(1, 1, R)
        , epsilon_neighborhood(0.001, R, 3.1415926/4)
).

always( test(atan2) :-
        pre_test(atan2, 'atan2(-1, 0, R) => R=-Pi/2')
        , atan2(-1, 0, R)
        , epsilon_neighborhood(0.001, R, -3.1415926/2)
).

always( test(atan2) :-
        pre_test(atan2, 'atan2(1, 0, R) => R=Pi/2')
        , atan2(1, 0, R)
        , epsilon_neighborhood(0.001, R, 3.1415926/2)
).

always( test(atan2) :-
        pre_test(atan2, 'atan2(1, -1, R) => R=3*Pi/4')
        , atan2(1, -1, R)
        , epsilon_neighborhood(0.001, R, 3.1415926*3/4)
).

always( test(atan2) :-
        pre_test(atan2, 'atan2(-1, -1, R) => R=-3*Pi/4')
        , atan2(-1, -1, R)
        , epsilon_neighborhood(0.001, R, -3.1415926*3/4)
).

//============================================================================
//      convert_angle(+Radians, -Degrees)
//      convert_angle(-Radians, +Degrees)

always( test(convert_angle) :-
        pre_test(convert_angle, 'convert_angle(Pi/2, Val) => Val=90')
        , convert_angle(3.1415926/2, Val)
```

```
        , epsilon_neighborhood(0.001, Val, 90)
).

always( test(convert_angle) :-
        pre_test(convert_angle, 'convert_angle(Val, 90) => Val=Pi/2')
        , convert_angle(Val, 90)
        , epsilon_neighborhood(0.001, Val, 3.1415926/2)
).

always( test(convert_angle) :-
        pre_test(convert_angle, 'convert_angle(-Pi*3/4, Val) => Val=-135')
        , convert_angle(-3.1415926*3/4, Val)
        , epsilon_neighborhood(0.001, Val, -135)
).

always( test(convert_angle) :-
        pre_test(convert_angle, 'convert_angle(Val, -135) => Val=-Pi*3/4')
        , convert_angle(Val, -135)
        , epsilon_neighborhood(0.001, Val, -3.1415926*3/4)
).

always( test(convert_angle) :-
        pre_test(convert_angle, 'convert_angle(Pi*3/4, Val) => Val=135')
        , convert_angle(3.1415926*3/4, Val)
        , epsilon_neighborhood(0.001, Val, 135)
).

always( test(convert_angle) :-
        pre_test(convert_angle, 'convert_angle(Val, 135) => Val=Pi*3/4')
        , convert_angle(Val, 135)
        , epsilon_neighborhood(0.001, Val, 3.1415926*3/4)
).

//=============================================================================
//      ang_direction(+X, +Y, ?Ang)

always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(1, 0, R) => R=0')
        , ang_direction(1, 0, R)
        , epsilon_neighborhood(0.001, R, 0.0)
).

always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(1, -1, R) => R=-45')
        , ang_direction(1, -1, R)
        , epsilon_neighborhood(0.001, R, -45)
).

always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(1, 1, R) => R=45')
        , ang_direction(1, 1, R)
        , epsilon_neighborhood(0.001, R, 45)
).

always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(0, -1, R) => R=-90')
        , ang_direction(0, -1, R)
        , epsilon_neighborhood(0.001, R, -90)
).

always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(0, 1, R) => R=90')
        , ang_direction(0, 1, R)
        , epsilon_neighborhood(0.001, R, 90)
).
```

```
always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(-1, 1, R) => R=135')
        , ang_direction(-1, 1, R)
        , epsilon_neighborhood(0.001, R, 135)
).

always( test(ang_direction) :-
        pre_test(ang_direction, 'ang_direction(-1, -1, R) => R=-135')
        , ang_direction(-1, -1, R)
        , epsilon_neighborhood(0.001, R, -135)
).

//============================================================================
//      ang_norm(+Ang, ?Norm)

always( test(ang_norm) :-
        pre_test(ang_norm, 'ang_norm(270, Norm) => Norm=-90')
        , ang_norm(270, Norm)
        , epsilon_neighborhood(0.001, Norm, -90)
).

always( test(ang_norm) :-
        pre_test(ang_norm, 'ang_norm(-225, Norm) => Norm=135')
        , ang_norm(-225, Norm)
        , epsilon_neighborhood(0.001, Norm, 135)
).

//============================================================================
//      move_direction_is(+Agent, ?Dir)

// ----------------------------------------------------------------------------
// some data needed for testing
  0 ! has_image_geometry(obj_dir_7,   0,   0, 50, 170).
  1 ! has_image_geometry(obj_dir_7,   1,   1, 50, 170).
  2 ! has_image_geometry(obj_dir_7,   2,   2, 50, 170).
  3 ! has_image_geometry(obj_dir_7,  10,  10, 50, 170).
  4 ! has_image_geometry(obj_dir_7,  11,  11, 50, 170).
  5 ! has_image_geometry(obj_dir_7, 100, 100, 50, 170).
  6 ! has_image_geometry(obj_dir_7, 101, 101, 50, 170).
// ----------------------------------------------------------------------------

always( test(move_direction_is) :-
        pre_test(move_direction_is,
                 '3 ? move_direction_is(obj_dir_1, Val), Val=-90')
        , 3 ? move_direction_is(obj_dir_1, Val)
        , epsilon_neighborhood(0.001, Val, -90)
).

always( test(move_direction_is) :-
        pre_test(move_direction_is,
                 '3 ? move_direction_is(obj_dir_7, Val), Val=45')
        , 3 ? move_direction_is(obj_dir_7, Val)
        , epsilon_neighborhood(0.001, Val, 45)
).

//============================================================================
//      distance_direction_is(+Agent, +Patient, ?Dir)

// ----------------------------------------------------------------------------
// some data needed for testing:                   5     6
//
//                                                     1 2 --->
//                                                 7   3 4
//                                                     |
```

```
//                                                  V
always(has_ground_geometry(obj_dd_1,   0,   0)).
always(has_ground_geometry(obj_dd_2,   1,   0)).
always(has_ground_geometry(obj_dd_3,   0,   1)).
always(has_ground_geometry(obj_dd_4,   1,   1)).
always(has_ground_geometry(obj_dd_5,  -2,  -2)).
always(has_ground_geometry(obj_dd_6,   1,  -2)).
always(has_ground_geometry(obj_dd_7,  -2,   1)).
// ------------------------------------------------------------------------

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_1, obj_dd_2, Val), Val=0')
        , distance_direction_is(obj_dd_1, obj_dd_2, Val)
        , epsilon_neighborhood(0.001, Val, 0)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_1, obj_dd_3, Val), Val=90')
        , distance_direction_is(obj_dd_1, obj_dd_3, Val)
        , epsilon_neighborhood(0.001, Val, 90)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_1, obj_dd_4, Val), Val=45')
        , distance_direction_is(obj_dd_1, obj_dd_4, Val)
        , epsilon_neighborhood(0.001, Val, 45)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_2, obj_dd_1, Val), Val=+-180')
        , distance_direction_is(obj_dd_2, obj_dd_1, Val)
        , abs(Val, Abs)
        , epsilon_neighborhood(0.001, Abs, 180)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_2, obj_dd_3, Val), Val=135')
        , distance_direction_is(obj_dd_2, obj_dd_3, Val)
        , epsilon_neighborhood(0.001, Val, 135)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_3, obj_dd_1, Val), Val=-90')
        , distance_direction_is(obj_dd_3, obj_dd_1, Val)
        , epsilon_neighborhood(0.001, Val, -90)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_3, obj_dd_2, Val), Val=-45')
        , distance_direction_is(obj_dd_3, obj_dd_2, Val)
        , epsilon_neighborhood(0.001, Val, -45)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_4, obj_dd_1, Val), Val=-135')
        , distance_direction_is(obj_dd_4, obj_dd_1, Val)
        , epsilon_neighborhood(0.001, Val, -135)
```

```
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_5, obj_dd_6, Val), Val=0')
        , distance_direction_is(obj_dd_5, obj_dd_6, Val)
        , epsilon_neighborhood(0.001, Val, 0)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_5, obj_dd_7, Val), Val=90')
        , distance_direction_is(obj_dd_5, obj_dd_7, Val)
        , epsilon_neighborhood(0.001, Val, 90)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_5, obj_dd_4, Val), Val=45')
        , distance_direction_is(obj_dd_5, obj_dd_4, Val)
        , epsilon_neighborhood(0.001, Val, 45)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_6, obj_dd_5, Val), Val=+-180')
        , distance_direction_is(obj_dd_6, obj_dd_5, Val)
        , abs(Val, Abs)
        ,epsilon_neighborhood(0.001,  Abs, 180)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_6, obj_dd_7, Val), Val=135')
        , distance_direction_is(obj_dd_6, obj_dd_7, Val)
        , epsilon_neighborhood(0.001, Val, 135)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_7, obj_dd_5, Val), Val=-90')
        , distance_direction_is(obj_dd_7, obj_dd_5, Val)
        , epsilon_neighborhood(0.001, Val, -90)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_7, obj_dd_6, Val), Val=-45')
        , distance_direction_is(obj_dd_7, obj_dd_6, Val)
        , epsilon_neighborhood(0.001, Val, -45)
).

always( test(distance_direction_is) :-
        pre_test(distance_direction_is,
                 'distance_direction_is(obj_dd_4, obj_dd_5, Val), Val=-135')
        , distance_direction_is(obj_dd_4, obj_dd_5, Val)
        , epsilon_neighborhood(0.001, Val, -135)
).

//============================================================================
//      associate_angle(+Dir1, +Dir2, ?Value)

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(0, 90, right)')
        , associate_angle(0, 90, right)
```

```
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(90, 180, right)')
        , associate_angle(90, 180, right)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(180, -90, right)')
        , associate_angle(180, -90, right)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(-90, 0, right)')
        , associate_angle(-90, 0, right)
).

//=============================================================================
//      associate_angle(+Ang, ?Value)

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(0, ahead)')
        , associate_angle(0, ahead)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(45, right_ahead)')
        , associate_angle(45, right_ahead)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(-45, left_ahead)')
        , associate_angle(-45, left_ahead)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(-90, left)')
        , associate_angle(-90, left)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(90, right)')
        , associate_angle(90, right)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(135, right_behind)')
        , associate_angle(135, right_behind)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(-135, left_behind)')
        , associate_angle(-135, left_behind)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(-180, behind)')
        , associate_angle(-180, behind)
).

always( test(associate_angle) :-
        pre_test(associate_angle, 'associate_angle(180, behind)')
        , associate_angle(180, behind)
).
```

```
//=============================================================================
//      relative_position_is(+Agent, +Patient, ?Angle)

// -----------------------------------------------------------------------------
// some data needed for testing:                          |
//                                                  3     2
//                                                 /
//                                                      +----->
//                                                 -4   |-1
//                                                      |
//                                                      V
always(has_ground_geometry(obj_rp_1,   1,   1)).
always(has_ground_geometry(obj_rp_2,   1,  -2)).
always(has_ground_geometry(obj_rp_3,  -2,  -2)).
always(has_ground_geometry(obj_rp_4,  -2,   1)).
always(has_view_direction(obj_rp_1, 180)).
always(has_view_direction(obj_rp_2, -90)).
always(has_view_direction(obj_rp_3, 135)).
always(has_view_direction(obj_rp_4, 180)).
// -----------------------------------------------------------------------------

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_1, obj_rp_2, Val) => Val=90')
        , relative_position_is(obj_rp_1, obj_rp_2, Val)
        , epsilon_neighborhood(0.001, Val, 90)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_1, obj_rp_3, Val) => Val=45')
        , relative_position_is(obj_rp_1, obj_rp_3, Val)
        , epsilon_neighborhood(0.001, Val, 45)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_1, obj_rp_4, Val) => Val=0')
        , relative_position_is(obj_rp_1, obj_rp_4, Val)
        , epsilon_neighborhood(0.001, Val, 0)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_2, obj_rp_1, Val) => Val=+-180')
        , relative_position_is(obj_rp_2, obj_rp_1, Val)
        , abs(Val, Abs)
        , epsilon_neighborhood(0.001, Abs, 180)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_2, obj_rp_3, Val) => Val=-90')
        , relative_position_is(obj_rp_2, obj_rp_3, Val)
        , epsilon_neighborhood(0.001, Val, -90)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_2, obj_rp_4, Val) => Val=-135')
        , relative_position_is(obj_rp_2, obj_rp_4, Val)
        , epsilon_neighborhood(0.001, Val, -135)
).

always( test(relative_position_is) :-
```

```
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_3, obj_rp_1, Val) => Val=-90')
        , relative_position_is(obj_rp_3, obj_rp_1, Val)
        , epsilon_neighborhood(0.001, Val, -90)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_3, obj_rp_2, Val) => Val=-135')
        , relative_position_is(obj_rp_3, obj_rp_2, Val)
        , epsilon_neighborhood(0.001, Val, -135)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_3, obj_rp_4, Val) => Val=-45')
        , relative_position_is(obj_rp_3, obj_rp_4, Val)
        , epsilon_neighborhood(0.001, Val, -45)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_4, obj_rp_1, Val) => Val=+-180')
        , relative_position_is(obj_rp_4, obj_rp_1, Val)
        , abs(Val, Abs)
        , epsilon_neighborhood(0.001, Abs, 180)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_4, obj_rp_2, Val) => Val=135')
        , relative_position_is(obj_rp_4, obj_rp_2, Val)
        , epsilon_neighborhood(0.001, Val, 135)
).

always( test(relative_position_is) :-
        pre_test(relative_position_is,
                 'relative_position_is(obj_rp_4, obj_rp_3, Val) => Val=90')
        , relative_position_is(obj_rp_4, obj_rp_3, Val)
        , epsilon_neighborhood(0.001, Val, 90)
).

//============================================================================
//      have_relative_position(+Agent, +Patient, ?Value)

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_1, obj_rp_2, Val) => Val=right')
        , have_relative_position(obj_rp_1, obj_rp_2, Val)
        , Val == right
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_1, obj_rp_3, Val) => Val=half_right')
        , have_relative_position(obj_rp_1, obj_rp_3, Val)
        , Val == half_right
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_1, obj_rp_4, Val) => Val=straight')
        , have_relative_position(obj_rp_1, obj_rp_4, Val)
        , Val == straight
).
```

```
always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_2, obj_rp_1, Val) => Val=straight')
        , have_relative_position(obj_rp_2, obj_rp_1, Val)
        , Val == straight
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_2, obj_rp_3, Val) => Val=left')
        , have_relative_position(obj_rp_2, obj_rp_3, Val)
        , Val == left
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_2, obj_rp_4, Val) => Val=half_left')
        , have_relative_position(obj_rp_2, obj_rp_4, Val)
        , Val == half_left
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_3, obj_rp_1, Val) => Val=left')
        , have_relative_position(obj_rp_3, obj_rp_1, Val)
        , Val == left
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_3, obj_rp_2, Val) => Val=half_left')
        , have_relative_position(obj_rp_3, obj_rp_2, Val)
        , Val == half_left
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_3, obj_rp_4, Val) => Val=half_left')
        , have_relative_position(obj_rp_3, obj_rp_4, Val)
        , Val == half_left
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_4, obj_rp_1, Val) => Val=straight')
        , have_relative_position(obj_rp_4, obj_rp_1, Val)
        , Val == straight
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_4, obj_rp_2, Val) => Val=half_right')
        , have_relative_position(obj_rp_4, obj_rp_2, Val)
        , Val == half_right
).

always( test(have_relative_position) :-
        pre_test(have_relative_position,
           'have_relative_position(obj_rp_4, obj_rp_3, Val) => Val=right')
        , have_relative_position(obj_rp_4, obj_rp_3, Val)
        , Val == right
).

//============================================================================
```

```
//      have_view_angle(+Agent, +Patient, ?Value)

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_1, obj_rp_2, Val) => Val=right')
        , have_view_angle(obj_rp_1, obj_rp_2, Val)
        , Val == right
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_1, obj_rp_3, Val) => Val=right_ahead')
        , have_view_angle(obj_rp_1, obj_rp_3, Val)
        , Val == right_ahead
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_1, obj_rp_4, Val) => Val=ahead')
        , have_view_angle(obj_rp_1, obj_rp_4, Val)
        , Val == ahead
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_2, obj_rp_1, Val) => Val=behind')
        , have_view_angle(obj_rp_2, obj_rp_1, Val)
        , Val == behind
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_2, obj_rp_3, Val) => Val=left')
        , have_view_angle(obj_rp_2, obj_rp_3, Val)
        , Val == left
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_2, obj_rp_4, Val) => Val=left_behind')
        , have_view_angle(obj_rp_2, obj_rp_4, Val)
        , Val == left_behind
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_3, obj_rp_1, Val) => Val=left')
        , have_view_angle(obj_rp_3, obj_rp_1, Val)
        , Val == left
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_3, obj_rp_2, Val) => Val=left_behind')
        , have_view_angle(obj_rp_3, obj_rp_2, Val)
        , Val == left_behind
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_3, obj_rp_4, Val) => Val=left_ahead')
        , have_view_angle(obj_rp_3, obj_rp_4, Val)
        , Val == left_ahead
).
```

```
always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_4, obj_rp_1, Val) => Val=behind')
        , have_view_angle(obj_rp_4, obj_rp_1, Val)
        , Val == behind
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_4, obj_rp_2, Val) => Val=right_behind')
        , have_view_angle(obj_rp_4, obj_rp_2, Val)
        , Val == right_behind
).

always( test(have_view_angle) :-
        pre_test(have_view_angle,
                 'have_view_angle(obj_rp_4, obj_rp_3, Val) => Val=right')
        , have_view_angle(obj_rp_4, obj_rp_3, Val)
        , Val == right
).

//============================================================================
//      associate_relative_position(+Ang, ?Value)

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(0, straight)')
        , associate_relative_position(0, straight)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(179, straight)')
        , associate_relative_position(179, straight)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(-179, straight)')
        , associate_relative_position(-179, straight)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(45, half_right)')
        , associate_relative_position(45, half_right)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(135, half_right)')
        , associate_relative_position(135, half_right)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(-45, half_left)')
        , associate_relative_position(-45, half_left)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(-135, half_left)')
        , associate_relative_position(-135, half_left)
).
```

```
always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(90, right)')
        , associate_relative_position(90, right)
).

always( test(associate_relative_position) :-
        pre_test(associate_relative_position,
                 'associate_relative_position(-90, left)')
        , associate_relative_position(-90, left)
).

//=============================================================================
//      have_configuration(+Agent, +Patient, ?Value)

always( test(have_configuration) :-
        pre_test(have_configuration,
                 'have_configuration(obj_rp_1, obj_rp_4, Val) => Val=front')
        , have_configuration(obj_rp_1, obj_rp_4, Val)
        , Val == front
).

always( test(have_configuration) :-
        pre_test(have_configuration,
                 'have_configuration(obj_rp_3, obj_rp_2, Val) => Val=behind')
        , have_configuration(obj_rp_3, obj_rp_2, Val)
        , Val == behind
).

always( test(have_configuration) :-
        pre_test(have_configuration,
                 'have_configuration(obj_rp_4, obj_rp_1, Val) => Val=behind')
        , have_configuration(obj_rp_4, obj_rp_1, Val)
        , Val == behind
).

always( test(have_configuration) :-
        pre_test(have_configuration,
                 'have_configuration(obj_rp_1, obj_rp_2, Val) => Val=beside')
        , have_configuration(obj_rp_1, obj_rp_2, Val)
        , Val == beside
).

always( test(have_configuration) :-
        pre_test(have_configuration,
                 'have_configuration(obj_rp_3, obj_rp_1, Val) => Val=beside')
        , have_configuration(obj_rp_3, obj_rp_1, Val)
        , Val == beside
).

//=============================================================================
//      associate_configuration(+Ang, ?Value)

always( test(associate_configuration) :-
        pre_test(associate_configuration,
                 'associate_configuration(0, front)')
        , associate_configuration(0, front)
).

always( test(associate_configuration) :-
        pre_test(associate_configuration,
                 'associate_configuration(135, behind)')
        , associate_configuration(135, behind)
).
```

```
always( test(associate_configuration) :-
        pre_test(associate_configuration,
                 'associate_configuration(-135, behind)')
        , associate_configuration(-135, behind)
).

always( test(associate_configuration) :-
        pre_test(associate_configuration,
                 'associate_configuration(90, beside)')
        , associate_configuration(90, beside)
).

always( test(associate_configuration) :-
        pre_test(associate_configuration,
                 'associate_configuration(-90, beside)')
        , associate_configuration(-90, beside)
).

//============================================================================
//      have_move_angle(+Agent, +Patient, ?Value)

// ----------------------------------------------------------------------------
// some data needed for testing
// 1
// . . 2
// 3 4      /\
// 5 6      ||
//
  0 ! has_ground_geometry(obj_dir_1,   0,  80).
  0 ! has_ground_geometry(obj_dir_2,   2,  81).
  0 ! has_ground_geometry(obj_dir_3,   0,  82).
  0 ! has_ground_geometry(obj_dir_4,   1,  82).
  0 ! has_ground_geometry(obj_dir_5,   0,  83).
  0 ! has_ground_geometry(obj_dir_6,   1,  83).
  1 ! has_ground_geometry(obj_dir_1,   0,  70).
  1 ! has_ground_geometry(obj_dir_2,   2,  71).
  1 ! has_ground_geometry(obj_dir_3,   0,  72).
  1 ! has_ground_geometry(obj_dir_4,   1,  72).
  1 ! has_ground_geometry(obj_dir_5,   0,  73).
  1 ! has_ground_geometry(obj_dir_6,   1,  73).
  2 ! has_ground_geometry(obj_dir_1,   0,  60).
  2 ! has_ground_geometry(obj_dir_2,   2,  61).
  2 ! has_ground_geometry(obj_dir_3,   0,  62).
  2 ! has_ground_geometry(obj_dir_4,   1,  62).
  2 ! has_ground_geometry(obj_dir_5,   0,  63).
  2 ! has_ground_geometry(obj_dir_6,   1,  63).
  3 ! has_ground_geometry(obj_dir_1,   0,  50).
  3 ! has_ground_geometry(obj_dir_2,   2,  51).
  3 ! has_ground_geometry(obj_dir_3,   0,  52).
  3 ! has_ground_geometry(obj_dir_4,   1,  52).
  3 ! has_ground_geometry(obj_dir_5,   0,  53).
  3 ! has_ground_geometry(obj_dir_6,   1,  53).
  4 ! has_ground_geometry(obj_dir_1,   0,  40).
  4 ! has_ground_geometry(obj_dir_2,   2,  41).
  4 ! has_ground_geometry(obj_dir_3,   0,  42).
  4 ! has_ground_geometry(obj_dir_4,   1,  42).
  4 ! has_ground_geometry(obj_dir_5,   0,  43).
  4 ! has_ground_geometry(obj_dir_6,   1,  43).
  5 ! has_ground_geometry(obj_dir_1,   0,  30).
  5 ! has_ground_geometry(obj_dir_2,   2,  31).
  5 ! has_ground_geometry(obj_dir_3,   0,  32).
  5 ! has_ground_geometry(obj_dir_4,   1,  32).
  5 ! has_ground_geometry(obj_dir_5,   0,  33).
  5 ! has_ground_geometry(obj_dir_6,   1,  33).
```

```
  6 ! has_ground_geometry(obj_dir_1,   0,  20).
  6 ! has_ground_geometry(obj_dir_2,   2,  21).
  6 ! has_ground_geometry(obj_dir_3,   0,  22).
  6 ! has_ground_geometry(obj_dir_4,   1,  22).
  6 ! has_ground_geometry(obj_dir_5,   0,  23).
  6 ! has_ground_geometry(obj_dir_6,   1,  23).
// ------------------------------------------------------------------------

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_1, obj_dir_2, Val), Val=right')
        , 3 ? have_move_angle(obj_dir_1, obj_dir_2, Val)
        , Val == right
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_4, obj_dir_1, Val), Val=left_ahead')
        , 3 ? have_move_angle(obj_dir_4, obj_dir_1, Val)
        , Val == left_ahead
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_3, obj_dir_1, Val), Val=ahead')
        , 3 ? have_move_angle(obj_dir_3, obj_dir_1, Val)
        , Val == ahead
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_6, obj_dir_2, Val), Val=right_ahead')
        , 3 ? have_move_angle(obj_dir_6, obj_dir_2, Val)
        , Val == right_ahead
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_3, obj_dir_4, Val), Val=right')
        , 3 ? have_move_angle(obj_dir_3, obj_dir_4, Val)
        , Val == right
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_1, obj_dir_4, Val), Val=right_behind')
        , 3 ? have_move_angle(obj_dir_1, obj_dir_4, Val)
        , Val == right_behind
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_1, obj_dir_5, Val), Val=behind')
        , 3 ? have_move_angle(obj_dir_1, obj_dir_5, Val)
        , Val == behind
).

always( test(have_move_angle) :-
        pre_test(have_move_angle,
                 'have_move_angle(obj_dir_2, obj_dir_6, Val), Val=left_behind')
        , 3 ? have_move_angle(obj_dir_2, obj_dir_6, Val)
        , Val == left_behind
).

always( test(have_move_angle) :-
```

```
        pre_test(have_move_angle,
                ’have_move_angle(obj_dir_4, obj_dir_3, Val), Val=left’)
        , 3 ? have_move_angle(obj_dir_4, obj_dir_3, Val)
        , Val == left
).

//=============================================================================
//      have_course(+Agent, +Patient, ?Value)

// -----------------------------------------------------------------------------
// some data needed for testing ^
//                              ’
//                              ’
//                              ’
//                           ^  ’      ^
//                           +  ’     .
//                           +  4    .
//                           +  1  <.,,,,,3
//                           +      .
//                           +     .
//                           +    .
//                           5  2

  0 ! has_ground_geometry(obj_course_1,    0,    0).
  0 ! has_ground_geometry(obj_course_2,    0,   40).
  0 ! has_ground_geometry(obj_course_3,   90,    0).
  0 ! has_ground_geometry(obj_course_4,    0,  -10).
  0 ! has_ground_geometry(obj_course_5,  -40,   40).

  1 ! has_ground_geometry(obj_course_1,    0,    0).
  1 ! has_ground_geometry(obj_course_2,   10,   30).
  1 ! has_ground_geometry(obj_course_3,   80,    0).
  1 ! has_ground_geometry(obj_course_4,    0,  -20).
  1 ! has_ground_geometry(obj_course_5,  -40,   30).

  2 ! has_ground_geometry(obj_course_1,    0,    0).
  2 ! has_ground_geometry(obj_course_2,   20,   20).
  2 ! has_ground_geometry(obj_course_3,   70,    0).
  2 ! has_ground_geometry(obj_course_4,    0,  -30).
  2 ! has_ground_geometry(obj_course_5,  -40,   20).

  3 ! has_ground_geometry(obj_course_1,    0,    0).
  3 ! has_ground_geometry(obj_course_2,   30,   10).
  3 ! has_ground_geometry(obj_course_3,   60,    0).
  3 ! has_ground_geometry(obj_course_4,    0,  -40).
  3 ! has_ground_geometry(obj_course_5,  -40,   10).

  4 ! has_ground_geometry(obj_course_1,    0,    0).
  4 ! has_ground_geometry(obj_course_2,   40,    0).
  4 ! has_ground_geometry(obj_course_3,   50,    0).
  4 ! has_ground_geometry(obj_course_4,    0,  -50).
  4 ! has_ground_geometry(obj_course_5,  -40,    0).

  5 ! has_ground_geometry(obj_course_1,    0,    0).
  5 ! has_ground_geometry(obj_course_2,   50,  -10).
  5 ! has_ground_geometry(obj_course_3,   40,    0).
  5 ! has_ground_geometry(obj_course_4,    0,  -60).
  5 ! has_ground_geometry(obj_course_5,  -40,  -10).

  6 ! has_ground_geometry(obj_course_1,    0,    0).
  6 ! has_ground_geometry(obj_course_2,   60,  -20).
  6 ! has_ground_geometry(obj_course_3,   30,    0).
  6 ! has_ground_geometry(obj_course_4,    0,  -70).
  6 ! has_ground_geometry(obj_course_5,  -40,  -20).
// -----------------------------------------------------------------------------
```

```
always( test(have_course) :-
        pre_test(have_course,
                 '3 ? have_course(obj_course_2, obj_course_1, passing_right)')
        , 3 ? have_course(obj_course_2, obj_course_1, passing_right)
).

always( test(have_course) :-
        pre_test(have_course,
                 '3 ? have_course(obj_course_3, obj_course_1, approaching)')
        , 3 ? have_course(obj_course_3, obj_course_1, approaching)
).

always( test(have_course) :-
        pre_test(have_course,
                 '3 ? have_course(obj_course_4, obj_course_1, leaving)')
        , 3 ? have_course(obj_course_4, obj_course_1, leaving)
).

always( test(have_course) :-
        pre_test(have_course,
                 '3 ? have_course(obj_course_5, obj_course_1, passing_left)')
        , 3 ? have_course(obj_course_5, obj_course_1, passing_left)
).


//=============================================================================
//      course_is(+Agent, +Patient, ?Course)

always( test(course_is) :-
        pre_test(course_is,
                 '3 ? course_is(obj_course_3, obj_course_1, Val) => Val=0')
        , 3 ? course_is(obj_course_3, obj_course_1, Val)
        , epsilon_neighborhood(0.001, Val, 0)
).

//=============================================================================
//      associate_course(+Course, ?Value)

always( test(associate_course) :-
        pre_test(associate_course, 'associate_course(0, approaching)')
        , associate_course(0, approaching)
).

always( test(associate_course) :-
        pre_test(associate_course, 'associate_course(120, passing_left)')
        , associate_course(120, passing_left)
).

always( test(associate_course) :-
        pre_test(associate_course, 'associate_course(-60, passing_right)')
        , associate_course(-60, passing_right)
).

always( test(associate_course) :-
        pre_test(associate_course, 'associate_course(170, leaving)')
        , associate_course(170, leaving)
).

always( test(associate_course) :-
        pre_test(associate_course, 'associate_course(-170, leaving)')
        , associate_course(-170, leaving)
).

//=============================================================================
```

```
//      have_course_change(+Agent, +Patient, ?Value)

// ------------------------------------------------------------------------
// some data needed for testing       0        -3 |
//           .                        1        -2 | |
//           .<10                     2        -1 | | |
//           .                        3  -2     0 + | | |
//           .                        4  -1     1 | + | | |-3
//            .      0 10         0   5   0     2 | | + | |-2
//             .     V  V         V   6   1     3 | | | + |-1
//      0-10>1  .....2  ++++++++++3   7   2         | | | + 0
//                                    8               | | | 1
//                                    9                 | | 2
//                                   10                  | 3
//                                      course speed/move_direction_is
//
   0 ! has_ground_geometry(obj_cc_2,   80,   0).
   0 ! has_ground_geometry(obj_cc_1,    0,   0).
   0 ! has_ground_geometry(obj_cc_3,  210,   0).
   1 ! has_ground_geometry(obj_cc_1,    0,   0).
   1 ! has_ground_geometry(obj_cc_2,   70,   0).
   1 ! has_ground_geometry(obj_cc_3,  200,   0).
   2 ! has_ground_geometry(obj_cc_1,    0,   0).
   2 ! has_ground_geometry(obj_cc_2,   60,   0).
   2 ! has_ground_geometry(obj_cc_3,  190,   0).
   3 ! has_ground_geometry(obj_cc_1,    0,   0).
   3 ! has_ground_geometry(obj_cc_2,   50,   0).
   3 ! has_ground_geometry(obj_cc_3,  180,   0).
   4 ! has_ground_geometry(obj_cc_1,    0,   0).
   4 ! has_ground_geometry(obj_cc_2,   40,   0).
   4 ! has_ground_geometry(obj_cc_3,  170,   0).
   5 ! has_ground_geometry(obj_cc_1,    0,   0).
   5 ! has_ground_geometry(obj_cc_2,   30,   0).
   5 ! has_ground_geometry(obj_cc_3,  160,   0).
   6 ! has_ground_geometry(obj_cc_1,    0,   0).
   6 ! has_ground_geometry(obj_cc_2,   20,  -1).
   6 ! has_ground_geometry(obj_cc_3,  150,   0).
   7 ! has_ground_geometry(obj_cc_1,    0,   0).
   7 ! has_ground_geometry(obj_cc_2,   10,  -2).
   7 ! has_ground_geometry(obj_cc_3,  140,   0).
   8 ! has_ground_geometry(obj_cc_1,    0,   0).
   8 ! has_ground_geometry(obj_cc_2,    0,  -3).
   8 ! has_ground_geometry(obj_cc_3,  130,   0).
   9 ! has_ground_geometry(obj_cc_1,    0,   0).
   9 ! has_ground_geometry(obj_cc_2,    0,  -4).
   9 ! has_ground_geometry(obj_cc_3,  120,   0).
  10 ! has_ground_geometry(obj_cc_1,    0,   0).
  10 ! has_ground_geometry(obj_cc_2,    0,  -5).
  10 ! has_ground_geometry(obj_cc_3,  110,   0).
// ------------------------------------------------------------------------

always( test(have_course_change) :-
        pre_test(have_course_change,
           '5 ? have_course_change(obj_cc_2, obj_cc_1, Val) => Val=changing')
        , 5 ? have_course_change(obj_cc_2, obj_cc_1, Val)
        , Val == changing
).

always( test(have_course_change) :-
        pre_test(have_course_change,
           '5 ? have_course_change(obj_cc_3, obj_cc_1, Val) => Val=constant')
        , 5 ? have_course_change(obj_cc_3, obj_cc_1, Val)
        , Val == constant
).
```

```
//============================================================================
//      associate_course_change(+Deriv, ?Value)

//============================================================================
//      have_difference_in_orientation(+Agent, +Patient, ?Value)

// ----------------------------------------------------------------------------
// some data needed for testing:              \  \
//                                             1  2  3  4
//                                                 /    \
always(has_view_direction(obj_ori_1, -135)).
always(has_view_direction(obj_ori_2, -135)).
always(has_view_direction(obj_ori_3,  135)).
always(has_view_direction(obj_ori_4,   45)).
// ----------------------------------------------------------------------------

always( test(have_difference_in_orientation) :-
        pre_test(have_difference_in_orientation,
           'have_difference_in_orientation(obj_ori_1, obj_ori_2, equal)')
        , have_difference_in_orientation(obj_ori_1, obj_ori_2, equal)
).

always( test(have_difference_in_orientation) :-
        pre_test(have_difference_in_orientation,
           'have_difference_in_orientation(obj_ori_4, obj_ori_3, crossing)')
        , have_difference_in_orientation(obj_ori_4, obj_ori_3, crossing)
).

always( test(have_difference_in_orientation) :-
        pre_test(have_difference_in_orientation,
           'have_difference_in_orientation(obj_ori_3, obj_ori_4, crossing)')
        , have_difference_in_orientation(obj_ori_3, obj_ori_4, crossing)
).

always( test(have_difference_in_orientation) :-
        pre_test(have_difference_in_orientation,
           'have_difference_in_orientation(obj_ori_1, obj_ori_4, opposite)')
        , have_difference_in_orientation(obj_ori_1, obj_ori_4, opposite)
).

//============================================================================
//      difference_in_orientation_is(+Agent, +Patient, ?Diff)

always( test(difference_in_orientation_is) :-
        pre_test(difference_in_orientation_is,
           'difference_in_orientation_is(obj_ori_1, obj_ori_2, Val) => Val=0')
        , difference_in_orientation_is(obj_ori_1, obj_ori_2, Val)
        , epsilon_neighborhood(0.001, Val, 0)
).

always( test(difference_in_orientation_is) :-
        pre_test(difference_in_orientation_is,
           'difference_in_orientation_is(obj_ori_1, obj_ori_3, Val) => Val=90')
        , difference_in_orientation_is(obj_ori_1, obj_ori_3, Val)
        , epsilon_neighborhood(0.001, Val, 90)
).

always( test(difference_in_orientation_is) :-
        pre_test(difference_in_orientation_is,
           'difference_in_orientation_is(obj_ori_3, obj_ori_1, Val) => Val=90')
        , difference_in_orientation_is(obj_ori_3, obj_ori_1, Val)
        , epsilon_neighborhood(0.001, Val, 90)
).

always( test(difference_in_orientation_is) :-
```

```
        pre_test(difference_in_orientation_is,
           'difference_in_orientation_is(obj_ori_4, obj_ori_2, Val) => Val=180')
        , difference_in_orientation_is(obj_ori_4, obj_ori_2, Val)
        , epsilon_neighborhood(0.001, Val, 180)
).

//=============================================================================
//      associate_difference_in_orientation(+Diff, ?Value)

always( test(associate_difference_in_orientation) :-
        pre_test(associate_difference_in_orientation,
                 'associate_difference_in_orientation(0, equal)')
        , associate_difference_in_orientation(0, equal)
).

always( test(associate_difference_in_orientation) :-
        pre_test(associate_difference_in_orientation,
                 'associate_difference_in_orientation(90, crossing)')
        , associate_difference_in_orientation(90, crossing)
).

always( test(associate_difference_in_orientation) :-
        pre_test(associate_difference_in_orientation,
                 'associate_difference_in_orientation(180, opposite)')
        , associate_difference_in_orientation(180, opposite)
).

//=============================================================================
//      have_difference_in_orientation_change(+Agent, +Patient, ?Value)

// ----------------------------------------------------------------------------
// some data needed for testing:  \>   \
//                                 1    2    3  4
//                                        </    \>
 0 ! has_view_direction(obj_oc_1, -135).
 0 ! has_view_direction(obj_oc_2, -135).
 0 ! has_view_direction(obj_oc_3,  135).
 0 ! has_view_direction(obj_oc_4,   45).
 1 ! has_view_direction(obj_oc_1, -123).
 1 ! has_view_direction(obj_oc_2, -135).
 1 ! has_view_direction(obj_oc_3,  147).
 1 ! has_view_direction(obj_oc_4,   33).
 2 ! has_view_direction(obj_oc_1, -111).
 2 ! has_view_direction(obj_oc_2, -135).
 2 ! has_view_direction(obj_oc_3,  158).
 2 ! has_view_direction(obj_oc_4,   21).
 3 ! has_view_direction(obj_oc_1, -100).
 3 ! has_view_direction(obj_oc_2, -135).
 3 ! has_view_direction(obj_oc_3,  170).
 3 ! has_view_direction(obj_oc_4,    9).
 4 ! has_view_direction(obj_oc_1,  -88).
 4 ! has_view_direction(obj_oc_2, -135).
 4 ! has_view_direction(obj_oc_3, -178).
 4 ! has_view_direction(obj_oc_4,   -3).
// ----------------------------------------------------------------------------

always( test(have_difference_in_orientation_change) :-
        pre_test(have_difference_in_orientation_change,
           '2 ? have_difference_in_orientation_change(obj_oc_1, obj_oc_2, Val)'+
           ' => Val=changing_away')
        , 2 ? have_difference_in_orientation_change(obj_oc_1, obj_oc_2, Val)
        , Val == changing_away
).

always( test(have_difference_in_orientation_change) :-
```

```
        pre_test(have_difference_in_orientation_change,
           '2 ? have_difference_in_orientation_change(obj_oc_3, obj_oc_4, Val)'+
           ' => Val=changing_away')
        , 2 ? have_difference_in_orientation_change(obj_oc_3, obj_oc_4, Val)
        , Val == changing_away
).

always( test(have_difference_in_orientation_change) :-
        pre_test(have_difference_in_orientation_change,
           '2 ? have_difference_in_orientation_change(obj_oc_1, obj_oc_3, Val)'+
           ' => Val=constant')
        , 2 ? have_difference_in_orientation_change(obj_oc_1, obj_oc_3, Val)
        , Val == constant
).

always( test(have_difference_in_orientation_change) :-
        pre_test(have_difference_in_orientation_change,
           '2 ? have_difference_in_orientation_change(obj_oc_1, obj_oc_4, Val)'+
           ' => Val=cchanging_towards')
        , 2 ? have_difference_in_orientation_change(obj_oc_1, obj_oc_4, Val)
        , Val == changing_towards
).

always( test(have_difference_in_orientation_change) :-
        pre_test(have_difference_in_orientation_change,
           '2 ? have_difference_in_orientation_change(obj_oc_2, obj_oc_3, Val)'+
           ' => Val=cchanging_towards')
        , 2 ? have_difference_in_orientation_change(obj_oc_2, obj_oc_3, Val)
        , Val == changing_towards
).

always( test(have_difference_in_orientation_change) :-
        pre_test(have_difference_in_orientation_change,
           '2 ? have_difference_in_orientation_change(obj_oc_2, obj_oc_4, Val)'+
           ' => Val=cchanging_towards')
        , 2 ? have_difference_in_orientation_change(obj_oc_2, obj_oc_4, Val)
        , Val == changing_towards
).

//=============================================================================
//      associate_difference_in_orientation_change(+Deriv, ?Value)

always( test(associate_difference_in_orientation_change) :-
        pre_test(associate_difference_in_orientation_change,
                 'associate_difference_in_orientation_change(0, constant)')
        , associate_difference_in_orientation_change(0, constant)
).

always( test(associate_difference_in_orientation_change) :-
        pre_test(associate_difference_in_orientation_change,
                 'associate_difference_in_orientation_change(90, changing_away)')
        , associate_difference_in_orientation_change(90, changing_away)
).

always( test(associate_difference_in_orientation_change) :-
        pre_test(associate_difference_in_orientation_change,
                 'associate_difference_in_orientation_change(-90, changing_towards)')
        , associate_difference_in_orientation_change(-90, changing_towards)
).

//=============================================================================
//      has_direction(+Agent, ?Value)

// ----------------------------------------------------------------------------
// some data needed for testing:               1-.....     .  -3  .
```

```
//                                                    .    .   .
//                                                  .     .    .
//                                                .      .     .
//                                              .       .      .
//                                           \ .        .       .
//                                            2         .        4
//                                                                \
always(has_view_direction(obj_reldir_1,    0)).
always(has_view_direction(obj_reldir_2, -135)).
always(has_view_direction(obj_reldir_3,  180)).
always(has_view_direction(obj_reldir_4,   45)).

  0 ! has_ground_geometry(obj_reldir_1,    0,    0).
  1 ! has_ground_geometry(obj_reldir_1,   10,    0).
  2 ! has_ground_geometry(obj_reldir_1,   20,    0).
  3 ! has_ground_geometry(obj_reldir_1,   30,    0).
  4 ! has_ground_geometry(obj_reldir_1,   40,    0).
  5 ! has_ground_geometry(obj_reldir_1,   50,    0).
  6 ! has_ground_geometry(obj_reldir_1,   60,    0).

  0 ! has_ground_geometry(obj_reldir_2,   60,   60).
  1 ! has_ground_geometry(obj_reldir_2,   70,   50).
  2 ! has_ground_geometry(obj_reldir_2,   80,   40).
  3 ! has_ground_geometry(obj_reldir_2,   90,   30).
  4 ! has_ground_geometry(obj_reldir_2,  100,   20).
  5 ! has_ground_geometry(obj_reldir_2,  110,   10).
  6 ! has_ground_geometry(obj_reldir_2,  120,    0).

  0 ! has_ground_geometry(obj_reldir_3,  160,    0).
  1 ! has_ground_geometry(obj_reldir_3,  160,   10).
  2 ! has_ground_geometry(obj_reldir_3,  160,   20).
  3 ! has_ground_geometry(obj_reldir_3,  160,   30).
  4 ! has_ground_geometry(obj_reldir_3,  160,   40).
  5 ! has_ground_geometry(obj_reldir_3,  160,   50).
  6 ! has_ground_geometry(obj_reldir_3,  160,   60).

  0 ! has_ground_geometry(obj_reldir_4,   60,  250).
  1 ! has_ground_geometry(obj_reldir_4,   50,  240).
  2 ! has_ground_geometry(obj_reldir_4,   40,  230).
  3 ! has_ground_geometry(obj_reldir_4,   30,  220).
  4 ! has_ground_geometry(obj_reldir_4,   20,  210).
  5 ! has_ground_geometry(obj_reldir_4,   10,  200).
  6 ! has_ground_geometry(obj_reldir_4,    0,  190).
// ------------------------------------------------------------------------------

always( test(has_direction) :-
        pre_test(has_direction, '3 ? has_direction(obj_reldir_1, straight)')
        , 3 ? has_direction(obj_reldir_1, straight)
).

always( test(has_direction) :-
        pre_test(has_direction, '3 ? has_direction(obj_reldir_2, right)')
        , 3 ? has_direction(obj_reldir_2, right)
).

always( test(has_direction) :-
        pre_test(has_direction, '3 ? has_direction(obj_reldir_3, left)')
        , 3 ? has_direction(obj_reldir_3, left)
).

always( test(has_direction) :-
        pre_test(has_direction, '3 ? has_direction(obj_reldir_4, back)')
        , 3 ? has_direction(obj_reldir_4, back)
).
```

```
//=============================================================================
//      associate_direction(+Dir, ?Value)

always( test(associate_direction) :-
        pre_test(associate_direction, 'associate_direction(0, straight)')
        , associate_direction(0, straight)
).

always( test(associate_direction) :-
        pre_test(associate_direction, 'associate_direction(90, right)')
        , associate_direction(90, right)
).

always( test(associate_direction) :-
        pre_test(associate_direction, 'associate_direction(-90, left)')
        , associate_direction(-90, left)
).

always( test(associate_direction) :-
        pre_test(associate_direction, 'associate_direction(180, back)')
        , associate_direction(180, back)
).

always( test(associate_direction) :-
        pre_test(associate_direction, 'associate_direction(-180, back)')
        , associate_direction(-180, back)
).
```

Anhang C

Zeitplan

Zeitraum	Arbeitsschritt
15. 08. - 21. 08.	Finden von Beispielbildfolgen und deren Aufbereitung. (z.B. VIRAT) Zielsetzung: Beschreibung der zu erkennenden Situationen.
22. 08. - 28. 08.	Theoretische Grundlagen, [Zim01], Kapitel 8. Dissertation [Are04] durchlesen und relevante Abschnitte schriftlich fixieren.
29. 08. - 04. 09.	Theoretische Grundlagen. Unterschied Wahrscheinlichkeit & Unscharfe Logik. Kapitel schreiben: verwandte Arbeiten und Einordnung in die Forschungslandschaft. Kapitel schreiben: Grundlagen.
05.09. - 18. 09.	Strukturierte Basisbibliotheken von Prädikaten erstellen. Testfälle für Basisbibliotheken erstellen.
19. 09. - 02. 10.	Konzeption und Implementierung einer Erweiterung des SGT-Editors um Unsicherheits- und Vagheitsbehandlung. Insbesondere auch deren Visualisierung.
03. 10. - 09. 10.	Kapitel schreiben: Umsetzung der Unsicherheits- und Vagheitsbehandlung bei der Situationsgraphtraversierung.
10. 10. - 16. 10.	Erstellen passender SGTs zu den in Woche 1 definierten Situationen.
17. 10. - 23. 10.	Puffer
31. 10. - 06. 11.	Kapitel Experimente und Auswertung.
07. 11. - 13. 11.	Abschluss der Arbeit. Vorbereitung des Vortrages Vortrag am IOSB Ettlingen.