Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Department Informatik
Arbeitsbereich Wissens- und Sprachverarbeitung

Diplomarbeit

Matching von graphbasierten Repräsentationen von Wissen zur Auflösung von Routenbeschreibungen

Lars Müllerchen
4lmuelle@informatik.uni-hamburg.de
Studiengang Informatik
Matrikelnummer: 5706746

November 2009

Erstgutachterin:
Dr. Carola Eschenbach

Zweitgutachter:
Prof. Bernd Neumann, Ph.D.

ZUSAMMENFASSUNG

Der Geometrische Agent ist ein Projekt am Department Informatik der Universität Hamburg, in dem anhand eines Agenten in einer simulierten Umgebung die kognitiven Prozesse von Navigation nach einer instruierten Route untersucht werden. Aspekte, die dabei behandelt werden, sind die Eigenschaften und Strukturen von Routen und Routeninstruktionen, das Verstehen und Verarbeiten von Routenbeschreibungen in unterschiedlichen Instruktionsformen, die Planung und Befolgung einer Route unter Unsicherheit und der Vergleich von internen Modellen der instruierten Route und der perzipierten Umgebung.

Diese Diplomarbeit beschäftigt sich mit dem Abgleich der internen, konzeptuellen Modelle des Geometrischen Agenten, die in einer graphbasierten Repräsentation in der Beschreibungsprache CRIL (**C**onceptual **R**oute **I**nstruction **L**anguage) vorliegen. Für den Abgleich werden zusammengesetzte Ähnlichkeitsmaße über den graphbasierten Repräsentationen verwendet, die auf einer paarweisen Zuordnung der Teilkomponenten basieren. Es werden mehrere Verfahren, optimale, paarweise Zuordnungen zwischen Mengen und zwischen Graphen zu bestimmen, vorgestellt, analysiert und Vor- und Nachteile der einzelnen Verfahren und ihre Eignung hinsichtlich der Aufgabenstellung des Geometrischen Agenten diskutiert.

DANKSAGUNG

Ich möchte mich bedanken bei

- meinen Eltern, insbesondere meiner Mutter Magdalena Müllerchen für die liebevolle Unterstützung
- meiner Schwester Claudia Müllerchen, die mich immer aufgemuntert hat und beim Korrekturlesen eine unverzichtbare Hilfe war
- meiner Freundin Ai Ogitani, die immer Rücksicht genommen hat, wenn ich mal wieder keine Zeit hatte, weil das Kapitel noch fertig werden sollte
- und natürlich meinen Betreuern Dr. Carola Eschenbach und Prof. Bernd Neumann, ohne die diese Diplomarbeit nicht möglich gewesen wäre.

INHALTSVERZEICHNIS

1. EINLEITUNG

Routenbeschreibungen sind ein Mittel, um Informationen über eine Umgebung zwischen Agenten auszutauschen. Sie werden von einem Agenten, dem die Umgebung bekannt ist, gegeben, damit ein anderer Agent, dem die Umgebung unbekannt ist, den Weg zu seinem Ziel findet. Routenbeschreibungen werden beispielsweise zwischen zwei Menschen benutzt, wenn man in einer fremden Stadt nach dem Weg fragt. Auch werden sie von Computern, künstlichen Agenten, an Menschen gegeben, wenn man ein Navigationssystem oder einen Routenplaner benutzt. Der Geometrische Agent ist eine Simulationsplattform im Rahmen eines Projekts der Universität Hamburg, mit der untersucht wird, welche Voraussetzungen gegeben sein müssen, damit ein künstlicher Agent eine von einem Menschen oder einem anderen Agenten gegebene Routeninstruktion verarbeiten und befolgen kann. Für die Befolgung einer instruierten Route baut der Geometrische Agent ein kognitives Modell der in der Routeninstruktion beschriebenen Umgebung auf und vergleicht dieses Modell mit seiner Wahrnehmung anhand von Ähnlichkeitsmaßen, während er der Route folgt. Ich beschäftige mich in dieser Diplomarbeit mit dem *Matching* genannten Verfahren, welches der Agent für den Vergleich seines Wissens über Route und Umgebung benutzt und untersuche Probleme, die den Umfang und die Expressivität der vom Agenten behandelbaren Routeninstruktionen beschränken.

1.1 Motivation

Das Matching soll es dem Agenten ermöglichen, Teile der Routeninstruktion in seiner Umgebung zu erkennen und auf diese Weise die vom Instruktor intendierte Fortsetzung der Route aus mehreren Möglichkeiten zu bestimmen. Wie gut die Identifizierung der richtigen Objekte der Umgebung ist, hängt dabei von zwei Faktoren ab: zum einen von der Eignung der Merkmale, auf denen der Vergleich basiert, und zum anderen von der Anzahl der informationstragenden Elemente, die die Entscheidungsgrundlage bilden. Selbst wenn die untersuchten Merkmale grundsätzlich geeignet sind, kann die Menge der in den Repräsentationen enthaltenen Information nicht ausreichend sein, um zwischen zwei Objekten in der Umgebung zu entscheiden. Weil die für das

Matching verwendeten Verfahren eine hohe Komplexität teilweise im Bereich der NP-vollständigen Probleme haben, sind die Repräsentationen der Routeninstruktion und der Umgebung mit steigendem Detailgrad und damit steigender Anzahl an informationstragenden Komponenten schnell nicht mehr behandelbar. Schon bei der Umsetzung der Routeninstruktion in das interne Modell ist deswegen eine restriktive Auswahl nötig, was dazu führen kann, dass einige Routenbeschreibungen nicht adäquat behandelt werden können. Durch eine Verbesserung der Komplexität und der Laufzeit können diese Restriktionen gelockert werden und eine größere Menge an Routenbeschreibungen benutzt werden. Ziel dieser Arbeit ist es also, nicht die grundsätzliche Arbeitsweise des Matchings und die verwendeten Ähnlichkeitsmaße zu verändern, sondern das Matching auf eine robustere Grundlage zu stellen, die eine geringere Komplexität und geringere Laufzeit für komplexe Problemstellungen aufweist, so dass die verwendeten Modelle eine detailliertere Beschreibung des Wissens des Geometrischen Agenten erlauben.

1.2 Aufbau der Arbeit

In Kapitel 2 werden zunächst die Grundlagen des Geometrischen Agenten vorgestellt. Abschnitt 2.1 gibt einen Überblick über den Geometrischen Agenten und ordnet das Matching, den Vergleich zwischen der Repräsentation der Routeninstruktion und der perzipierten Umgebung, als ein Kernstück in den Gesamtkontext des Agenten ein. Die Abschnitte 2.2 und 2.3 stellen die im Geometrischen Agenten verwendete Repräsentationssprache CRIL (**C**onceptual **R**oute **I**nstruction **L**anguage) vor und erläutern, wie das Wissen des Agenten in CRIL repräsentiert wird.

In Kapitel 3 werden dann Ähnlichkeitsmaße definiert, die der Geometrische Agent benutzt, um Teile seiner Wissensrepräsentation zu vergleichen. Diese Ähnlichkeitsmaße werden dann im Matching benutzt, welches in Kapitel 4 detailliert untersucht wird. Es werden die drei Komponenten des Matchings identifiziert und dann einzeln die bisher verwendeten Verfahren vorgestellt, analysiert, welche Probleme diese Verfahren haben, und alternative Lösungen vorgeschlagen. Die Ergebnisse werden dann in Kapitel 5 zusammengefasst.

2. DER GEOMETRISCHE AGENT

Dieses Kapitel beschäftigt sich mit den Grundlagen des Geometrischen Agenten. Es wird der Aufbau des Geometrischen Agenten erläutert und genauer auf die verwendeten Wissensrepräsentationen eingegangen, die in Kapitel 4 verglichen werden.

2.1 Überblick über den Geometrischen Agenten

Tschander et al. (2003) haben als Idee einer Simulationsplattform für Untersuchungen zu Routenbeschreibungen den Geometrischen Agenten entwickelt. Die theoretische Ausarbeitung und Implementierung dieses Geometrischen Agenten wurde seitdem in mehreren Diplom- und Studienarbeiten (u. a. Helwich, 2003; Bosch, 2004; Bittkowski, 2005) und in einigen Hauptstudiumsprojekten am Department Informatik der Universität Hamburg weitergeführt. Der Geometrische Agent simuliert einen künstlichen Agenten, wie etwa einen Roboter, der eine Routeninstruktion in einer Umgebung befolgt. Als Agenten auffassen kann man nach Russell und Norvig (2003) alles, was seine Umgebung über Sensoren perzipiert und mit der Umgebung über Aktuatoren agiert (siehe Abbildung 2.1.1a). Die Aktion ist dabei über die sogenannte Agentenfunktion bestimmt, die jeder Perzeptsequenz aller bisher perzipierten Perzepte eine Aktion zuweist.

Auch der Geometrische Agent lässt sich unter diesen Gesichtspunkten betrachten. Tschander et al. entwerfen einen Agenten, der Routeninstruktionen verarbeiten und dann befolgen kann. Es wird der Fall einer von der Route räumlich und zeitlich getrennten Routeninstruktion untersucht, welche im Voraus gegeben wird, ohne dass der Instruierende und der Agent die Route perzipieren können. Daraus ergeben sich zwei Phasen, die der Agent durchläuft, die Instruktionsphase und die Navigationsphase. Dieses Zwei-Phasen-Modell ist in Abbildung 2.1.1b dargestellt.

In der Instruktionsphase gibt der Instruktor, der mit der Umgebung vertraut ist, eine Routenbeschreibung. Der Agent nimmt die Instruktion vom Instruktor über geeignete Kanäle (Sensoren) auf. Die Instruktion kann dabei in unterschiedlicher Form, z. B. mündlich, schriflich, graphisch oder auch

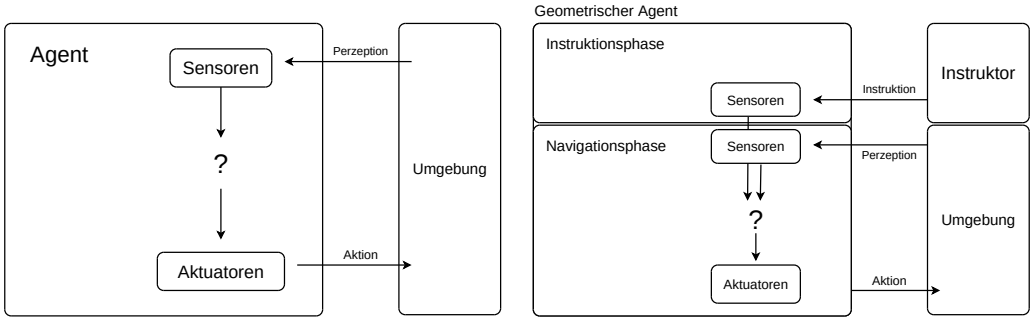


Abbildung 2.1.1: (a) Eine Darstellung der Agentencharakterisierung nach Russell und Norvig (2003) und (b) diese Charakterisierung angewendet auf das Zwei-Phasen-Modell des Geometrischen Agenten.

multimodal, vorliegen. Der Agent baut dann ein mentales Modell der Routeninstruktion auf und speichert die Instruktion in geeigneter Form als Repräsentation von räumlicher Information über die Route und eine Sequenz an durchzuführenden Schritten in der Wissensbasis. In der Navigationsphase orientiert sich der Agent in der Umgebung, indem er die perzipierte Umgebung mit der instruierten räumlichen Repräsentation der Route vergleicht, und verfolgt die Route, indem er die Aktionen aus der instruierten Aktionssequenz Schritt für Schritt abarbeitet. Aktionen sind zum einen durch die Art der Handlung (z. B. Fortbewegung, Orientierung) und zum anderen durch das Objekt der Handlung (z. B. den Pfad, der gegangen werden soll) charakterisiert. Um die Aktionen durchzuführen, müssen die für die Aktionen charakteristischen Objekte in der Umgebung durch Matching mit der perzipierten Welt identifiziert werden (Tschander et al., 2003). Ein detaillierteres Schema des Geometrischen Agenten findet sich in Abbildung 2.1.2.

Der Geometrische Agent ist eine Experimentierplattform, um Modellierung für die kognitiven Prozesse, die für das Befolgen von Routeninstruktionen nötig sind, und den Einfluss von Wahrnehmung auf die Fähigkeit zu Navigieren zu untersuchen. Um die Wahrnehmung (Perzeption) jederzeit kontrollieren zu können und auch um einige Grundprobleme der Robotik, die bei realen Robotern zwangsläufig auftreten, zu umgehen, interagiert der Geometrische Agent nicht mit einer realen Umgebung, sondern die Umgebung wird als virtuelle planare Ebene simuliert, in der sich der Agent anhand von vorgegebenen Wegen bewegt.

Wie in der Einleitung erwähnt wurde der Geometrische Agent vor allem in Diplomarbeiten ausgearbeitet. Bosch (2004) untersucht Routenbeschreibungen und liefert eine formale Grundlage zum einen für eine auf eukli-

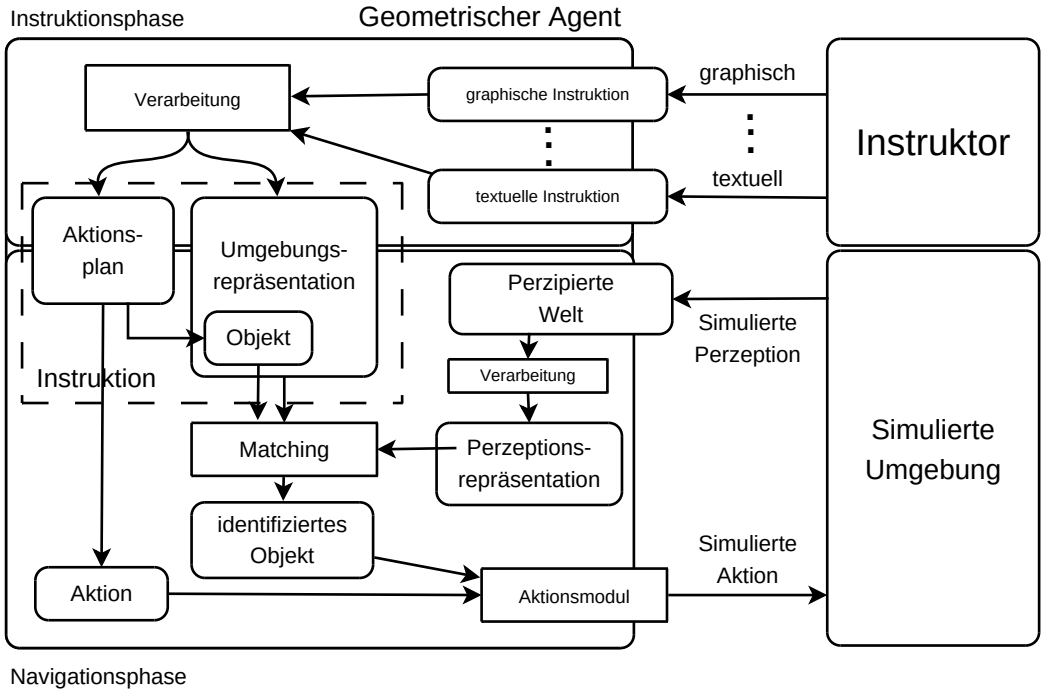


Abbildung 2.1.2: Eine detaillierte Sicht auf die Abläufe im Geometrischen Agenten.

discher Inzidenzgeometrie basierende Repräsentationssprache für räumliche Relationen und Aktionen in Routeninstruktionen und zum anderen für die Umsetzung von natürlichsprachlichen Routeninstruktionen in ein mentales Modell des Agenten. Bittkowski (2005) befasst sich mit der Interpretation und Umsetzung des Aktionsplans, mit der Planung, Auswahl und Durchführung von Aktionen sowie der Entscheidung unter Unsicherheit zwischen mehreren Fortsetzungen der Route. Helwich (2003) beschäftigt sich mit dem Matchingvorgang, dem Abgleich von Repräsentationen der Instruktion und der Perzeption, und schlägt Ähnlichkeitsmaße vor, um Teile der beiden Repräsentationen miteinander zu vergleichen, ähnliche Teile zu erkennen und so die identifizierten Objekte für die Aktionsdurchführung zu liefern.

In der vorliegenden Arbeit ist das Berechnungsverfahren für ähnliche Teile der Repräsentation Schwerpunkt.

Die für die Repräsentation des Wissens im Geometrischen Agenten benutzte Repräsentationssprache ist die Conceptual Route Instruction Language (CRIL) (vgl. Tschander et al., 2003), die in Abschnitt 2.2 definiert und dann für die Modellierung von räumlichen Informationen einer Route und der Umgebung in Abschnitt 2.3 benutzt wird. Beschreibungen für Objekte der Routeninstruktion und der perzipierten Umgebung, die beim Matching

verglichen werden, sind in CRIL als Graphen repräsentiert. Das Matching ist also ein Vorgang, in dem die Ähnlichkeit zwischen zwei Graphen bestimmt wird, um auf dieser Grundlage entscheiden zu können, ob in beiden Graphen dasselbe Objekt beschrieben wird. Dabei werden einzelne Teile der Graphen einander zugeordnet, so dass Entsprechungen der Beschreibungen in der Welt entstehen. Zwei einander im Matching zugeordnete Graphen, Knoten oder Kanten eines Graphen werden im Weiteren Match genannt.

Der Vergleich von Graphen ist eine berechnungstechnisch komplexe Problemstellung (vgl. Lewis, 1978; Poole und Campbell, 1995; Zhong et al., 2002). In der Praxis des Geometrischen Agenten treten auf Grund der Komplexität und struktureller Einschränkungen der Beschreibungen, die unternommen wurden, um das Komplexitätsproblem zu begrenzen, Probleme mit expressiven Routenbeschreibungen auf. So ist die Zahl der beschreibenden Eigenschaften von Objekten der Routenbeschreibung und die Benutzung von räumlichen Relationen zwischen Objekten in vom Geometrischen Agenten behandelbaren Routenbeschreibungen zu Beginn dieser Diplomarbeit stark eingeschränkt. Das Ziel dieser Diplomarbeit besteht darin, es dem Agenten durch Analyse der Probleme und der Verwendung besserer Berechnungsverfahren für die Ähnlichkeiten möglich zu machen, auch expressivere Routenbeschreibungen zu befolgen.

2.2 CRIL

CRIL ist die interne Sprache des Geometrischen Agenten (Tschander et al., 2003). Sie wird sowohl für die konzeptuelle Repräsentation der Instruktion als auch in der Navigationsphase für die Repräsentation der Umgebung benutzt. Eine CRIL-Repräsentation einer Routeninstruktion besteht aus zwei Teilen, einem CRIL-Graphen, der Objekte der Route und ihre Relation zueinander beinhaltet, und einem Aktionsplan, der die Aktionen enthält, die der Agent für die Befolgung der Route ausführen soll.

In CRIL-Graphen ist sowohl das instruierte als auch das perzipierte Wissen des Geometrischen Agenten über die Umgebung des Agenten repräsentiert. CRIL-Graphen bestehen aus CRIL-Knoten, die Entitäten der Welt repräsentieren, und (gerichteten) Kanten, die Knoten und die dadurch repräsentierten Entitäten miteinander in Relation setzen. Der Agent benutzt zwei Taxonomien, T_A und T_R, als Grundlage für sein Wissen über die Welt. T_A beschreibt das Verhältnis zwischen Konzepten, die Attribute der CRIL-Knoten beschreiben, und T_R beschreibt das Verhältnis zwischen Konzepten, die CRIL-Relationen zwischen den Knoten beschreiben, jeweils über Subsumptions- ($\leq$) und Exklusionsbeziehungen ($\leftrightarrow$). Ein Überblick über die für die Modellierung

der Anwendungsdomäne verwendeten Konzepte und typisch vorkommende CRIL-Graphen wird in Abschnitt 2.3 gegeben.

Der Vergleich von zwei Repräsentationen in CRIL erfolgt auf der Grundlage eines Ähnlichkeitsmaßes über Taxonomien. Die Aufteilung des Weltwissens auf zwei Taxonomien ist nicht unbedingt notwendig, doch ist die Unterscheidung sinnvoll für eine klare Trennung zwischen einstelligen Knotenprädikaten und mehrstelligen Relationenprädikaten und hat auch berechnungstechnische Vorteile, da ein Vergleich zwischen den beiden Konzeptarten nicht nötig ist (vergleiche dazu auch Kapitel 3). Der Agent benutzt sowohl in der Navigationsphase als auch in der Instruktionsphase dieselben Taxonomien und deswegen lassen sich CRIL-Graphen leicht durch Abbildung aus referentiellen Netzen erzeugen, die in der Sprachverarbeitung der Instruktionsphase verwendet werden (siehe Helwich, 2003; Habel, 1988).

Ein CRIL-Knoten kn beschreibt ein Objekt in der Welt des Agenten zum einen über Attributkonzepte der Taxonomie T_A und zum anderen über Namen. kn hat mindestens eine Sorte und eine beliebige Anzahl an Namen und Eigenschaften.

Definition 2.2.1
Ein CRIL-Knoten $kn = (\textsc{Sor}, \textsc{Eig}, \textsc{Nam})$ *besteht aus Sorten, Eigenschaften und Namen. Sorten und Eigenschaften sind Konzepte einer Taxonomie* $T_A = (S_A, \leq_A, \leftrightarrow_A)$, $\textsc{Sor} \neq \emptyset \subseteq S_A$ *und* $\textsc{Eig} \subseteq S_A$, *und Namen sind Wörter über einem Alphabet,* $\textsc{Nam} \subseteq \Sigma^+$.

Durch Kanten sind im CRIL-Graphen Relationen zwischen CRIL-Knoten repräsentiert. Relationen sind definiert durch das Konzept einer Taxonomie und n CRIL-Knoten mit $n \geq 2$, die in dieser Relation stehen. Da mehrstellige Relationen in CRIL erlaubt sind, handelt es sich bei CRIL-Graphen um Hypergraphen, die Kanten zwischen mehreren Knoten aufweisen. Außerdem sind mehrere Relationen zwischen den gleichen Knoten möglich. Um die Unterscheidung zwischen verschiedenen Kanten mit denselben Knoten deutlich zu machen, wird im Folgenden nicht mehr von Kanten, die Relationen repräsentieren, sondern nur noch von CRIL-Relationen gesprochen.

Definition 2.2.2
Eine n-stellige CRIL-Relation $r = (k_r, kn_1, \ldots, kn_n)$ *besteht aus einem Konzept* $k_r \in S_R$ *der Taxonomie* $T_R(S_R, \leq_R, \leftrightarrow_R)$ *und n CRIL-Knoten* $kn_1, \ldots, kn_n$.
Weiterhin sei knoten eine Funktion, welche die Menge der Knoten einer CRIL-Relation r zurückgibt.

$$knoten(r) = \{kn_1, kn_2, \ldots, kn_n\}$$

CRIL-Graphen bestehen aus CRIL-Knoten und CRIL-Relationen zwischen diesen Knoten.

Definition 2.2.3
Ein CRIL-Graph $G = (K, R, T_A, T_R)$ ist bestimmt durch die beiden Taxonomien $T_A(S_A, \leq_A, \leftrightarrow_A)$ und $T_R(S_R, \leq_R, \leftrightarrow_R)$, eine Menge K von CRIL-Knoten, deren Sorten und Eigenschaften durch T_A bestimmt sind, und eine Menge $R \subseteq S_R \times K^2 \cup S_R \times K^3 \cup \ldots$ von CRIL-Relationen.

Wie aus der Definition hervorgeht, können nur Relationen Teil des CRIL-Graphen sein, deren CRIL-Knoten in K sind. Helwich (2003) definiert CRIL-Graphen etwas anders und benutzt für die Beschreibung der CRIL-Relationen eine Operatorfunktion $op : E \subseteq (K^2 \cup K^3 \cup \ldots) \rightarrow S_R$ als Teil des CRIL-Graphen, die Kanten auf Relationenkonzepte der Taxonomie T_R abbildet. Dies impliziert jedoch, dass mehrere Relationen mit unterschiedlichen Konzepten zwischen denselben Knoten nicht möglich sind, und deswegen wurde hier die explizite Definition 2.2.2 für CRIL-Relationen bevorzugt.

Der Aktionsplan ist eine Sequenz von CRIL-Aktionen. Diese bestehen grundsätzlich aus einem Bezeichner — einem imperativen Verb — für eine Handlung, die der Agent ausführen kann, und dem Objekt der Handlung: `VERB` *objekt*. *objekt* ist dabei ein CRIL-Knoten aus der Instruktionsrepräsentation, der ein Objekt in der Welt repräsentiert, welches, um die Aktion ausführen zu können, in der perzipierten Umgebung erkannt werden muss. Der Aktionsplan setzt so durch die Sequenz der Aktionen die Beschreibungen der Route aus der Routeninstruktion mit Handlungen des Agenten in einen räumlichen und zeitlichen Zusammenhang. Auf den Aktionsplan wird im Weiteren nicht genauer eingegangen, da er für den Schwerpunkt dieser Arbeit keine weitere Rolle mehr spielt. Für einen Überblick über Planung und Durchführung von Handlungen sei auf Bittkowski (2005) verwiesen.

2.3 Repräsentation von Routen- und Umgebungswissen

In diesem Abschnitt wird dargestellt, wie die Beschreibungssprache des Agenten, CRIL, zur Repräsentation der instruierten Route und der perzipierten Umgebung benutzt wird. Zunächst wird die Struktur von Routeninstruktionen erläutert und dann die Konzepte eingeführt, die für eine Repräsentation von Routeninstruktion im geometrischen Agenten verwendet werden, sowie die typischen Strukturen der repräsentierten Routeninstruktion vorgestellt. Abschließend wird auf die Repräsentation der perzipierten Umgebung eingegangen.

2.3.1 Routeninstruktion

Eine Routeninstruktion (oder auch Routenbeschreibung) ist eine Anweisung eines Instruktors an einen Instruierten, wie man von einem Punkt A an einen Punkt B gelangt. Routeninstruktionen sind gut untersucht (vgl. Tschander et al., 2003; Maaß, 1993; Tversky und Lee, 1999) und es herrscht weitestgehend Einigkeit darüber, dass sie strukturelle Übereinstimmungen aufweisen. Diese gemeinsamen Strukturen sind auch bis auf wenige Unterschiede unabhängig vom Modus der Kommunikation zwischen Instruktor und Instruiertem vorhanden (Tversky und Lee, 1999). Routeninstruktionen haben einen Anfang, durch den der Startpunkt und die Ausrichtung zu Beginn der Route festgelegt werden, und ein Ende, durch das angezeigt wird, was das Ziel der Route ist und wann es erreicht wird. Dazwischen liegen mehrere Abschnitte, im Folgenden *Routenabschnitte* genannt.

Die Routenabschnitte verbinden Start- und Endpunkt der Route, beschreiben die Route abschnittsweise durch *Landmarken*, *Entscheidungspunkte* und räumliche Relationen zwischen diesen und geben eine zeitliche Abfolge für Aktionen des Instruierten an, die dadurch gegeben ist, dass Landmarken und Entscheidungspunkte in derselben Abfolge in der Routenbeschreibung vorkommen, wie sie ein *imaginärer Wanderer* (vgl. Klein, 1979; Bosch, 2004) antreffen würde, während er der Route folgt. Entscheidungspunkte sind hierbei Punkte, an denen sich der Instruierte zwischen mehreren Möglichkeiten entscheiden muss, seinen Weg fortzusetzen. Sie werden durch ihre räumliche Relation zu markanten Objekten in der Umgebung beschrieben, die Landmarken genannt werden (Tschander et al., 2003). Landmarken dienen in der Routeninstruktion darüber hinaus auch dazu, bei längeren Wegstücken sicherzustellen, dass der imaginäre Wanderer sich noch auf dem richtigen Weg befindet, indem sie als Orientierungspunkte benutzt werden.

2.3.2 Instruktionsrepräsentation

Die in den Routenabschnitten gegebenen Informationen werden benutzt, um ein internes Modell der Route im Geometrischen Agenten zu erstellen. Auf der einen Seite sind dies die Aktionen, die in den einzelnen Routenabschnitten vom imaginären Wanderer getätigt werden, aus denen der Aktionsplan des Agenten aufgestellt wird. Auf der anderen Seite sind dies räumliche Informationen über die einzelnen Abschnitte, die Landmarken und Entscheidungspunkte in Beziehung zur Route beschreiben. Beides wird in CRIL repräsentiert.

Bittkowski (2005) identifiziert vier Typen von CRIL-Aktionen, `GO`, `CH_ORIENT`, `BE_AT` und `VIEW`. CRIL-Aktionen haben einen CRIL-Knoten als

Ziel, dessen repräsentiertes Objekt in der Umgebung des zugehörigen Routenabschnitts beschrieben ist. Diese instruierte Beschreibung der Umgebung des Agenten in einem bestimmten Routenabschnitt wird in einem CRIL-Graph repräsentiert. Die Beschreibung aller räumlichen Relationen zwischen Landmarken und Routenabschnitten der gesamten Route sind im Instruktions-CRIL-Graphen[1] zusammengefasst.

Im Folgenden werden jetzt einige Konzepte eingeführt, die für die Beschreibung der Informationen der Routenabschnitte wichtig sind und Eingang in die Taxonomien von Prädikaten und Relationen finden, welche die Grundlage der CRIL-Modellierung bilden.

Da der Geometrische Agent sich nur auf durch die Simulation fest vorgegebenen Wegen, **Tracks**[2], bewegen kann, verläuft die Route über einen oder mehrere **Tracks** zum Ziel. **Tracks** sind zu unterscheiden von **Pfaden**. Während **Tracks** ungerichtet sind und Stücke von Straßen und Wegen repräsentieren, ist ein **Pfad** gerichtet und steht für eine mögliche Bewegung über einen oder mehrere **Tracks** vom derzeitigen Standpunkt des Agenten aus. Einzelne Abschnitte der Route können als **Pfad** vom Standpunkt des imaginären Wanderers, bzw. des Agenten, in der Route betrachtet werden[3]. **Pfade** und **Tracks** stehen in Beziehung zu Punkten von ihnen, die als **Position** bezeichnet werden. Es gibt die **Punkt**-Relationen **Endpunkt** und **Inpunkt**. Jeder **Track** und **Pfad** hat zwei **Endpunkte** und beliebig viele **Inpunkte**. Da **Pfade** gerichtet sind, werden die **Endpunkt**-Positionen durch die spezielleren Relationen **Startpunkt** und **Zielpunkt** unterschieden. Ein weiterer wichtiger Teil der Umgebung des Geometrischen Agenten sind **Landmarken**. **Landmarke** ist ein ziemlich weitgefasster Begriff und umfasst unter anderem **Bäume** und **Gebäude**. Auch **Tracks** zählen zu den **Landmarken**, da sie ebenfalls zur räumlichen Einbettung der Route in die Umgebung dienen können. **Landmarken** sind in der Regel durch zusätzliche Eigenschaftsprädikate beschrieben, hier sei als Bespiel **hoch** bzw. **niedrig** genannt.

Die räumlichen Beziehungen zwischen Objekten der Routeninstruktion werden durch **Regionen** dargestellt (vgl. Tschander et al., 2003). **Regionen** sind über eine räumliche Relation definiert. Dies kann eine von den

[1] Der Einfachheit halber im Weiteren auch *Instruktionsgraph* genannt.

[2] Track wird hier als Oberbegriff für Straßen, Wege und alle anderen ähnlichen Teile der Umgebung gewählt, auf denen sich der Agent fortbewegen kann.

[3] Ein Routenabschnitt, der als Pfad repräsentiert wird, enthält implizit den Entscheidungspunkt, an dem der Agent zwischen mehreren möglichen Fortsetzungen der Route entscheidet, und die Fortsetzung der Route von diesem Entscheidungspunkt aus. Es werden im internen Modell des Geometrischen Agenten also nicht die Entscheidungspunkte der Route und die Handlungen an diesen Entscheidungspunkten, sondern eher die Entscheidungen selbst beschrieben.

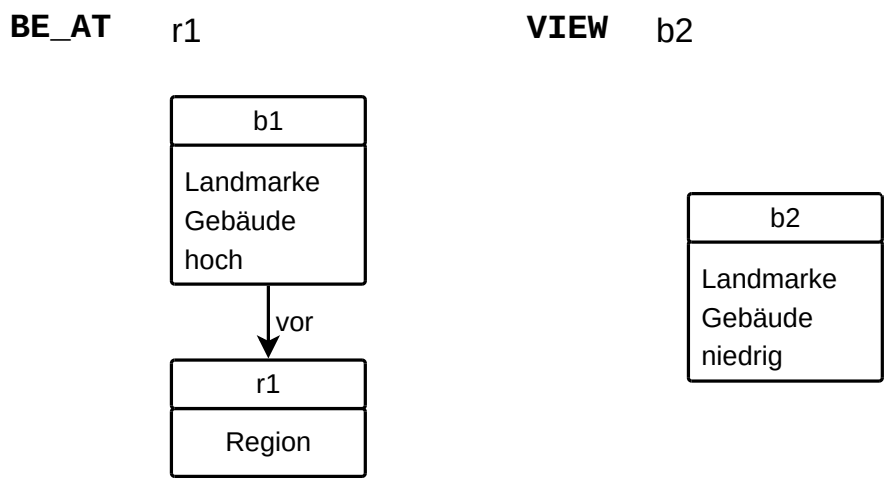


Abbildung 2.3.1: Typische CRIL-Repräsentationen, die die charakteristischen Objekte der Aktionen beschreiben, für die Aktionsarten `VIEW` und `BE_AT`.

fünf oft vorkommenden räumlichen Relationen zu einer einzelnen Landmarke, **links_von**, **rechts_von**, **vor**, **hinter** und **bei**, oder in einer Relation zu zwei Landmarken, z. B. **zwischen**, sein. Ebenso möglich ist eine Bezugnahme auf die derzeitige Ausrichtung des Geometrischen Agenten anstelle einer Landmarke. Die Objekte, mit denen die **Region** in der räumlichen Relation stehen, werden Referenzobjekte genannt. Die räumliche Relation von beschriebenen Objekten zu den Referenzobjekten einer **Region** wird über die Lokalisation in der **Region** oder außerhalb der **Region** angegeben. In der Regel geschieht dies für Positionen und Pfade. Ein CRIL-Knoten ist **lokalisiert** oder **nicht_lokalisiert** in einer **Region**.

Es sei angemerkt, dass dieser kurze Überblick über die Konzepte des Geometrischen Agenten keinen Anspruch auf Vollständigkeit hat — insbesondere wurde hier nicht auf Exklusivität zwischen Konzepten eingegangen — aber für eine Einschätzung typisch vorkommender CRIL-Graphstrukturen der Instruktion ist er ausreichend. Der Instruktionsgraph besteht aus einzelnen Teilen, welche Ziele der Aktionen beschreiben. Je nach Art der Aktion und damit des Ziels finden sich unterschiedliche Strukturen.

Anhand der Art des Ziels der Aktion lassen sich die Aktionen in zwei Gruppen aufteilen. `VIEW` und `BE_AT` sind Orientierungsaktionen, die anhand von Landmarken sicherstellen, dass der Agent auf dem richtigen Weg ist. `VIEW` hat eine Landmarke als Objekt, die der Agent von seinem Standpunkt sehen soll und die in vielen Fällen nur durch ihre Attribute beschrieben ist. `BE_AT` hat eine Region einer Landmarke, in der sich der Agent befinden soll, als Objekt und ist vor allem durch die räumliche Relation sowie die Attri-

bute der Landmarke charakterisiert. Für die einfachsten Fälle bestehen die Graphen der beiden Orientierungsaktionen also aus einem, respektive zwei CRIL-Knoten (siehe Abbildung 2.3.1).

`GO` und `CH_ORIENT` sind Bewegungsaktionen, die einen Orts- oder Ausrichtungswechsel des Agenten veranlassen. Sie haben einen Pfad als Objekt. Pfade werden durch komplexe Teilgraphen beschrieben, welche die räumlichen Relationen zwischen Landmarken und Pfad durch Regionen beschreiben. In Tschander et al. (2003) werden dafür drei mögliche komplexe Inzidenzbeziehungen zwischen Pfaden und Regionen identifiziert: `TO`, `FROM` und `VIA`. `TO` gilt für Pfade, die zu einer Region hinführen. Der Startpunkt ist außerhalb der Region, aber der Zielpunkt ist innerhalb. `FROM` hingegen gilt für Pfade aus einer Region heraus, wo der Startpunkt enthalten ist, der Zielpunkt jedoch nicht in der Region ist. `VIA` gilt für Pfade, die durch die Region hindurch gehen. Start- und Zielpunkt sind außerhalb der Region und ein Inpunkt des Pfades liegt in der Region. Beispiele für diese drei Beziehungen zwischen Pfad und Region sind in Abbildung 2.3.2 angegeben.

Die Beispiele zeigen, wie ein Pfad repräsentiert von drei, bzw. vier Knoten durch eine Region und ihr Referenzobjekt beschrieben ist. Ein mögliches Referenzobjekt wäre auch der Startpunk des Pfades, was vom Geometrischen Agenten als Bezug auf die aktuelle Ausrichtung interpretiert wird[4]. Eine Umgebungsrepräsentation für einen Pfad, der durch eine Region beschrieben ist, besteht also aus vier bis sechs CRIL-Knoten. Ebenso ist es möglich, dass der Pfad nicht nur durch eine, sondern durch mehrere Regionen beschrieben wird, so dass sich die Anzahl der Knoten um ein bis drei Knoten pro Region erhöht. Eine wiederkehrende Struktur, die erkennbar ist, ist ein Zyklus aus vier Knoten, da je zwei Positionsknoten sowohl Lokalisationsrelationen zu einem Regionsknoten sowie Punktrelationen zu dem Pfadknoten haben. Ein Instruktionsteilgraph, der das Objekt einer Bewegungsaktion beschreibt, besteht also aus mehreren solcher Zyklen, die sich mindestens im Pfadknoten überschneiden, plus den Referenzobjekten der Regionen.

2.3.3 Perzeptionsrepräsentation

Die Perzeptionsrepräsentation verwendet dieselben Konzepte zur Beschreibung wie die Instruktionsrepräsentation. Während der Navigationsphase wird ein Perzeptions-CRIL-Graph[5] aus der simulierten Umgebung auf Basis der durch den aktuellen Standpunkt des Agenten wahrnehmbaren Objekte erstellt. Direkt perzipieren kann der Geometrische Agent alle aktuell sichtbaren Landmarken und das Track-Netz, das alle wahrnehmbaren Tracks

[4] Bsp.: Repräsentation von *links* bei „Folge dem Pfad nach links“

[5] Der Einfachheit halber im Weiteren auch *Perzeptionsgraph* genannt.

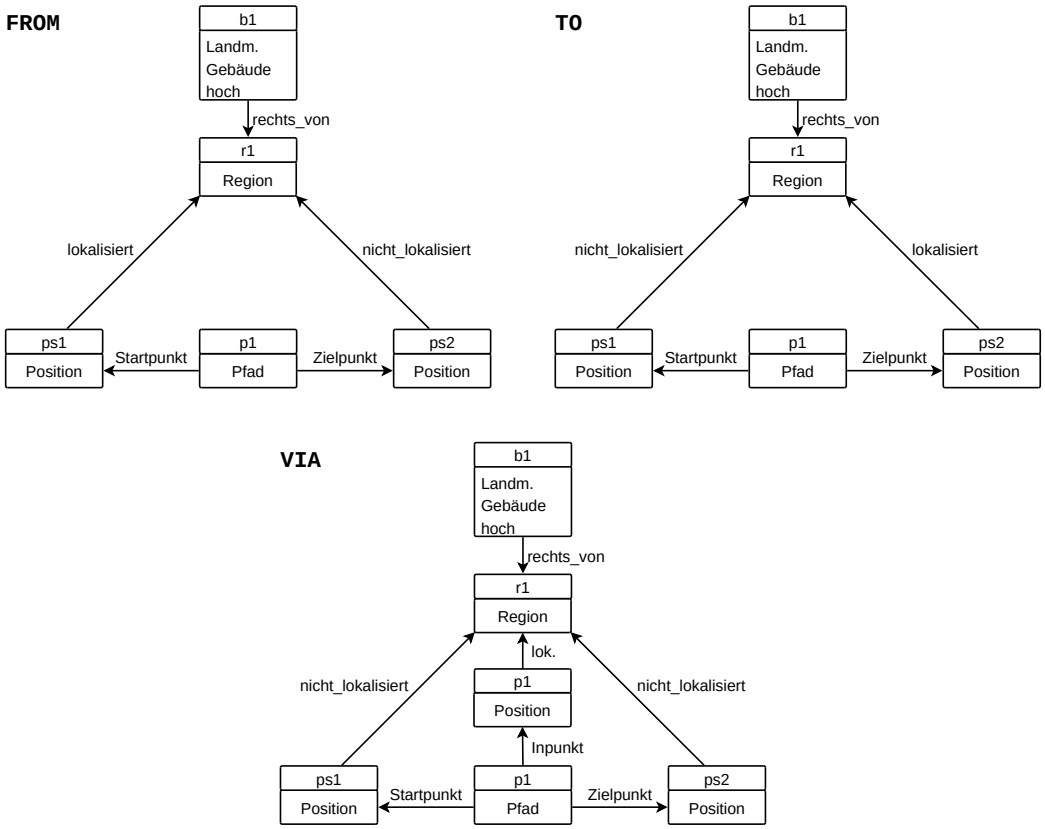


Abbildung 2.3.2: Typische CRIL-Repräsentationen für die Pfad-Region-Beziehungen `FROM`, `TO` und `VIA`

und ihre Endpunkte enthält und über gemeinsame Endpunkte Verbindungen zwischen Tracks herstellt. Um dieses Rohgerüst der Perzeption mit der Instruktion abgleichen zu können, muss der Perzeptionsgraph mit Regionen und Pfaden angereichert werden.

Pfade können über das Track-Netz bestimmt werden. Ausgehend vom Standpunkt des Agenten im Perzeptionsgraph können alle über das Track-Netz erreichbaren Positionsknoten bestimmt und je ein Pfad mit dem aktuellen Standpunkt als Startpunkt und dem erreichbaren Knoten als Zielpunkt zum Perzeptionsgraphen hinzugefügt werden. Damit sind alle Möglichkeiten des Geometrischen Agenten, seinen Weg fortzusetzen, im Perzeptionsgraph als Pfade repräsentiert. Dabei wird implizit die Annahme gemacht, dass der aktuelle Standpunkt des Agenten im Perzeptionsgraph derselbe Standpunkt ist, der in der Routeninstruktion als Standpunkt des imaginären Wanderers bei Durchführung der aktuellen Aktion angenommen wird (vgl. Bittkowski, 2005).

Um die Regionen zu berechnen und zu bestimmen, wie Objekte der Umgebung in den Regionen lokalisiert sind, greift der Geometrische Agent auf die geometrische Repräsentation der simulierten Umgebung zu. Dadurch können für jede Region Lokalisierungsrelationen zu allen Perzeptionsknoten im Graphen hinzugefügt werden. Die Anreicherung mit Regionen geschieht im Vorverarbeitungsschritt des Matchings, da es sinnvoll ist, nicht alle möglichen Regionenknoten hinzuzufügen, sondern eine Auswahl zu treffen. Dies wird in Abschnitt 4.3 näher behandelt.

3. ÄHNLICHKEITSBERECHNUNG IM AGENTEN

Um in CRIL repräsentierte Elemente der Instruktion und der Perzeption zu vergleichen, werden in Helwich (2003) Ähnlichkeitsmaße für Graphen, Relationen, Knoten und Konzepte von CRIL eingeführt. Diese Ähnlichkeitsmaße werden bis auf kleine Änderungen immer noch im Geometrischen Agenten benutzt. Weil Graphen aus Knoten und Relationen bestehen und Knoten eine Sammlung von Konzepten sind sowie Relationen durch ein Konzept definiert sind, werden die Ähnlichkeiten der CRIL-Repräsentationen aus den Einzelähnlichkeiten ihrer Teilkomponenten aufbauend auf einem Ähnlichkeitsmaß für Konzepte bestimmt.

3.1 Ähnlichkeit von Konzepten einer Taxonomie

Das Ähnlichkeitsmaß für Konzepte einer Taxonomie ist die Grundlage für alle Ähnlichkeitsberechnungen im Geometrischen Agenten. Die Subsumptionsrelation einer Taxonomie zeigt an, dass zwei Konzepte der Taxonomie gemeinsame Information enthalten. Dabei enthält das subsumierte Konzept mehr oder speziellere Information als das subsumierende. Nach Helwich (2003) ist die gemeinsame Information, die zwei Konzepte enthalten, ein Maß für die Ähnlichkeit der beiden Konzepte. Helwich führt dann den Begriff des spezifischen Vaters für die Bestimmung des gemeinsamen Informationsgehaltes ein. Der spezifische Vater zweier Konzepte A und B ist das Konzept, welches sowohl A als auch B subsumiert, und möglichst tief in der Subsumptionshierarchie liegt.

Definition 3.1.1
Ein Konzept k ist spezifischer Vater von zwei Konzepten $k_1, k_2 \in S$ einer Taxonomie $T = (S, \leq, \leftrightarrow)$, wenn gilt:

$$k_1 \leq k \wedge k_2 \leq k \wedge \forall k' \in S : ((k_1 \leq k' \wedge k_2 \leq k') \rightarrow k \leq k')$$

Ein spezifischer Vater ist das in der Subsumptionshierarchie erste Konzept, das beide Konzepte subsumiert, und enthält deshalb die gesamte Information, die beiden Konzepten gemeinsam ist. Wenn zu zwei Konzepten der spezifische Vater bekannt ist, weiß man also, welche Information in beiden

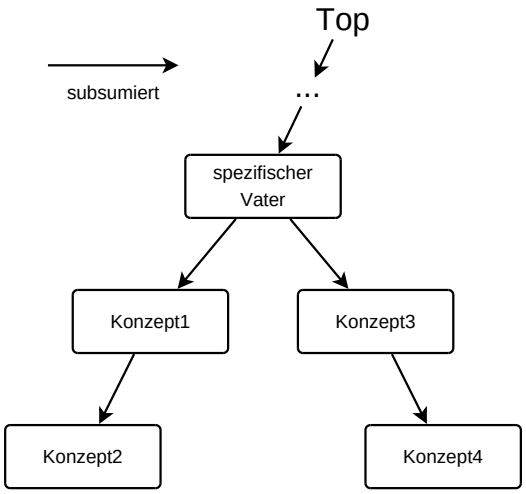


Abbildung 3.1.1: Der spezifische Vater zweier Konzepte ist das Konzept, welches beide Konzepte subsumiert und möglichst tief in der Taxonomie liegt. (hier für die Paare $(Konzept1, Konzept3)$, $(Konzept1, Konzept4)$, $(Konzept2, Konzept3)$) und $(Konzept2, Konzept4)$.

Konzepten vorhanden ist und in welcher Information sie sich unterscheiden. Diese Gemeinsamkeit bzw. der Unterschied muß jetzt quantisiert werden, um ein Maß der Ähnlichkeit zwischen zwei Konzepten zu bestimmen. Dazu definiert Helwich (2003) ein Distanzmaß zwischen Konzepten der Taxonomie. Zunächst wird ein Graph definiert, dessen Kanten durch die Subsumptionsrelation der Taxonomie ohne die transitive und reflexive Hülle bestimmt sind.

Definition 3.1.2
Ein Graph $G_{T\leq} = (V, E)$ ist der Subsumptionsgraph der Taxonomie $T(S, \leq, \leftrightarrow)$, wenn gilt:

$$V = S$$
$$\wedge$$
$$E = \{(k_1, k_2) \in \leq \mid k_1 \neq k_2 \wedge \neg\exists k : k_1 < k < k_2\}$$

In diesem Graphen liegen auf dem minimalen Pfad zwischen zwei Konzepten $k1$, $k2$ mit $k_1 \leq k_2$ genau die Konzepte, die k_2 subsumieren und von $k1$ subsumiert werden. Mit jedem Konzept näher an k_1 wird die Information des Konzepts k_2 weiter generalisiert — das bedeutet weniger Informationsgehalt als k_2 — mit jedem Konzept näher an k_2 wird die Information des Konzepts k_1 spezialisert — das bedeutet mehr Informationsgehalt. Sind k_1 und k_2 gleich, so hat der Pfad die Länge 0. Also ist die Länge der Pfade zwischen Konzepten im Subsumptionsgraphen ein Maß für den Unterschied

des Informationsgehalts. Für ein gleichmäßiges Maß ist allerdings Voraussetzung, dass die Granularität der Taxonomie gleichmäßig ist und in etwa die gleiche Spezialisierung der Konzepte durch eine direkte Subsumption zweier Konzepte ausgedrückt ist.

Definition 3.1.3
Die Entfernung zweier Konzepte k_1, k_2 einer Taxonomie $T(S, \leq, \leftrightarrow)$ mit $k_1 \leq k_2$ ist definiert als Länge des kürzesten Pfads zwischen k_1 und k_2 im Subsumptionsgraphen $G_{T\leq}$ der Taxonomie.

$$entfernung\,(k_1, k_2) \stackrel{def}{=} min\left(\left|pfad_{G_{T\leq}}(k_1, k_2)\right|\right)$$

Die Entfernung eines Konzepts k von Top heißt Tiefe:

$$tiefe(k) \stackrel{def}{=} entfernung(k, Top)$$

Da *Top* das allgemeinste Konzept einer Taxonomie ist und somit gar keine Information enthält, zeigt die Tiefe eines Konzepts an, wieviel Informationsgehalt überhaupt in einem Konzept enthalten ist. Dadurch kann die Tiefe dafür verwendet werden, den Informationsunterschied zwischen einem Konzept und seinem spezifischen Vater in eine relative Beziehung zu setzen.

Auf dieser Grundlage definiert Helwich (2003) zwei Ähnlichkeitsmaße für Taxonomien, nach den Vorschlägen für Ähnlichkeitsmaße aus der Literatur nach Poole und Campbell (1995) und Zhong et al. (2002). Dabei orientiert er sich an drei für die Ähnlichkeit von Taxonomien wünschenswerten Eigenschaften:

1. Die Ähnlichkeit von Konzepten ist umso größer, je größer die Tiefe des spezifischen Vaters ist.
2. Die Ähnlichkeit von Konzepten ist umso kleiner, je größer der Unterschied zwischen der Tiefe der Konzepte und der Tiefe des spezifischen Vaters ist.
3. (a) Die Ähnlichkeit von Vater und Sohn ist größer als die Ähnlichkeit von Brüdern.
   (b) Die Ähnlichkeit von Vater und Sohn ist kleiner als die Ähnlichkeit von Brüdern.

Da Punkt 3 von der Art der benutzten Taxonomie abhängt, werden zwei Ähnlichkeitsmaße definiert. Das Ähnlichkeitsmaß nach Poole und Campbell in Definition 3.1.4 erfüllt die Forderungen 1, 2 sowie 3a.

Definition 3.1.4
Die Ähnlichkeit zwischen zwei Konzepten k_1 und k_2 mit spezifischem Vater k_{spezV} ist:

$$\ddot{a}hnlich' \stackrel{def}{=} \frac{2\ tiefe\ (k_{spezV})}{tiefe\ (k_1) + tiefe\ (k_2)}$$

Um auch die Ähnlichkeit für Taxonomien, für die Punkt 3b gelten soll, berechnen zu können, wird für Definition 3.1.5 auf ein Ähnlichkeitsmaß, das in Zhong et al. (2002) benutzt wird, zurückgegriffen.

Definition 3.1.5
Die Ähnlichkeit zwischen zwei Konzepten k_1 und k_2 mit spezifischem Vater k_{spezV} ist:

$$\ddot{a}hnlich' \stackrel{def}{=} 1 - (2\ markstein(k_{spezV}) - (markstein(k_1) + markstein(k_2))$$

Wobei markstein definiert ist als:

$$markstein(k) \stackrel{def}{=} \frac{1}{2^{1+tiefe(k)}} = \frac{0,5}{2^{tiefe(k)}}$$

Definition 3.1.5 sorgt dafür, dass das Verhältnis des Unterschieds zwischen Konzepten zum Abstand exponentiell ist. Dieses Ähnlichkeitsmaß erfüllt die Forderungen 1, 2 sowie 3b (Für weitere Details zu diesem Ähnlichkeitsmaß siehe Helwich, 2003; Zhong et al., 2002).

Zusätzlich zur Subsumptionsrelation, muss für die Berechnung der Ähnlichkeit ebenfalls die Exklusionsrelation berücksichtigt werden. Sind zwei Konzepte exklusiv, sollen sie ungeachtet der Subsumptionsrelation die Ähnlichkeit 0 haben.[1]

Definition 3.1.6
Die Ähnlichkeit zwischen zwei Konzepten k_1 und k_2 einer Taxonomie $T = (S, \leq, \leftrightarrow)$ ist:

$$\ddot{a}hnlich(k_1, k_2) \stackrel{def}{=} \begin{cases} 0 & , wenn\ k_1 \leftrightarrow k_2 \\ \ddot{a}hnlich'(k_1, k_2) & , sonst \end{cases}$$

Damit kann zwischen beliebigen Konzepten einer Taxonomie die Ähnlichkeit bestimmt werden.

[1] Helwich (2003) spricht hier auch von einer geringen Ähnlichkeit.

3.2 Ähnlichkeit von Knoten

Die Ähnlichkeit eines Knoten-Matches (i, p) wird bestimmt über die Ähnlichkeiten der Prädikate, bestehend aus den Sorten SOR und Eigenschaften EIG, und den Namen NAM der beiden Knoten i und p. Prädikate sind Konzepte der Taxonomie und Namen sind Wörter über einem Alphabet Σ. Im Vergleich zu Helwich (2003) wird hier die Unterscheidung zwischen Sorten und Eigenschaften aufgegeben, da sich diese in der Praxis als nicht geeignet herausgestellt hat. Im Normalfall wird zunächst die Ähnlichkeitssumme der einzelnen Komponenten bestimmt, die schon ein Maß für Ähnlichkeit ist, und dann geeignet normiert, um den Wertebereich zwischen 0 und 1 zu halten.

Für die Berechnung der Ähnlichkeitssumme der Prädikate der beiden Knoten werden die beiden Mengen $\text{PRÄD}_i = \text{SOR}_i \cup \text{EIG}_i$ für i und $\text{PRÄD}_p = \text{SOR}_p \cup \text{EIG}_p$ für p miteinander verglichen. Exklusivität von Prädikaten stellt einen Sonderfall dar. Kommt in PRÄD_i ein Pädikat vor, welches exklusiv zu einem Prädikat aus PRÄD_p ist, so sind die beiden Knoten auf jeden Fall nicht ähnlich, die Ähnlichkeit ist 0. Denn Ähnlichkeit ist ein Maß dafür, wie wahrscheinlich es ist, dass die beiden Knoten dasselbe Objekt in der Welt repräsentieren. Da ein Objekt der Welt nicht gleichzeitig durch zwei zueinander exklusive Prädikate beschrieben werden kann, ist es unmöglich, dass i und p mit exklusiven Prädikaten dasselbe Objekt beschreiben. Die beiden Knoten sind *exklusiv*.

Definition 3.2.1
Wenn es für zwei Knoten i, p mit Prädikatsmengen PRÄD_i, PRÄD_p, ein $k_i \in \text{PRÄD}_i$ gibt, welches zu einem $k_p \in \text{PRÄD}_p$ exklusiv *ist, sind auch i und p* exklusiv *zueinander.*

Gibt es in den Mengen keine zueinander exklusiven Prädikate, wird die Ähnlichkeitssumme aus den Einzelähnlichkeiten von Paaren von Prädikaten bestimmt. Dafür wird zwischen den beiden Prädikatmengen eine Zuordnung $Z = \{(k_i, k_p) \mid (k_i, k_p) \in \text{PRÄD}_i \times \text{PRÄD}_p\}$ benötigt. Die Berechnung von Ähnlichkeit für Matches von Konzepten wurde im vorherigen Abschnitt 3.1 behandelt.

Definition 3.2.2
Die Ähnlichkeitssumme der Prädikate eines Knotenmatches aus den Knoten i und p ist abhängig von der Zuordnung der Prädikate Z definiert als:

$$\textit{ähnlichsumme}_Z^P(i, p, Z) \stackrel{def}{=} \sum_{(k_i,k_p)\in Z} \textit{ähnlich}(k_i, k_p)$$

Die Zuordnung muss, um für die Ähnlichkeitsberechnung geeignet zu sein, laut Helwich (2003) *konsistent* sein, d.h. die in den Prädikatmatches enthaltenen Prädikate sind paarweise verschieden.

Definition 3.2.3
Eine Zuordnung $Z = \{(k_i, k_p) \mid (k_i, k_p) \in \text{PRÄD}_i \times \text{PRÄD}_p\}$ *ist* konsistent, *wenn für alle* k_i, k_i', k_p, k_p' *mit* $k_i \neq k_i'$ *und* $k_p \neq k_p'$ *gilt:*

$$(k_i, k_p) \in Z \Rightarrow (k_i', k_p) \notin Z \wedge (k_i, k_p') \notin Z$$

Es wird also eine 1:1-Zuordnung der Prädikate gesucht. Für ein Konzept k_i aus PRÄD_i wird die Ähnlichkeit zu genau einem Konzept k_p aus PRÄD_p untersucht, aber wenn in einem der Knoten mehr Prädikate als in dem anderen vorliegen, können höchstens soviele Paare gebildet werden wie die kleinere Anzahl von beiden und die überzähligen Prädikate tragen nichts zur Ähnlichkeitssumme bei. Dies bedeutet auch, dass für die Ähnlichkeitsberechnung davon ausgegangen wird, dass Prädikate klar gegeneinander abgegrenzt sind und die Beschreibung eines Attributes des Objekts der Welt nicht in dem einen Knoten durch ein Prädikat und in dem anderen durch zwei ähnliche Prädikate vorliegt.

Da es grundsätzlich mehrere konsistente Zuordnungen gibt, muss zusätzlich noch Optimalität der Zuordnung gefordert werden, um die Ähnlichkeitssumme eindeutig zu bestimmen. Bei Helwich (2003) wird die Optimalität einer Zuordnung Z über die maximale Ähnlichkeitssumme definiert.

Definition 3.2.4
Gegeben zwei Knoten i, p. *Eine konsistente Zuordnung* Z *ist optimal, wenn für alle konsistenten Zuordnungen* Z' *gilt:*

$$\ddot{a}hnlichsumme_Z^P(i, p, Z) \geq \ddot{a}hnlichsumme_{Z'}^P(i, p, Z')$$

Die Definition für die Ähnlichkeitssumme der Prädikate zwischen zwei Knoten ergibt sich dann wie folgt:

Definition 3.2.5
Ist Z *eine konsistente, optimale Zuordnung für die Knoten* i *und* p, *ist die Ähnlichkeitssumme von* i *und* p

$$\ddot{a}hnlichsumme^P(i, p) \stackrel{def}{=} \ddot{a}hnlichsumme_Z^P(i, p, Z)$$

Definition 3.2.4 ist jedoch nicht die einzige Möglichkeit, Optimalität zu definieren. Grundsätzlich sind alle Optimalitätskriterien, die dafür sorgen, dass für alle optimalen Zuordnungen die Ähnlichkeitssumme gleich ist, für

eine eindeutige Bestimmung der Ähnlichkeit ausreichend. Da das Optimalitätskriterium für das Verfahren der Bestimmung der Zuordnung eine große Rolle spielt, wird es hier nicht festgelegt, sondern dies in Abschnitt 4.1 den jeweiligen Berechnungsverfahren überlassen.

Die Ähnlichkeit zwischen Namen wird durch einfache Gleichheit geprüft und ist deswegen trivial, da zwei Namen entweder gleich sind und somit eine Ähnlichkeit von 1 haben oder nicht gleich sind und die Ähnlichkeit somit 0 ist. Wie bei den Prädikatmatches gibt es einen Sonderfall, der Exklusivität nach sich zieht. Wenn beide Knoten Namen haben, jedoch kein Name in beiden Knoten gleich ist, sind die beiden Knoten *exklusiv*. Dies entspricht der Annahme, dass wenn ein Objekt in der Welt Namen hat, entweder alle oder kein Name bekannt[2] ist und somit Knoten mit verschiedenen Namen nicht dasselbe Objekt in der Welt repräsentieren können. Ist dies nicht der Fall, wird für jeden gleichen Namen 1 zur Ähnlichkeitssumme der Namen addiert.

Definition 3.2.6
Die Ähnlichkeitssumme der Namen ist gleich der Anzahl an gleichen Namen in NAM_i *und* NAM_p *der beiden Knoten* i *und* p*.*

$$\textit{ähnlichsumme}^N(i,p) \stackrel{def}{=} |\text{NAM}_i \cap \text{NAM}_p|$$

Die Ähnlichkeit für Knoten ist dann unter Vorbehalt der Exklusivität der Knoten definiert.

Definition 3.2.7
Die Ähnlichkeit zweier Knoten i*,* p *ist:*

$$\textit{ähnlich}(i,p) \stackrel{def}{=} \begin{cases} 0 & \text{, wenn } i,p \text{ exlusiv} \\ \textit{ähnlich}'(i,p) & \text{, sonst} \end{cases}$$

Bei Helwich (2003) wird in der analogen Definition auch noch auf Identität eingegangen, die sich aus den Sorten und Namen ergeben kann. In der Praxis hat sich dies aber als nicht sinnvoll herausgestellt und findet sich in der aktuellen Implementation nicht wieder.

Für $\textit{ähnlich}'$ sind mehrere Alternativen definiert, die sich durch die Normierung der Ähnlichkeitssumme und Namensähnlichkeitssumme unterscheiden und dadurch unterschiedliche Gewichtung auf Instruktionsknoten und Perzeptionsknoten legen sowie Unterschiede in der Anzahl der Prädikate und Namen in den Knoten anders bewerten.

[2] d. h. instruiert oder perzipiert

Definition 3.2.8
Die Ähnlichkeit zweier nicht exklusiver Knoten i, bestimmt durch PRÄD_i, NAM_i, *und p, bestimmt durch* PRÄD_p, NAM_p, *ist:*

$$\ddot{a}hnlich'(i,p) \stackrel{def}{=} \frac{\ddot{a}hnlichsumme^P(i,p) + \ddot{a}hnlichsumme^N(i,p)}{|\text{PRÄD}_i| + |\text{NAM}_i|}$$

Definition 3.2.9
Die Ähnlichkeit zweier nicht exklusiver Knoten i, bestimmt durch PRÄD_i, NAM_i, *und p, bestimmt durch* PRÄD_p, NAM_p, *ist:*

$$\ddot{a}hnlich'(i,p) \stackrel{def}{=} \frac{\ddot{a}hnlichsumme^P(i,p) + \ddot{a}hnlichsumme^N(i,p)}{min\left(|\text{PRÄD}_i| + |\text{NAM}_i|, |\text{PRÄD}_p| + |\text{NAM}_p|\right)}$$

Definition 3.2.10
Die Ähnlichkeit zweier nicht exklusiver Knoten i, bestimmt durch PRÄD_i, NAM_i, *und p, bestimmt durch* PRÄD_p, NAM_p, *ist:*

$$\ddot{a}hnlich'(i,p) \stackrel{def}{=} \frac{2\left(\ddot{a}hnlichsumme^P(i,p) + \ddot{a}hnlichsumme^N(i,p)\right)}{|\text{PRÄD}_i| + |\text{NAM}_i| + |\text{PRÄD}_p| + |\text{NAM}_p|}$$

In Definition 3.2.8 wird eine asymmetrische Ähnlichkeit definiert. Die Normierung für den Wertebereich findet nur über die Anzahl der Prädikate und Namen des ersten Knoten i statt, der im Anwendungsfall dann der Instruktionsknoten ist. So wird der Information, die im Instruktionsknoten enthalten ist, stärkeres Gewicht gegeben. Es können in der Zuordnung Z zur Berechnung der Ähnlichkeitssumme auf Grund der Konsistenzbedingung höchstens soviele Prädikatspaare sein, wie Prädikate in der kleineren der Mengen PRÄD_i und PRÄD_p vorliegen. Normiert man nur über die Anzahl der Prädikate des Instruktionsknoten, so wird für fehlende Information auf der Perzeptionsseite die Ähnlichkeit geringer, während fehlende Information auf der Instruktionsseite für die Ähnlichkeit ignoriert wird. Ähnliches gilt für die Namen.

Die Definitionen 3.2.9 und 3.2.10 sind hingegen symmetrisch. Definition 3.2.9 benutzt die minimale Anzahl an informationstragenden Elementen (Prädikaten und Namen) zur Normierung und überschüssige Information wird ignoriert. Definition 3.2.10 bildet einen Mittelwert, bei dem Prädikate und Namen, die keine Entsprechung im anderen Knoten haben, sich stärker negativ auf die Ähnlichkeit auswirken.

Welche dieser drei Definitionen am besten für die Ähnlichkeitsberechnung geeignet ist, soll hier wie auch schon bei Helwich nicht abschließend geklärt werden. Allerdings werden im Abschnitt 3.5 noch einmal einige Entscheidungen des Gesamtansatzes diskutiert.

3.3 Ähnlichkeit von Relationen

Die Ähnlichkeitsbestimmung von Relationen-Matches ist trivial, weswegen bei Helwich (2003) nur implizit darauf eingegangen wird. Relationen sind jeweils durch ein Konzept einer Taxonomie bestimmt und für dieses Konzept lässt sich die Ähnlichkeit zu dem Konzept der gematchten Relation nach Definition 3.1.6 berechnen. Zu bedenken ist hier nur die Stelligkeit der Relationen. Exklusivität verschiedenstelliger Relationen kann aber auch in der Taxonomie modelliert werden.

Definition 3.3.1
Die Ähnlichkeit eines Relationenmatches der Relationen $r_i = (k_i, i_1, \ldots, i_n)$ *und* $r_p = (k_p, p_1, \ldots, p_m)$ *mit* k_i *und* k_p *als Konzepte der Taxonomie* T_R *und* $i_1, \ldots, i_n$ *und* $p_1, \ldots, p_m$ *als CRIL-Knoten ist:*

$$\ddot{a}hnlich(r_i, r_p) \stackrel{def}{=} \ddot{a}hnlich(k_i, k_p)$$

Die Ähnlichkeit der Knoten in einer Relation wird für die Ähnlichkeitsbestimmung von Relationen nicht berücksichtigt. Die Abhängigkeit der Relationen von den Knoten wird in Abschnitt 3.4 bei der Bestimmung der Ähnlichkeit von Graphen einbezogen.

3.4 Ähnlichkeit von Graphen

Da nun durch die Definitionen 3.2.7 und 3.3.1 die Ähnlichkeit von CRIL-Knoten und CRIL-Relationen bestimmt ist, kann eine zusammengesetzte Ähnlichkeit für CRIL-Graphen bestimmt werden. Dies erfolgt weitestgehend analog zu der Bestimmung der Ähnlichkeit der Knoten durch Aufsummierung der Einzelähnlichkeiten von einander zugeordneten Paaren aus Knoten bzw. Relationen und geeignete Normierung auf den Wertebereich zwischen 0 und 1. Zunächst muss eine Zuordnung der beiden zu vergleichenden Graphen gefunden werden.

Definition 3.4.1
Für zwei CRIL-Graphen $G_1 = (K_1, R_1, T_A, T_R)$ *und* $G_2 = (K_2, R_2, T_A, T_R)$ *ist eine Zuordnung* $Z = (KM, RM)$ *bestimmt durch die beiden Mengen* $KM \subseteq K_1 \times K_2$ *und* $RM \subseteq R_1 \times R_2$ *und es muss gelten:*

$$\forall(r_1, r_2) \in RM : [\forall kn_1 \in knoten(r_1) : (\exists kn_2 \in knoten(r_2) : (kn_1, kn_2) \in KM)]$$

Die Bedingung stellt sicher, dass in jeder Zuordnung für jede Relation auch die entsprechenden Knoten mit einem Paar in der Zuordnung vertreten

sind. So bilden die in der Zuordnung gematchten Knoten und Relationen eines der Graphen G_1 und G_2 immer einen Teilgraph des jeweiligen Graphen.

Für eine Zuordnung, die Definition 3.4.1 genügt, lässt sich nun die Ähnlichkeitssumme definieren.

Definition 3.4.2
Eine Ähnlichkeitssumme einer Zuordnung $Z = (KM, RM)$ für zwei CRIL-Graphen G_1 und G_2 ist :

$$\begin{gathered} \mathit{ähnlichsumme}(G_1, G_2, Z) \stackrel{def}{=} \\ \sum_{(kn_1,kn_2)\in KM} \mathit{ähnlich}(kn_1, kn_2) + \sum_{(r_1,r_2)\in RM} \mathit{ähnlich}(r_1, r_2) \end{gathered}$$

Mit Hilfe dieser Ähnlichkeitssumme wiederum definiert Helwich (2003) analog zu den Definition 3.2.8 und 3.2.10 Ähnlichkeit für Graphen. Es sei darauf hingewiesen, dass Ähnlichkeit zwischen zwei Graphen nur in Bezug auf eine Zuordnung definiert wird, da im Gegensatz zu den Knoten nicht die Ähnlichkeit selbst gesucht ist, sondern insbesondere diese Zuordnung der einzelnen Knoten und Relationen Teil der gesuchten Lösung ist, um das Match eines bestimmten Knoten zu finden. Eine Verallgemeinerung auf die maximale (bzw. optimale) Ähnlichkeit zwischen zwei Graphen unabhängig von der Zuordnung wie bei den Knoten in Definition 3.2.5 ist deswegen nicht sinnvoll.

Definition 3.4.3
Die Ähnlichkeit zwischen zwei Graphen $G_1 = (K_1, R_1, T_A, T_R)$ und $G_2 = (K_2, R_2, T_A, T_R)$ basierend auf einer Zuordnung Z ist:

$$\mathit{ähnlich}(G_1, G_2, Z) \stackrel{def}{=} \frac{\mathit{ähnlichsumme}(G_1, G_2, Z)}{|K_1| + |R_1|}$$

Dieses asymmetrische Ähnlichkeitsmaß normiert die Ähnlichkeitssumme durch die Anzahl der Knoten und Relationen des ersten Graphen. Es ist dafür vorgesehen den Instruktionsgraphen mit dem Perzeptionsgraphen zu vergleichen und wird normalerweise für die Ähnlichkeitsberechnungen im Geometrischen Agenten benutzt. Zusätzlich wird in Helwich (2003) noch ein zweites, symmetrisches Ähnlichkeitsmaß definiert.

Definition 3.4.4
Die Ähnlichkeit zwischen zwei Graphen $G_1 = (K_1, R_1, T_A, T_R)$ und $G_2 = (K_2, R_2, T_A, T_R)$ basierend auf einer Zuordnung Z ist:

$$\mathit{ähnlich}(G_1, G_2, Z) \stackrel{def}{=} \frac{2\ \mathit{ähnlichsumme}(G_1, G_2, Z)}{|K_1| + |R_1| + |K_2| + |R_2|}$$

Ebenfalls analog zur Knotenähnlichkeit wird Konsistenz der Zuordnung definiert. Allerdings ist die Konsistenz kein notwendiges Kriterium für die Ähnlichkeitsberechnung. Jedoch ist nur für konsistente Zuordnungen garantiert, dass die Ähnlichkeit zwischen 0 und 1 liegt und es werden nur konsistente Lösungen in den Berechnungsverfahren in Kapitel 4 als Lösungen gesucht.

Definition 3.4.5
Eine Zuordnung $Z = (KM, RM)$ *zwischen den beiden Graphen* $G_1 = (K_1, R_1, T_A, T_R)$ *und* $G_2 = (K_2, R_2, T_A, T_R)$ *ist* konsistent, *wenn für alle* k_1, k_1', k_2, k_2' *und* r_1, r_1', r_2, r_2' *mit* $k_1 \neq k_1'$, $k_2 \neq k_2'$, $r_1 \neq r_1'$ *und* $r_2 \neq r_2'$ *gilt:*

$$(k_1, k_2) \in KM \Rightarrow (k_1', k_2) \notin KM \wedge (k_1, k_2') \notin KM$$
$$\wedge$$
$$(r_1, r_2) \in RM \Rightarrow (r_1', r_2) \notin RM \wedge (r_1, r_2') \notin RM$$

3.5 Betrachtungen zur Eignung der Ähnlichkeitsmaße

Eine vollständige Analyse der im Geometrischen Agenten verwendeten Ähnlichkeitsmaße soll nicht Teil dieser Diplomarbeit sein. Trotzdem werde ich kurz auf die verwendeten Ansätze eingehen und einige Aspekte kritisch beleuchten. Es können und sollen hier keine abschließenden Antworten gegeben werden, weil dies den Rahmen dieser Diplomarbeit übersteigen würde.

Wie oben schon angedeutet beruht die Rechtfertigung für die bei Helwich (2003) vorgestellte Berechnung der Ähnlichkeit zwischen Konzepten auf der Annahme, dass die Abstände zwischen Konzepten im Subsumptionsgraphen der Taxonomie mit dem Informationsgehalt korrespondieren. Dies bedeutet, dass auf der gleichen Tiefe der Taxonomie alle Konzepte die gleiche Menge an Informationsgehalt haben müssen. Da zwischen einem Konzept und seinem Vater allerdings beliebig viele Zwischenkonzepte eingefügt werden könnten, wodurch die Tiefe vergrößert würde, gilt diese Annahme nicht für alle Taxonomien. Ob die hier vorgestellten Ähnlichkeitsmaße adäquate Aussagen über das Verhältnis zwischen Konzepten treffen können, hängt also auch stark von der Modellierung der Taxonomie ab.

Bei der Berechnung der Knotenmatches ist zum einen die Behandlung der Namen problematisch. Bei Helwich (2003) werden Namen dadurch, dass ein in beiden Knoten vorkommender Name ungeachtet der Prädikate zu einer Ähnlichkeit von 1 führt, weit stärker berücksichtigt, als es in der jetzigen Implementation des Geometrischen Agenten der Fall ist. In der jetzigen Version haben nur verschiedene Namen eine Sonderrolle, während gleiche Namen nur

denselben Einfluss wie ein Prädikat mit Ähnlichkeit 1 haben. So ist der Einfluss von Namen auf Knoten mit vielen Prädikaten geringer als auf solche mit wenigen Prädikaten. Wird Definition 3.2.9 verwendet, haben Namen für einige Fälle überhaupt keine Auswirkung.

Angenommen, zwei Knoten p, p' unterscheiden sich nur dadurch, dass p' keinen Namen hat, während p einen hat, die Prädikate sind also gleich. Werden diese beiden Knoten nun mit einem Knoten i, der dieselben Prädikate wie p und p' und außerdem den Namen von p hat, verglichen, so ist die Ähnlichkeit zwischen i und p sowie i und p' beidesmal gleich 1. Denn die Einzelähnlichkeit der Prädikate ist immer 1 und damit ist die Ähnlichkeitssumme von (i, p') gleich der Anzahl der Prädikate, mit der dann auf 1 normiert wird. Für i und p wird dazu noch die Ähnlichkeit 1 für den Namen hinzugezählt und dann ebenfalls auf 1 normiert, da ein Name zu der Anzahl der Prädikate hinzugezählt wird. Trotz des Namens ist es dem Geometrischen Agenten so nicht möglich, zwischen den beiden Knoten zu unterscheiden.

Ein weiterer Punkt bei der Bestimmung der Knotenähnlichkeit, der diskutiert werden kann, ist die Forderung einer 1:1-Zuordnung der Prädikate. Es wird versucht, jedem Prädikat genau ein anderes Prädikat des anderen Knoten zuzuordnen. Wenn es zwei Prädikate mit hoher Ähnlichkeit gegenüber einem Prädikat der Gegenseite gibt, fließt diese hohe Ähnlichkeit nur einmal in die Ähnlichkeitssumme ein und für das nicht gematchte Prädikat wird entweder gar kein Match oder eines mit geringerer Ähnlichkeit gefunden. Dies hat besonders Auswirkungen, wenn es sich bei dem Knoten mit den mehreren Prädikaten um einen Instruktionsknoten handelt. Als Beispiel sei hier ein Instruktionsknoten i mit den beiden Prädikaten *haus* und *gebäude* genannt, der mit einem Perzeptionsknoten p mit dem einen Prädikat *gebäude* über das asymmetrische Ähnlichkeitsmaß aus Definition 3.2.8 verglichen wird. Die Ähnlichkeit von gleichen Konzepten ist 1 und, da ein Haus ein Gebäude ist, wird eine hohe Ähnlichkeit für das Paar $(haus, gebäude)$, $ähnlich(haus, gebäude) = 0,9$, angenommen. Namen werden nicht betrachtet. Die Ähnlichkeit errechnet sich dann mit einer 1:1-Zuordnung als:

$$ähnlich(i, p) = \frac{ähnlich(gebäude, gebäude)}{|\{gebäude, haus\}|} = \frac{1}{2} = 0,5$$

Die Ähnlichkeit ist mit $0,5$ relativ niedrig, obwohl alle möglichen Prädikatpaare eine hohe Ähnlichkeit aufweisen. Ebenso ist zu bedenken, dass für alle anderen Instruktionsknoten, die *gebäude* und ein weiteres Prädikat haben, die gleiche Ähnlichkeit gilt. Eine n:1-Zuordnung jedoch, die jedem Prädikat des Instruktionsknoten genau ein Prädikat der Perzeption zuordnet, würde

eine weit höhere Ähnlichkeit erreichen.

$$\begin{aligned} \ddot{a}hnlich(i,p) &= \frac{\ddot{a}hnlich(gebäude,gebäude)+\ddot{a}hnlich(haus,gebäude)}{|\{gebäude,haus\}|} \\ &= \frac{1+0,9}{2} = 0,95 \end{aligned}$$

Es ist zu überlegen, ob es wirklich gerechtfertig ist, dass der Agent ein Prädikat für den Vergleich nur ein einziges Mal verwenden darf, um die Ähnlichkeit zu bestimmen. Zwar kommen im Anwendungsfall zum einen durch die Modellierung der Taxonomie und zum anderen durch die Art der Verarbeitung der Instruktion und Perzeption zu CRIL-Repräsentationen Fälle wie das oben genannte Beispiel nicht vor, allerdings ist leicht eine Situation für reale Anwendungen vorstellbar, in der die Perzeption, etwa durch Sensorfehler oder Ähnliches, eingeschränkt ist und in vergleichbarer Weise nur rudimentäre Knoten liefert.

Die Ähnlichkeitsberechnung des Geometrischen Agenten und insbesondere die für CRIL-Graphen wird von Helwich (2003) in eine Klassifikation eingeordnet, welche von Poole und Campbell (1995) aufgestellt wurde. Er spricht dabei von oberflächlicher, struktureller und thematischer Ähnlichkeit, die wie folgt charakterisiert ist:

- Oberflächliche Ähnlichkeit:
  Basiert auf der Ähnlichkeit von Objekten und Attributen. Relationen oder Muster werden nicht berücksichtigt.
- Strukturelle Ähnlichkeit:
  Basiert auf Ähnlichkeiten bezüglich strukturellen Merkmalen wie z. B. Anzahl von Kanten, Knotengrad oder Pfadlängen.
- Thematische Ähnlichkeit:
  Bedeutung basiert auf dem Vorhandensein von Relationen zwischen Knoten und bestimmten Mustern von Konzepten und Relationen.

Eine reine oberflächliche Ähnlichkeit, die sich nur auf die Konzepte bzw. in diesem Fall Knoten stützt, oder reine strukturelle Ähnlichkeit, die nur von der Anzahl von Komponenten und Beziehung unter diesen, wie dem Knotengrad abhängt, liegt offensichtlich nicht vor. Helwich ordnet das Ähnlichkeitsmaß für Graphen als oberflächlich-thematisch ein und begründet dies damit, dass eine rein thematische Ähnlichkeit nicht gegeben ist. Obwohl sowohl Knoten mit ihren Konzepten als auch Relationen zwischen Knoten in die Ähnlichkeit eingehen, gäbe es keine dem Geometrischen Agenten explizit bekannten, domänenspezifischen Muster, über die in beiden Graphen die Ähnlichkeit bestimmt wird, wie sie von Poole und Campbell (1995) erwähnt werden.

Meiner Meinung nach vernachlässigt diese Sichtweise jedoch zu stark den strukturellen Aspekt des Ähnlichkeitsmaßes, der durch die Bedingung in De-

finition 3.4.1 gegeben ist, die nur Relationen-Matches erlaubt, deren Knoten ebenfalls gematcht sind. Auch die Normierung, orientiert an der Anzahl von Knoten und Relationen der Graphen und nicht an der Anzahl der zugeordneten Paare, hat für mich strukturelle Aspekte. Also würde ich von einem Ähnlichkeitsmaß auf einem thematischen Level sprechen, welches sowohl strukturelle als auch oberflächliche Aspekte vereint.

Abschließend ist zu sagen, dass diese Kritikpunkte weitestgehend unabhängig von den Problemen sind, die das Matching aufweist und die Gegenstand von Kapitel 4 und Fokus dieser Diplomarbeit sind. Die Verfahren des Matchings suchen nur Zuordnungen nach bestimmten Kriterien und nutzen dafür die Ähnlichkeiten der Komponenten unabhängig davon, wie diese berechnet wurden. Die dabei auftretenden Probleme sind eher komplexitätstechnischer Natur und hängen nicht damit zusammen, ob die Ähnlichkeitsmaße geeignet für die Modellierung der Domäne sind.

Die oben angesprochene Verwendung von n:1-Zuordnungen bei der Knotenberechnung, würde zwar einen anderen Algorithmus als die in Abschnitt 4.1 diskutierten Lösungen für eine 1:1-Zuordnung benötigen, die Lösung des n:1-Problems ist jedoch sehr einfach durch Zuordnung des Maximums über allen Paaren (i, p) zu jedem Knoten i bestimmbar. Deswegen wird sich hier auf die adäquate Lösung des 1:1-Zuordnungsproblem konzentriert und weiterführenden Arbeiten die von den Komplexitätsproblemen unabhängige Entscheidung zwischen den beiden Möglichkeiten überlassen.

Die von Helwich (2003) abweichende Einordnung des Ähnlichkeitsmaßes der CRIL-Graphen hat zunächst nichts mit der Eignung des Ähnlichkeitsmaßes für die gestellte Aufgabe zu tun, sondern ist nur eine andere Sichtweise auf das Ähnlichkeitsmaß. Allerdings führt diese geänderte Sichtweise zur Überlegung, wie die strukturellen Aspekte der Graphenähnlichkeit in den Matchingverfahren besser ausgenutzt werden könnten, und damit zu dem in Abschnitt 4.2 vorgestellten neuen Ansatz die Zuordnung für die CRIL-Graphen zu bestimmen.

4. MATCHING FÜR DEN GEOMETRISCHEN AGENTEN

Im Matching soll einem bestimmten instruierten CRIL-Knoten ein Knoten der Perzeption zugeordnet werden. Dieser Knoten wird im Folgenden *Ziel des Matchings* genannt. Das Match dieses Knotens wird z.B. für die Ausführung der CRIL-Aktionen benötigt, es sind aber noch andere Anwendungsfälle denkbar.

Um das Ziel des Matchings in der Umgebung zu erkennen, vergleicht der Geometrische Agent die perzipierte Umgebung mit der gespeicherten Instruktion auf übereinstimmende Strukturen. Dazu werden zunächst geeignete Teilgraphen der beiden als CRIL-Graph vorliegenden Wissensbasen ausgewählt, über Schlüsse mit weiterem Wissen angereichert und dann auf Ähnlichkeit überprüft. Der Vorgang kann in drei Komponenten zerlegt werden, die im Folgenden einzeln betrachtet werden.

Mit der Auswahl und Anreicherung des bestehenden Wissens findet zunächst eine Vorverarbeitung statt. Im eigentlichen Matching wird dann die Ähnlichkeit von Teilgraphmatches bestimmt. Die Ähnlichkeit setzt sich wie in Kapitel 3 dargestellt aus Einzelähnlichkeiten der CRIL-Relationen- und CRIL-Knoten-Matches zusammen. Die kleinste Einheit, deren Ähnlichkeit bestimmt werden kann, ist ein Konzept-Match, bestehend aus zwei Konzepten der Taxonomie. Die Berechnungen dafür werden hier nicht weiter behandelt, da sie nur auf einfacher Suche im Subsumptionsgraphen beruhen. Über die Ähnlichkeit der Konzept-Matches werden die optimalen Zuordnungen der Konzepte der CRIL-Knoten bestimmt und so die Ähnlichkeit von Knotenmatches berechnet. Aus diesen Ähnlichkeiten können dann unter Berücksichtigung der CRIL-Relationen Zuordnungen der Teilgraphen bestimmt und Teilgraphmatches gebildet werden.

Der Matchingvorgang liefert eine Reihe von Graph-Matches. Diese entsprechen jeweils einer Interpretation des gesuchten Routenabschnitts in der perzipierten Umgebung. Das Ziel des Matchings kann aus diesen Möglichkeiten über geeignete Kriterien, wie der Ähnlichkeit und Routenverfolgungsheuristiken, identifiziert werden. Es ist zwischen vollständigen Matches und partiellen Matches zu unterscheiden. Bei vollständigen Matches konnten alle Komponenten des betrachteten Instruktionsteilgraphen einer Komponente der perzipierten Umgebung zugeordnet werden. Bei partiellen Matches sind

nur Teile des instruierten Graphen gematcht, weil keine geeigneten Entsprechungen im Perzeptionsgraphen vorhanden waren.

Im Folgenden untersuche ich die drei Komponenten des Matchingverfahrens in der Realisierung vor meiner Diplomarbeit, arbeite Probleme heraus und stelle alternative Lösungen vor. Dabei beginne ich mit der Grundlage, dem Knotenmatching, das als Basis für die anderen Teile des Matchings dient und in der jetzigen Form große Probleme aufweist. Danach widme ich mich dem Teilgraphmatching, welches das Kernstück des Matchingverfahrens darstellt und auf dem Knotenmatching aufbaut. Die Vorverarbeitung wird als letztes thematisiert, da die notwendigen Vorverarbeitungsschritte auch stark von der Expressivität und Effizienz des verwendeten Matchingverfahrens abhängen.

4.1 Knoten- und Taxonomiematching

Das Knotenmatching wird benötigt, um die Ähnlichkeit zwischen zwei Knoten zu bestimmen und so beurteilen zu können, ob die Knoten dasselbe Objekt in der Welt beschreiben können. Dazu müssen die Eigenschaften des durch den einen Knoten beschriebenen Objekts mit den Eigenschaften des durch den anderen Knoten beschriebenen Objekts verglichen werden. Die Ähnlichkeit zwischen zwei Knoten ist dabei definiert über die Einzelähnlichkeiten einer optimalen 1:1-Zuordnung der Eigenschaftsprädikate, welches wiederum Konzepte der Taxonomie sind. Es ergibt sich das Problem, aus der Menge N der Prädikate des Instruktionsknotens n Elemente den m Elementen der Menge M der Prädikate des Perzeptionsknotens so zuzuordnen, dass die resultierende Zuordnung optimal ist, was in der Literatur als 'NxM-Matchingproblem' oder auch 'linear assignment problem' bezeichnet wird (vgl. Lawler, 1963; Burkard, 1999; Karp, 1980; Gale und Shapley, 1962; Munkres, 1957). Dabei sind grundsätzlich unterschiedliche Optimalitätskriterien möglich.

Ich werde im Folgenden darstellen, wie dieses Problem bisher im Matching gelöst wurde, herausarbeiten, warum es anders gelöst werden muss, und mehrere alternative Verfahren untersuchen.

4.1.1 Problem des bisherigen Verfahrens

Der Algorithmus, der im alten Matching verwendet wurde, berechnet eine optimale Zuordnung durch Herausstreichen von unähnlichen Kombinationen aus der Gesamtmenge aller möglichen Kombinationen. Das Optimalitätskriterium ist dabei die Maximierung der Ähnlichkeitssumme und somit im Endeffekt der Durchschnittsähnlichkeit, bis eine 1:1-Zuordnung entsteht. Der Al-

gorithmus kann im Grunde genommen als eine kostenbasierte Graphensuche über der Potenzmenge aller möglichen Kombinationen von Eigenschaftsprädikaten angesehen werden und lässt sich wie folgt in Pseudocode beschreiben:

procedure findePrädikatMatches(IPRÄDIKATE,PPRÄDIKATE)
  PRÄDIKATMATCHES = $\{(i,p) \mid i \in$ IPRÄDIKATE$, p \in$ PPRÄDIKATE$\}$
  while unverträgliche Matches $m1, m2 \in$ PRÄDIKATMATCHES **do**
    PRÄDIKATMATCHES$'$ = PRÄDIKATMATCHES $\setminus \{m1\}$
    PRÄDIKATMATCHES$''$ = PRÄDIKATMATCHES $\setminus \{m2\}$
    RAND = RAND $\bigcup \{$PRÄDIKATMATCHES$', $PRÄDIKATMATCHES$''\}$
    PRÄDIKATMATCHES = max(RAND)
    RAND = RAND $\setminus$ PRÄDIKATMATCHES
  end while
  return PRÄDIKATMATCHES
end procedure

Ausgehend von der Gesamtmenge aller möglichen Prädikat-Matches wird die aktuell zu untersuchende Menge von Prädikat-Matches PRÄDIKATMATCHES solange auf miteinander unverträgliche Matches untersucht, bis eine konsistente 1:1-Zuordnung gefunden wurde.

Definition 4.1.1
Unverträglich sind zwei Matches $(i1, p1)$, $(i2, p2)$ *dann, wenn entweder* $i1 = i2$ *oder* $p1 = p2$ *gilt, aber nicht beides.*

Sind zwei Matches unverträglich, werden aus der zur Zeit untersuchten Menge PRÄDIKATMATCHES zwei neue Mengen gebildet, in denen jeweils nur noch eines der beiden Matches enthalten ist, und zu der Menge RAND der zu untersuchenden Matches hinzugefügt. Dann wird aus RAND die Prädikat-Matchmenge mit der höchsten Ähnlichkeitssumme herausgenommen, um sie weiter zu untersuchen.

Werden keine unverträglichen Matches gefunden, wurde eine 1:1-Zuordnung erreicht. Diese Zuordnung hat die höchste Gesamtähnlichkeit aller möglichen konsistenten Zuordnungen und ist damit optimal. Ich verzichte hier auf einen formalen Beweis, da es sehr leicht nachvollziehbar ist, dass die Ähnlichkeitssumme von PRÄDIKATMATCHES in jeder Iteration kleiner wird oder mindestens gleich bleibt und somit in späteren Iterationen keine Zuordnung mit einer größeren Gesamtähnlichkeit erreicht werden kann.

Der Algorithmus kann als Graphensuche mit äquidistanter Welle in einem binären Graphen mit der Ausgangsmenge als Wurzel verstanden werden. RAND ist dabei die Grenze der Welle, aus der die nächste Kombination über eine Kostenfunktion ausgewählt wird. Kosten sind in diesem Fall die Ähnlichkeitsdifferenzen zur Ausgangsmenge und diese werden minimal gehalten,

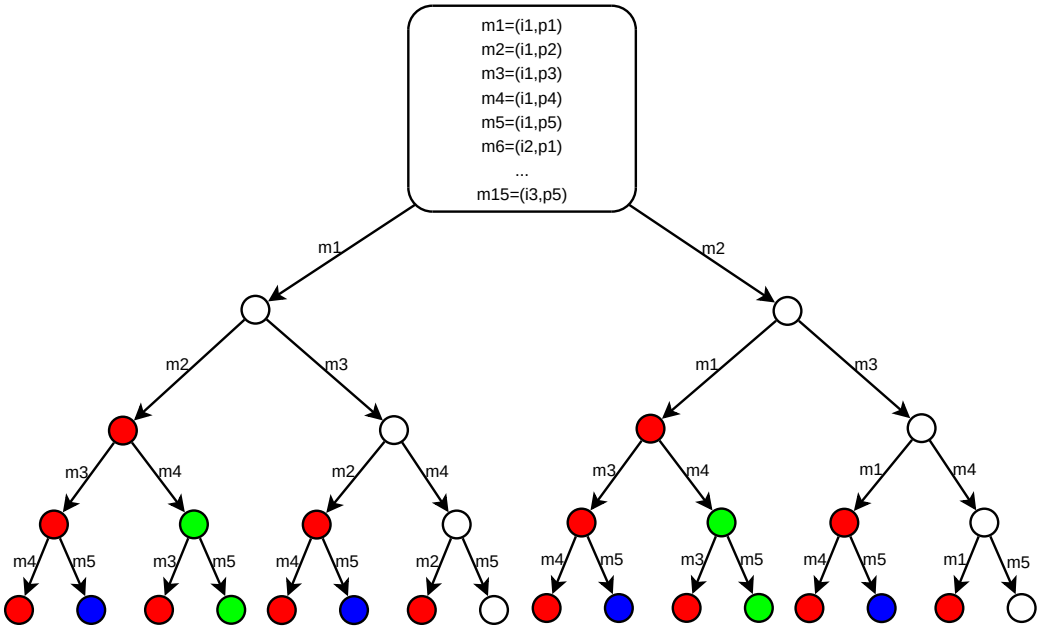


Abbildung 4.1.1: Veranschaulichung des Binärbaums für ein Beispiel mit $n = 3$ und $m = 5$ mit allen resultierenden Mengen mit bis zu vier Rausstreichungen. Für jede Stufe des Baumes sind Knoten, die dieselbe Lösung repräsentieren, farblich markiert. Weiße Knoten sind Einzellösungen. An den Kanten ist markiert, welches Match gestrichen wurde.

um die größtmögliche Ähnlichkeit einer konsistenten Menge zu finden. Die Verwendung dieses Algorithmus ist in drei Punkten fragwürdig.

Zum einen wird in einer Graphensuche der kürzeste Pfad zu einem Ziel gesucht. Diese Pfadinformation ist jedoch für die Lösung des NxM-Matchingproblems nicht notwendig.

Zum anderen fließt in die Kostenfunktion für die Auswahl des nächsten Schritts keine Information über das Ziel ein. Die Suche ist also nur über die Graphstruktur selbst auf das Ziel hin gerichtet.

Hinzu kommt, dass die Graphstruktur ein Binärbaum ist, in der jedoch auf einer Stufe mehrere Knoten dieselbe Menge an Matches repräsentieren können (vergleiche auch Abbildung 4.1.1). Eine unterschiedliche Reihenfolge des Herausstreichens führt zu gleichen Ergebnissen, die jedoch vom Algorithmus nicht als gleich erkannt werden und somit zu Mehrfachbehandlungen führen.

Zusammen ergibt sich daraus das Problem, dass nach einem beliebigen Zwischenschritt des Algorithmus alle Pfade zur bisher gefundenen Teillösung abgearbeitet werden müssen, bevor der Pfad weiter auf die Lösung hin ver-

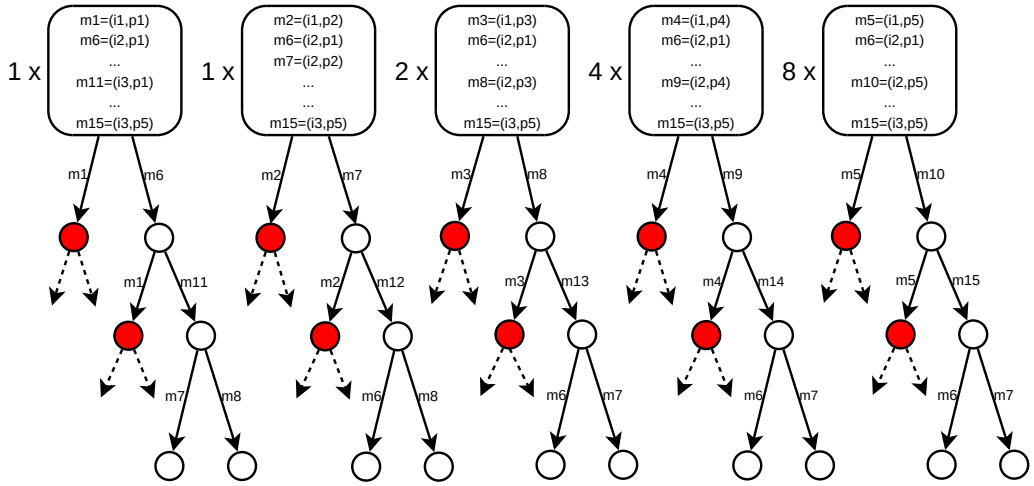


Abbildung 4.1.2: Weiterführung des Beispiels aus Abbildung 4.1.1. Die markierten Knoten zeigen Mengen an, die nicht mehr zu einer vollständigen Lösung führen können.

folgt werden kann, da alle angefangenen Pfade, die zur gleichen Teillösung führen, noch auf einem höheren Ähnlichkeitsniveau sein müssen.

Die Auswahl, welche Matches unverträglich sind und rausgestrichen werden, findet statisch über die Reihenfolge statt, die sich aus der Konstruktion der möglichen Matches ergibt. Dabei entstehen lange Folgen von Matches, die untereinander unverträglich bezüglich des N-Elementes sind, insgesamt m Folgen mit jeweils n unverträglichen Matches.

Wie man in Abbildung 4.1.1 sehen kann, kommt es dazu, dass sich Gruppen von gleichen Lösungen bilden. Bei einer Tiefe von $k < n$ Rausstreichungen können im Baum also nur $k + 1$ unterschiedliche Teillösungen entstehen, die sich auf Gruppen verschiedener Größe verteilen, von denen die größte die Hälfte aller Knoten dieser Stufe ausmacht. Von den insgesamt $k!$ möglichen Reihenfolgen, eine Teillösung zu erreichen, bildet die Struktur des Binärbaums also im schlimmsten Fall 2^{k-1} Möglichkeiten in Pfaden ab. Ist eine solche maximale Tiefe $k = n - 1$ erreicht, in der nur noch eines der vorher n unverträglichen Matches in den Teillösungen vorhanden ist, werden zunächst die zu diesem Match über das Element der Menge M unverträglichen Matches aus den verbleibenden Folgen entfernt (Vergleiche Abbildung 4.1.2). Wenn $n < m$ gilt, entstehen so auch Mengen, aus denen keine vollständige 1:1-Zuordnung mehr berechnet werden kann, da schon alle Matches für ein bestimmtes $i \in N$ herausgestrichen wurden. Daraufhin befindet sich hinter dem nun festgelegten Match eine weitere Folge N-unverträglicher Matches

mit Länge $n-1$, die wieder einen Baum — ähnlich desjenigen in Abbildung 4.1.1 dargestellten — bilden.

Dies geschieht genau $min(n,m)$ Mal, wobei sich die Anzahl der Matches und analog auch die Höhe der Teilbäume jeweils um eins verringert. Die Anzahl der Pfade für den schlimmsten Fall, einer Zuordnung, die nur Matches enthält, die zu den größten Gruppen gehören, ergibt sich also als Produkt der höchsten Anzahl gleicher Teillösungen der Teilbäume. Angefangen mit einem Teilbaum der Höhe $n-1$ mit 2^{n-1} Blättern und deswegen 2^{n-2} gleichen Lösungen ist dies für den Fall $n \leq m$:

$$O\left(2^{n-2} \cdot 2^{n-3} \ldots 2^{n-m-1}\right) = O\left(2^{m(n-m)}\right)$$

Falls $m < n$ ergibt sich nur:

$$O\left(2^{m-1} \cdot 2^{m-2} \ldots 2^{1}\right) \approx O\left(2^{mm}\right)$$

Zusätzlich hat jeder Pfad noch eine Anzahl Schritte, die der Gesamtzahl herausgestrichener Matches entsprechen. Die Gesamtzahl der möglichen Matches ergibt sich aus dem Produkt der Mächtigkeit der beiden Mengen $|\text{IPRÄDIKATE}| \cdot |\text{PPRÄDIKATE}|$ oder $n \cdot m$ und eine vollständige 1:1-Zuordnung hat dann genau $min\,(n,m)$ Matches. Es müssen also $nm - m$ bzw. $nm - n$ Matches herausgestrichen werden. Es folgt eine Worst-Case-Komplexität von $O\left(2^{m(n-m)}\,(nm-m)\right)$ allein für die Pfade, die zur richtigen Lösung führen. Dies wird weiter dadurch gesteigert, dass auch Pfade untersucht werden, welche nicht zur Lösung führen, da der Algorithmus nicht ausreichend zielgerichtet ist.

Daraus ist leicht ersichtlich, dass das Verfahren nur für sehr kleine Mengen von Eigenschaften in angemessener Zeit ein Ergebnis liefert. Dies limitiert die Expressivität der Objektbeschreibung, vor allem auch, da das Knotenmatchingproblem nur ein Unterschritt des gesamten Matchings ist und mehrfach mit verschiedenen Knotenkombinationen aufgerufen werden muss.

4.1.2 Alternative Lösung der Problemstellung

Im vorherigen Abschnitt wurde erläutert, warum der bisher verwendete Algorithmus zwar das Problem löst, jedoch zu ineffizient ist, um größere Instanzen des NxM-Matchingproblems zu behandeln. In diesem Abschnitt werden alternative Lösungen für das NxM-Matchingproblem untersucht.

In der Literatur finden sich zwei grundsätzliche Lösungsansätze (Burkard, 1999; Gale und Shapley, 1962; Marie und Gal, 2007; Karp, 1980; Kuhn, 2005; Munkres, 1957): die Familie der Stable-Marriage-Algorithmen (zuerst in Gale und Shapley, 1962) und das Maximum-Weighted-Bipartite-Graph-Matching

(vgl. Karp, 1980). Beiden ist gemeinsam, dass sie nicht von der Gesamtmenge aller möglichen Matches ausgehend einzelne Matches ausschließen, sondern die 1:1-Zuordnung von unten aufbauen, indem einzelne Matches ausgewählt und nötigenfalls ausgetauscht werden. Prinzipiell vergrößert sich die Gesamtähnlichkeit also, bis eine optimale Zuordnung gefunden wird. Durch diese Herangehensweise ist es bei beiden Verfahren möglich, polynomielle Komplexität zu erreichen.

Die Stable-Marriage-Algorithmen (Gale und Shapley, 1962) benutzen für die Bestimmung einer optimalen Zuordnung nicht die Maximierung der Ähnlichkeitssumme, sondern ein Optimalitätskriterium, bei dem lokale Stabilität angestrebt wird. Eine Zuordnung ist dann stabil, wenn zwei nicht gematchte Objekte sich nicht ähnlicher sind als ihren jeweiligen Matchpartnern. Diese Verfahren können grundsätzlich auch 1:n-Zuordnungen erstellen und wurden ursprünglich für asymmetrische Ähnlichkeitsmaße entwickelt.

Das Maximum-Weighted-Bipartite-Graph-Matching (Karp, 1980), hat dasselbe Optimalitätskriterium der Maximierung der Ähnlichkeitssumme wie der ursprüngliche Knotenmatcher. Es beruht auf der Suche nach minimalen alternierenden Pfaden über einen bipartiten Graphen mit den zu matchenden Objekten als Knoten und den möglichen Matches als Kanten zwischen zwei Knoten aus N und M. Dabei werden Kanten in einem Pfad alternierend als zur Lösung und nicht zur Lösung gehörend angenommen und bei Hinzunahme eines neuen Matches zum Pfad getauscht.

Stable-Marriage

Das Stable-Marriage-Verfahren wurde zuerst von Gale und Shapley (1962) eingeführt. Sie diskutieren das NxM-Matchingproblem anhand des Problems der Zulassung von Studenten an Universitäten und der Bildung stabiler Ehen, woraus der Name „Stable Marriage“ resultiert.

Das Optimalitätskriterium des Verfahrens ist es, eine optimale stabile Zuordnung zu finden. Eine stabile Zuordnung soll vorliegen, wenn die bestehenden Matches nicht zu Gunsten eines nicht bestehenden Matches aufgegeben werden können. Dazu wird angenommen, dass jedes Element einer der beiden Mengen N, M eindeutige Präferenzen für alle Elemente der anderen Menge hat. Diese lassen sich für den Fall des Matchingverfahrens aus den Ähnlichkeiten ableiten, worauf ich später noch gesondert eingehen werde.

Zunächst ist zu definieren, was Stabilität einer Zuordnung im Hinblick auf die Präferenzen der einzelnen Elemente bedeutet. Dann kann die Optimalität einer stabilen Zuordnung geklärt werden.

Definition 4.1.2
Eine Zuordnung Z ist genau dann instabil, wenn es ein Paar i, p gibt, bei dem i p mehr präferiert als seinen in Z zugeordneten Partner und p seinerseits i mehr präferiert als seinen in Z zugeordneten Partner. Ansonsten ist sie stabil.

Optimale Stabilität soll dann bedeuten, dass keines der Elemente der stabilen Zuordnung in einer anderen stabilen Zuordnung mit einem Partner gematcht ist, den es mehr präferiert. Dies ist im Allgemeinen jedoch nicht möglich, da folgender Fall eintreten kann: Mit $i1, i2 \in N$ und $p1, p2 \in M$, präferiert $i1$ $p1$ und $i2$ $p2$, umgekehrt jedoch präferiert $p2$ $i1$ und $p1$ $i2$. Es ist also keine Zuordnung möglich, die allen vier Elementen gerecht wird. Um diesen Konflikt zu lösen, kann Optimalität nur bei Dominanz der einen Menge über die andere definiert werden.

Definition 4.1.3
Eine stabile Zuordnung ist genau dann optimal für eine Menge A, wenn jedes $a \in A$ in einem Match mit mindestens ebenso hoher Präferenz ist wie in jeder anderen stabilen Zuordnung.

Es ist leicht ersichtlich, dass bei symmetrischem Ähnlichkeitsmaß die N-optimale stabile Zuordnung gleich der M-optimalen stabilen Zuordnung ist.

Um das Konzept der optimalen stabilen Zuordnung für die Lösung des NxM-Matchingproblems zu verwenden, muss zunächst gesichert sein, dass für alle möglichen Mengen N, M eine stabile Zuordnung existiert.

Theorem 4.1.4
Es existiert immer eine stabile Zuordnung.

Den Beweis dafür führen Gale und Shapley an Hand der Angabe eines iterativen Algorithmus, welcher eine solche Zuordnung konstruiert. Der Stable-Marriage-Algorithmus benutzt ein System von vorgeschlagenen und vorläufig angenommenen Matches, um eine optimale stabile Zuordnung zu erreichen. Im Folgenden werde ich deswegen den Begriff *Vorschlag* von i an p verwenden, wenn aus Sicht des Elements i das Match mit p für die 1:1-Zuordnung günstig ist und den Begriff *vorläufiges Match* verwenden, wenn aus der Sicht von p das Match (i, p) zur Zeit Teil der 1:1-Zuordnung ist.

Als Anfang machen alle $i \in N$ einen Vorschlag für ein Match an das von ihnen am höchsten präferierte Element aus M. Jedes $p \in M$ bildet nun ein vorläufiges Match mit dem am meisten präferierten Vorschlagenden i und weist alle anderen ab. Es ist natürlich durchaus möglich, dass für einige p kein Vorschlag vorlag. In der nächsten Runde machen nun alle zuvor abgewiesenen i, die sich ja noch nicht in einem vorläufigen Match befinden, dem nächsten p auf ihrer Präferenzliste einen neuen Vorschlag. Jedes p wiederum sucht

sich aus den Vorschlägen und dem eventuell schon vorhandenen vorläufigen Match den meistpräferierten Partner und bildet ein vorläufiges Match mit ihm, während alle anderen abgewiesen werden.

Dies wird solange fortgesetzt, bis die Menge der abgewiesenen Vorschlagenden leer ist (Fall $n \leq m$) oder alle $i \in N$ entweder Teil eines vorläufigen Matches oder von allen p bereits abgewiesen worden sind (Fall $m \geq n$). Dann wird aus allen vorläufigen Matches eine 1:1-Zuordnung gebildet.

Eine so erhaltene Zuordnung ist stabil, denn: Angenommen, bei zwei nicht miteinander gematchten i und p präferiert i p gegenüber dem gematchten Element aus M. Dann hat i, bevor es zu dem aktuellen Match kam, einen Vorschlag an p gemacht und wurde zu Gunsten eines von p stärker präferierten Kandidaten abgelehnt. Also präferiert p das derzeitige Match und es gibt keine Instabilität. Andersherum, falls p i seinem jetzigen Matchingpartner vorzieht, folgt, dass i die aktuelle Zuordnung präferiert, da es seinem Partner zuerst den Vorschlag gemacht haben muss. Wiederum liegt keine Instabilität vor.

Diese Zuordnung ist jedoch nicht nur stabil, sondern sie ist auch optimal für die vorschlagende Menge.

Theorem 4.1.5
Beim Ergebnis des Stable-Marriage-Algorithmus ist jedes Element der vorschlagenden Menge in einem Match mit mindestens ebenso hoher Präferenz wie in jeder anderen stabilen Zuordnung.

Beweis: Die Optimalität des Verfahrens wird mit Hilfe eines Induktionsbeweises über die Folge von Zeitpunkten gezeigt, zu denen Matches abgelehnt werden.

Die Induktionsannahme ist, dass noch kein Vorschlag abgelehnt wurde, dessen Match zu einer stabilen Zuordnung führt. Dies gilt trivialerweise für den Beginn des Verfahrens. Betrachtet wird ein Zeitpunkt, zu dem die Annahme gilt und der Vorschlag eines $i1$ abgelehnt wird. Das $i1$ ablehnende p hat also einen anderen Vorschlag beziehungsweise ein vorläufiges Match mit einem $i2$, welches von p gegenüber dem $i1$ präferiert wird. Es gibt nun keine stabile Zuordnung, in der $i1$ und p gematcht sind, denn $i2$ präferiert p gegenüber allen anderen Elementen von M oder wurde bereits von ihnen abgelehnt. Für alle abgelehnten Vorschläge gibt es dabei nach Annahme keine stabile Zuordnung. Würde jetzt $i1$ mit p gematcht, müsste, um eine stabile Zuordnung zu erreichen, $i2$ mit einem Element von M mit geringerer Präferenz gematcht werden. Diese Zuordnung kann also auch nicht stabil sein, denn $i2$ und p würden ihre Matches beide zu Gunsten des gemeinsamen Matches aufgeben.

Also gilt für alle Vorschläge, die abgelehnt werden, dass sie nicht zu einer stabilen Zuordnung führen können, und daraus folgt, dass im Stable-Marriage-Verfahren nur Vorschläge von i an p abgelehnt werden, die in keiner stabilen Zuordnung ein Match bilden können. Also ist die resultierende 1:1-Zuordnung optimal.

Das Stable-Marriage-Verfahren kann also eingesetzt werden, um 1:1-Zuordnungen zu erstellen, die immer einem bestimmten Optimalitätskriterium genügen. Allerdings muss dafür für jedes Element der beiden Mengen, in diesem Fall der Menge N aller Instruktionsprädikate und der Menge M aller Perzeptionsprädikate, eine eindeutige Präferenzordnung der jeweils anderen Menge vorliegen. Diese Präferenzordnung lässt sich über das Ähnlichkeitsmaß zwischen den Prädikaten erstellen. Dabei erscheint es sinnvoll, für den Algorithmus die Ordnung für die dominante, vorschlagende Menge N in Form von Priorityqueues zu realisieren. Priorityqueues sind Datenstrukturen, die eine Menge von Elementen in einer gegebenen Ordnung sortiert vorhalten und damit konstante Zugriffszeit auf das am höchsten präferierte Element gewährleisten. Die Komplexität der Befüllung einer solchen Queue ist $O(\log(k))$ (mit k als Länge der Queue) und, da n Queues mit jeweils m Elementen befüllt werden müssen, ergibt sich für die anfängliche Sortierung eine Komplexität von $O(nm\log(m))$.

Für M kann dann bei der Abarbeitung jedes Vorschlags nach einem Vergleich der Ähnlichkeiten zwischen Vorschlag und vorläufigem Match entschieden werden, welches der beiden Elemente in einem neuen vorläufigen Match behalten und welches abgelehnt wird. Dies entspricht bei einer Iteration über alle $i \in N$ einer einfachen Maximumsbildung über alle Vorschläge und ein möglicherweise vorhandenes vorläufiges Match für jedes $p \in M$. Da es maximal m Vorschläge pro i geben kann, ergibt sich hier die Komplexität $O(nm)$. Die Komplexität des gesamten Algorithmus wird also durch die Komplexität der Vorsortierung dominiert $O(nm\log(m))$.

Mit der Vorsortierung in Queues sieht dann der Algorithmus folgendermaßen aus:

```
procedure stableMarriage(N,M)
  for all i ∈ N do
    for all p ∈ M do
      füge p in die Queue von i ein
    end for
  end for
  VORSCHLAGENDE=N
  while Zuordnung ist nicht stabil und vollständig do
    ABGELEHNTE= ∅
```

```
    for all i ∈ VORSCHLAGENDE do
      p =nächste Präferenz aus Queue von i
      if p hat kein vorläufiges Match then
        (i, p) ist das neue vorläufige Match für p
      else
        iAlt = vorläufiger Matchpartner von p
        if ähnlich((i, p))>ähnlich((iAlt, p)) then
          (i, p) ist das neue vorläufige Match für p
          ABGELEHNTE = ABGELEHNTE ⋃ {iAlt}
        else
          (iAlt, p) bleibt das vorläufige Match für p
          ABGELEHNTE = ABGELEHNTE ⋃ {i}
        end if
      end if
    end for
    VORSCHLAGENDE = ABGELEHNTE
  end while
end procedure
```

Maximum Weighted Bipartite Graph

Die zweite Methode, das NxM-Matchingproblem zu lösen, die ich hier vorstellen möchte, ist Maximum-Weighted-Bipartite-Graph-Matching (vgl. Karp, 1980) — im Folgenden MWBG genannt – an Hand der ungarischen Methode. Die ungarische Methode wurde von Kuhn (zitiert nach Neuveröffentlichung 2005, ursprünglich 1955 veröffentlicht) eingeführt und von Munkres (1957) diskutiert und erweitert.

Das MWBG-Verfahren betrachtet das NxM-Matchingproblem als einen bipartiten Graph $G = (V, E)$ mit V als einer bipartiten Menge der Knoten $V = N \cup M$ und den Kanten $E = \{\{i, p\} \mid i \in N, p \in M\}$ als möglichen Matches. Die Kanten sind ungerichtet und für eine einheitliche Notation werde ich im Weiteren (i, p) für eine Kante schreiben. Zusätzlich sind die Kanten aus E gewichtet über eine Funktion $w\,(i, p)$, im Falle des Knotenmatchings entspricht diese der Ähnlichkeitsfunktion.

Zur Vereinfachung der Argumentation werde ich im Folgenden annehmen, dass die beiden zu matchenden Mengen N und M gleich groß sind. Dies geschieht ohne Einschränkung der Allgemeinheit, denn es ist leicht einzusehen, dass das allgemeine Problem auf diese spezielle Variante zurückgeführt werden kann, wenn der kleineren der beiden Mengen genügend Elemente mit Ähnlichkeit 0 zu allen anderen Elementen hinzugefügt werden. Zumal der Al-

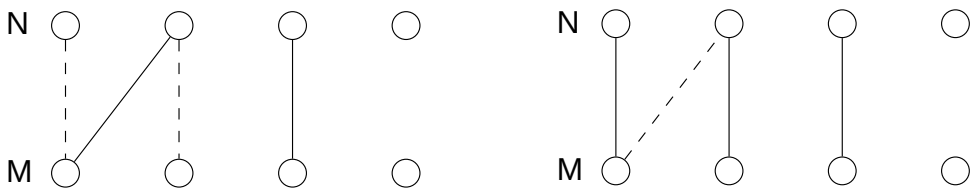


Abbildung 4.1.3: Beispiel für einen anreichernden Pfad bezüglich eines Z und das Ergebnis nach der Anreicherung. Durchgezogene Linien zeigen Matches aus Z an.

gorithmus, der aus der Argumentation resultiert, ebenso für den Fall $n < m$ funktioniert.

Ziel des Verfahrens ist es dann, eine Menge $Z \subset E$ zu finden, so dass jeder Knoten v Teil einer einzigen Kante $z \in Z$ ist und die Gesamtgewichtung $w(Z) = \sum_{z \in Z} w(z)$ maximal ist.

Dies geschieht über sukzessive Bestimmung maximaler anreichernder Pfade bezüglich Z. Zunächst ist dazu zu definieren, was ein anreichernder Pfad ist.

Ist ein Knoten $v \in V$ Teil einer Kante aus Z, so wird v *gematcht* genannt, andernfalls ist er *frei*. Ein zyklenfreier Pfad über G, dessen Kanten abwechselnd in Z enthalten und nicht in Z enthalten sind, wird *alternierend* bezüglich Z genannt.

Definition 4.1.6
Ein anreichernder Pfad *ist ein alternierender Pfad, dessen Endpunkte beide frei sind.*

Die beiden Endkanten eines anreichernden Pfades sind also nicht Teil der Zuordnung Z. Hat man einen anreichernden Pfad P gefunden und bildet die symmetrische Differenz zu Z, $Z' = P \oplus Z$, erhält man eine neue Zuordnung, die ein Match mehr umfasst als die ursprüngliche Zuordnung (siehe Abbildung 4.1.3). Dabei werden die Matches, die Teil des Pfades sind, gegen neue Matches ausgetauscht, an denen auch die bisher freien Knoten beteiligt sind. Die Anzahl der gematchten Knoten wird also um zwei erhöht. Dieses Vorgehen soll im Folgenden *Anreichern* von Z durch P genannt werden.

Durch sukzessives Anreichern entsteht eine 1:1-Zuordnung, die bei geeigneter Auswahl der anreichernden Pfade auch maximal sein kann. Die rudimentäre Strukturierung eines Algorithmus zur Berechnung der maximalen 1:1-Zuordnung im MWBG-Verfahren sieht also wie folgt aus:

procedure maximumWeightedBipartitGraphMatching(N,M)
  $Z = \emptyset$

while $|Z| < n$ **do** ▷ Zuordnung Z ist nicht vollständig
finde maximalen anreichernden Pfad P zu Z
reicher Z mit P an: $Z = Z \oplus P$
end while
return Z
end procedure

Um den maximalen anreichernden Pfad zu Z zu finden, wird nun die ungarische Methode angewandt (Kuhn, 2005; Munkres, 1957). Dafür wird eine Labelfunktion $l : V \to \mathbb{R}$ eingeführt, die dazu dient, die Maximalität bei der Auswahl einzelner Matches zu gewährleisten. l ist *zulässig*, wenn für alle $i \in N$ und $p \in M$

$$l(i) + l(p) \geq w(i, p)$$

gilt. Eine zulässige Labelfunktion ist leicht zu finden, beispielsweise könnte jedes Label auf das Maximum aller Kantengewichte gesetzt werden. Auf die Konstruktion einer guten zulässigen Labelfunktion wird später noch einmal genauer eingegangen.

Über die Labelfunkion kann dann eine Menge E_l bestimmt werden, in der alle Kanten (i, p) enthalten sind, für die

$$l(i) + l(p) = w(i, p)$$

gilt. Findet man für diese Menge E_l eine vollständige Zuordnung $Z \subseteq E_l$, so ist diese Zuordnung optimal.

Theorem 4.1.7
Wenn l eine zulässige Labelfunktion und Z eine vollständige Zuordnung in E_l ist, dann ist Z optimal.

Beweis: Bei einer beliebigen vollständigen Zuordnung Z', nicht notwendigerweise $\subseteq E_l$, ist jedes $v \in V$ genau einmal Teil eines Matches $z' \in Z'$. Also ergibt sich

$$w(Z') = \sum_{z' \in Z'} w(z') \leq \sum_{(i,p) \in Z'} (l(i) + l(p)) = \sum_{v \in V} l(v)$$

und somit ist $\sum_{v \in V} l(v)$ eine obere Grenze für beliebige vollständige Zuordnungen. Für das $Z \subseteq E_l$ ist jedoch

$$w(Z) = \sum_{z \in Z} w(z) = \sum_{v \in V} l(v)$$

und demnach $w(Z') \leq w(Z)$, woraus folgt, dass Z optimal ist.

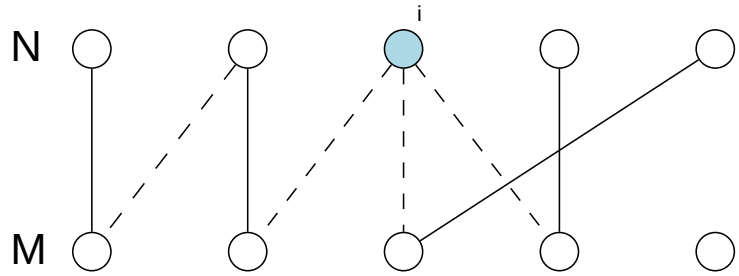


Abbildung 4.1.4: Ein alterniernder Baum mit dem freien Anfangsknoten i und drei alternierenden Pfaden. Die nächste Erweiterung ergibt einen anreichernden Pfad.

Wenn man also durch sukzessives Anreichern eine vollständige Zuordnung erreicht, ist diese Zuordnung dann optimal, wenn die Kanten der anreichernden Pfade in E_l enthalten sind. Es genügt also, die Suche nach den anreichernden Pfaden auf den Graph $G_l = (V, E_l)$ zu beschränken. Die Suche kann als einfache Breitensuche oder Tiefensuche nach einem freien $p \in M$ ausgehend von einem beliebigen freien $i \in N$ über die Nachbarschaftsbeziehungen

$$N_l(a) = \{b \mid (a,b) \in E_l\} = \{b \mid l(a) + l(b) = w(a,b)\}$$

in G_l für Kanten von N nach M und die schon gematchten $z \in Z$ für die Kanten von M nach N realisiert werden. So entsteht in der Suche ein alternierender Baum, ein Baum aus alternierenden Pfaden, mit dem freien Anfangsknoten i als Wurzel, wie in Abbildung 4.1.4 zu sehen. Der Pfadalgorithmus sieht dann in etwa so aus:

procedure findePfad()
  $S = \{$ beliebiges freies i $\}$
  $T = \emptyset$
  repeat
    NACHBARN$= \bigcup_{s \in S} N_l(s)$
    wähle $p \in$ NACHBARN$\backslash T$
    $T = T \bigcup \{p\}$
    if p ist nicht frei **then**
      finde $(i1, p) \in Z$
      $S = S \bigcup \{i1\}$
    end if
  until p ist frei
  rekonstruiere Pfad P von p nach i
  return P
end procedure

Es werden also zwei Mengen $S \subseteq N$ und $T \subseteq M$ benutzt, um über die schon besuchten Knoten Buch zu führen. Am Anfang enthält S nur i. Dann wird iterativ aus den Nachbarn von S in G_l ein p gewählt, um das der alternierende Baum erweitert wird. Ist p gematcht, so wird S und damit der bisher gefundene alternierende Pfad von i über p um den Matchpartner von p erweitert. Dadurch erweitert sich die Menge der Nachbarn und die Nachbarn können neu berechnet werden. Ist p frei so wurde ein anreichernder Pfad zwischen i und p gefunden, der einfach über die Matches in Z und die Nachbarschaftsbeziehungen rekonstruiert werden kann.

Ein Problem, das für dieses Vorgehen noch auftreten kann, ist, dass es gar keinen anreichernden Pfad in dem durch die Labelfunktion l entstandenen Graphen G_l gibt, obwohl diese zulässig ist. Ist dies der Fall, so ist in einer Iteration $T \subset M$ (also nicht $T = M$), woraus ein vollständiges Matching resultieren würde, und $N_l(S) = T$, was bedeutet, dass durch die Untersuchung aller $t \in T$ keine neuen Nachbarn hinzugekommen sind. In diesem Fall ist es nötig, die Labelfunktion anzupassen, um E_l zu erweitern.

Aus einer zulässigen Labelfunktion l kann durch geeignete Subtraktion bzw. Addition auf die einzelnen Label eine neue zulässige Funktion l' entstehen. Dafür definiert man zunächst:

$$\alpha_l = min_{s \in S, t \notin T} \left(l(s) + l(t) - w(s,t) \right)$$

Damit hat man die kleinste Differenz zwischen Summe der Label und Kantengewichtung in der G-Nachbarschaft von S, die noch nicht Teil von G_l ist. l' wird jetzt wie folgt definiert:

$$l'(v) = \begin{cases} l(v) - \alpha_l & \text{für } v \in S \\ l(v) + \alpha_l & \text{für } v \in T \\ l(v) & sonst \end{cases}$$

Dieses l' ist zulässig, da sich bei allen Kanten zwischen Knoten aus S und T und allen Kanten zwischen Knoten, die nicht in S und T liegen, die Summe nicht verändert und somit das Zulässigkeitskriterium weiterhin gilt. Für Kanten zwischen Knoten $\notin S$ und $\in T$ erhöht sich die Summe um α_l und das Kriterium gilt ebenfalls weiterhin. Für Kanten, die von S aus zu Knoten führen, die nicht in T enthalten sind, wird nur die minimale Differenz zwischen Labelsumme und Gewichtung abgezogen und somit gilt auch hier weiterhin $l'(s) + l'(t) \leq w(s,t)$. Außerdem gibt es in dieser Gruppe mindestens eine Kante, für die nun $l'(s) + l'(t) = l(s) - \alpha_l + l(t) = w(s,t)$ gilt. Diese Kante ist Teil von $E_{l'}$ und dadurch gilt jetzt $N_{l'(S)} \neq T$. Dies stellt sicher, dass E_l solange durch Anpassen der Labels erweitert werden kann, bis eine vollständige und damit optimale Zuordnung gefunden ist.

Nun muss für den Gesamtalgorithmus nur noch eine gute initiale zulässige Labelfunktion gefunden werden. Dies kann leicht über die Maxima der Kanten eines $v \in N$ geschehen.

$$l(v) = \begin{cases} max_{p \in M}(w(v,p)) & \text{für } v \in N \\ 0 & \text{für } v \in M \end{cases}$$

Durch die Verwendung der Maxima ist das Zulässigkeitskriterium erfüllt und außerdem sichergestellt, dass jedes $i \in N$ mindestens eine Kante in E_l hat. Der gesamte Algorithmus kann dann wie folgt dargestellt werden:

procedure maximumWeightedBipartiteGraphMatching(N,M)
  $Z = \emptyset$
  for all $i \in N$ **do**
    $l(i) = max_{p \in M}(w(v,p))$
  end for
  for all $p \in M$ **do**
    $l(p) = 0$
  end for
  while $|Z| < n$ **do**
    $S = \{$ beliebiges freies $i \}$
    $T = \emptyset$
    repeat
      NACHBARN$= \bigcup_{s \in S} N_l(s)$
      if NACHBARN$= T$ **then**
        passe Labels an
        NACHBARN$= \bigcup_{s \in S} N_l(s)$
      end if
      wähle $p \in$ NACHBARN$\backslash T$
      $T = T \bigcup \{p\}$
      if p ist nicht frei **then**
        finde $(i1, p) \in Z$
        $S = S \bigcup \{i1\}$
      end if
    until p ist frei
    rekonstruiere Pfad P von p nach i
    reichere Z mit P an: $Z = Z \oplus P$
  end while
  return Z
end procedure

Die Komplexität dieses Algorithmus liegt bei $O(n^2m^2)$. Eine vollständige 1:1-Zuordnung hat n Matches. Bei jedem Anreichern von Z wird die Anzahl

der Matches um eins erhöht und, da mit der leeren Menge begonnen wird, muss n-mal ein anreichernder Pfad berechnet werden.

Für die Berechnung des anreichernden Pfades tragen vor allem die Bestimmung von α_l für das Anpassen der Labels und die Bestimmung der Nachbarschaft zur Komplexität bei. Ein einfaches Berechnen von α_l durch Minimumsbildung über alle Kanten zwischen $s \in S \subseteq N$ und $t \in M/T$ hat höchstens die Komplexität $O(nm)$. Gleiches gilt für die Bestimmung der Nachbarschaft, die alle Kanten zwischen S und M untersucht. Für die Erweiterung des alternierenden Baumes um eine Kante sind höchstens zwei Berechnungen der Nachbarschaft und eine Anpassung der Labels notwendig und, da bei jeder Erweiterung T vergrößert wird und $T \subseteq M$ ist, ergibt sich eine Worst-Case-Komplexität von $O(nm^2)$ für die Berechnung eines anreichernden Pfades. Bei n Berechnungen resultiert die Gesamtkomplexität $O(n^2m^2)$.

4.1.3 Evaluation

Die beiden vorgestellten Verfahren bieten also eine Lösung des Matchingproblems in polynomieller Zeit und sind damit wesentlich effizienter als der vorher verwendete Algorithmus. MWBG-Matching benutzt dabei sogar dasselbe Ähnlichkeitskriterium, so dass gesichert ist, dass bis auf Ambiguitätsentscheidungen bei gleicher Ähnlichkeit von Matches das gleiche Ergebnis, wenigstens aber die gleiche Ähnlichkeitssumme resultiert. Stable-Marriage-Matching benutzt ein anderes Kriterium und so können sich die Ergebnisse in der Ähnlichkeitssumme vor allem bei komplexen Problemstellungen vom MWBG- und dem alten Verfahren unterscheiden. Da jedoch im weiteren Matchingvorgang nur Ergebnisse von den Verfahren untereinander verglichen werden müssen, sind die Unterschiede unerheblich.

4.2 Teilgraphmatching

Das Matching von Teilgraphen, die Knoten und Relationenkanten enthalten, ist das Kernstück des Matchingvorgangs. Ziel ist es, einen möglichst genau übereinstimmenden Teil der Graphen zu finden, um diese Information für die Navigation zu nutzen. Dafür wird vorausgesetzt, dass die Ähnlichkeit zwischen Knoten des Instruktions- und Perzeptionsgraphen und den jeweiligen Relationenkanten berechnet werden kann. Für den Anwendungsfall kann davon ausgegangen werden, dass der Instruktionsteilgraph, der gematcht werden soll, ein Ziel enthält, zusammenhängend ist und weniger Knoten hat als der Graph der perzipierten Umgebung.

Das Ziel des Matchingvorgangs ist der Knoten, dessen Entsprechung in der Welt gesucht werden soll. Das kann direkt das Ziel des Gesamtmatchings sein oder ein Knoten, der ein Zwischenziel darstellt und in der Vorverarbeitung bestimmt wurde. Wäre der Graph nicht zusammenhängend, wären die Teile, welche nicht mit dem Ziel zusammenhängen, nicht mit dem Ziel über Relationen in Beziehung gesetzt. Sie sind also für die Erkennung des Ziels nicht von weiterer Bedeutung und es reicht aus, nur den zusammenhängenden Teilgraphen zu betrachten, der das Ziel enthält. Weiterhin ist die Instruktion eine Beschreibung eines Teilstücks der Welt, das in dem perzipierten Ausschnitt möglichst vollständig wiedererkannt werden soll. Es ist davon auszugehen, dass der perzipierte Ausschnitt der Welt weit mehr enthält als die Entsprechung des instruierten Teilgraphen und deswegen der Perzeptionsgraph größer ist als der Instruktionsgraph.

Zusätzlich kann davon ausgegangen werden, dass einige Knoten des Perzeptionsgraphen als schon gematcht angenommen werden. Dies kann zum einen geschehen, wenn diese Knoten bei einem vorherigen Matchingdurchlauf koreferenziert wurden, und zum anderen, da der derzeitige Standpunkt des Agenten in der Umgebung als der Standpunkt im derzeitigen Routenabschnitt der Instruktion angenommen wird. Diese zusätzlichen Informationen können ausgenutzt werden, um die Komplexität weiter zu reduzieren.

Im Gegensatz zum Knotenmatching, bei dem nur die Berechnung der Ähnlichkeit entscheidend ist, ist die Struktur der Lösung des Graphmatchingproblems wichtig, da der mit dem Ziel gematchte Knoten identifiziert werden soll. Es muss also nicht eine Lösung, sondern es müssen alle Lösungen mit höchster Ähnlichkeit gefunden werden. Aus den gefundenen Lösungen kann dann situationsabhängig die für die jeweilige Pfadverfolgungsstrategie passende Lösung durch höhergeordnete Entscheidungsverfahren bestimmt werden.

Grundsätzlich sind für die Lösungsmenge vollständige Teilgraphmatches anzustreben. Partielle Matches können zwar genügend Informationen enthalten, um das Ziel des Matchings zu identifizieren. Es ist aber dem Teilgraphmatchingverfahren nicht möglich, zwischen trivialen Teilgraphen, die auf Grund der Modellierung der Anwendungsdomäne nur in dieser Konstellation vorkommen können (Beispiel: Jeder Pfad hat eine End- und eine Startposition), und Teilgraphen, die ein signifikantes Alleinstellungsmerkmal haben, zu unterscheiden.

4.2.1 Bisher verwendetes Verfahren

Der bisher für das Teilgraphmatchingproblem verwendete Algorithmus aus Helwich (2003) geht auf einen Algorithmus von Poole und Campbell (1995) zurück. Er verwendet dasselbe Prinzip wie der Algorithmus zum Knotenmat-

ching, eine Graphensuche ausgehend von der Menge aller möglichen Matches über dem Gesamtraum der Matchmengen. Aus einer ausgewählten Menge werden zwei neue Mengen durch Streichen von unverträglichen Matches erstellt. Bei Entfernen eines Knotenmatches werden dabei auch alle Matches von Relationen, die mit diesem Match in Verbindung stehen, entfernt. Über den entstehenden Binärbaum von Matchmengen wird die Teillösung mit der größten Ähnlichkeitssumme gesucht. Unverträglichkeit wird dabei analog zur Definition 4.1.1 verstanden.

Anders als beim Knotenmatching wird jedoch in jedem Schritt ein Match betrachtet und für die Lösung angenommen oder abgelehnt. Die beiden entstehenden Mengen beinhalten also entweder das betrachtete Match nicht mehr oder nur noch dieses und alle mit diesem unverträglichen Matches wurden herausgestrichen.

Die Auswahl der nächsten Teillösung geschieht dann über die Berechnung der Ähnlichkeiten der Mengen, für deren Berechnung dann auch die Struktur des Graphen berücksichtigt wird. Der ursprüngliche Algorithmus aus Poole und Campbell (1995) sieht dabei so aus:

procedure pooleCampbellMatching($graph1$,$graph2$)
  C = produktVerknüpfung($graph1$,$graph2$)
  while C ist unverträglich **do**
    k = finde unverträglichen Knoten
    C' =kopiere C
    lösche k aus C
    lege k in C' fest
    RAND = RAND $\bigcup \{C, C'\}$
    C = max(Rand)
    RAND = RAND $\setminus$ C
  end while

Hier hat der Algorithmus eine optimale Lösung C gefunden. Zu Beginn sind alle möglichen Knotenmatches und Relationenmatches, die keine zueinander exklusiven Matchpartner beinhalten, in der betrachteten Menge. Jede Teillösung, die betrachtet wird, hat derzeitig die maximale Ähnlichkeit und durch das Ausschließen unverträglicher Matches steigt die Ähnlichkeit nicht. Wird also eine Lösung gefunden, die keine unverträglichen Knotenmatches hat, so ist es eine Lösung mit maximal möglicher Ähnlichkeit.

Nun müssen zusätzlich noch alle anderen Graphmatches gefunden werden, welche die gleiche Ähnlichkeit aufweisen, sich aber strukturell unterscheiden. Dies geschieht, indem die gefundene optimale Ähnlichkeit als Abbruchkriterium benutzt wird und die Suche solange wiederholt wird, bis eine betrachtete Teillösung unter diesen Schwellenwert fällt.

```
optimum = ähnlichkeit(C)
MATCHES = ∅
while ähnlichkeit(C) ≥ optimum do
  if C ist unverträglich then
    k = finde unverträglichen Knoten
    C' =kopiere C
    lösche k aus C
    lege k in C' fest
    RAND = RAND ⋃ {C, C'}
  else
    MATCHES =MATCHES⋃ { C }
  end if
  C = max(Rand)
  RAND = RAND \ C
end while
return MATCHES
end procedure
```

Der Algorithmus findet alle Teilgraphmatches mit der größten Ähnlichkeit. Sind die besten Matches nur partiell, so werden auch diese gefunden, denn C ist auch verträglich, wenn zu einem Knoten kein einziges Match vorhanden ist. Da partielle Matches mindestens ein Relationenmatch oder Knotenmatch weniger haben als vollständig zugeordnete Matches, haben sie auch eine niedrigere Ähnlichkeit. Also gibt es keine verträglichen vollständigen Matches, wenn ein partielles Match gefunden wird. Soll der Algorithmus nur vollständige Matches finden, so kann am Anfang nach der Erstellung des initialen Matches und, sobald das erste verträgliche Match gefunden wurde, überprüft werden, ob C für jeden Knoten und jede Relation des Instruktionsgraph ein Match beinhaltet, und im negativen Fall der Matchingvorgang abgebrochen werden.

4.2.2 Bewertung des Poole-Campbell-Algorithmus

Poole und Campbell identifizieren das Problem selbst als NP-vollständig und deswegen nicht grundsätzlich in polynomieller Zeit lösbar (man vergleiche auch Lewis, 1978). Dies gilt schon für die Suche nach einer einzigen optimalen Lösung und wird durch die Notwendigkeit, alle optimalen Lösungen zu finden, noch verschlimmert. Allerdings sind für die Praxis die Worst-Case-Fälle nicht von Bedeutung. Das schlimmste Verhalten wird dann erzielt, wenn alle Knoten ungefähr gleich ähnlich zu allen möglichen Matchpartnern sind und auch über die Relationenstruktur kein differenzierter Einfluss auf die Ähnlichkeit eines Graphmatches besteht. Das heißt dann aber auch, dass

in den zu matchenden Graphen der strukturelle Informationsgehalt an sich sehr niedrig ist, was allein durch die Aufgabenstellung weitestgehend ausgeschlossen wird, da der Teil einer Routenbeschreibung und ein perzipierter Weltausschnitt komplexe strukturelle Beschreibungen beinhalten.

Außerdem wird die Effizienz des Algorithmus durch die beiden Auswahlverfahren stark beeinflusst. Die Auswahl des nächsten zu streichenden Knotenmatches bestimmt den Aufbau des Binärbaums und somit den Pfad zu den Lösungen und die Länge des Pfades. Die Auswahl der nächsten zu untersuchenden Teillösung, die von der Ähnlichkeitsbestimmung abhängt, beeinflusst die Zielrichtung der Suche erheblich. Gelingt es für beide Auswahlen also, gute heuristische Verfahren zu entwickeln, kann die Laufzeit für den Normalfall deutlich reduziert werden.

Die Auswahl der nächsten Teillösung folgt dem Prinzip des A*-Algoritmus. Es wird die Ähnlichkeit der Teillösung geschätzt. Dabei muss für die Wahrung der Optimalität die geschätzte Ähnlichkeit immer größer sein als die der konsistenten Graphmatches, die von der Teillösung noch erreichbar sind. Die einfachste Schätzung wäre, die Summe der Ähnlichkeiten aller Knoten- und Relationenmatches der Teillösungen aus Definition 3.4.2 zu nehmen . Eine bessere Heuristik, die im vorliegenden Algorithmus verwendet wird, ist nur jeweils die maximale Ähnlichkeit aller Matches eines Instruktionsknotens oder einer Instruktionsrelation für die Schätzung zu benutzen. Dadurch ist die Schätzung immer genau so groß wie das Graphmatch, das entstehen würde, wenn alle maximalen Matches miteinander verträglich wären.

Die Auswahl des zu streichenden beziehungsweise festzulegenden Knotenmatches kann über die Anzahl der zu dem Match unverträglichen Matches geschehen. Im Algorithmus werden zwei Varianten benutzt, zum einen Auswahl nach minimaler Unverträglichkeit und zum anderen Auswahl nach maximaler Unverträglichkeit. Bis die erste Lösung gefunden wird, wird immer das Knotenmatch mit den wenigsten unverträglichen Matches ausgewählt und neue Teillösungen durch Festlegen und Herausstreichen generiert. Für die Suche aller weiteren Lösungen wird immer ein maximal unverträgliches Match ausgewählt. Man kann argumentieren, dass in jedem Lösungspfad die Anzahl der Teillösungen, die durch Festlegung entstanden sind, gleich der Anzahl der Knotenmatches in einer 1:1-Zuordnung sein muss. Die Länge des Pfades und indirekt auch der Gesamtaufwand, zu dieser Lösung zu kommen, ist also abhängig von der Anzahl der Teillösungen, die durch Herausstreichen entstanden sind.

Durch Minimumsauswahl geschehen die ersten Festlegungen in allen Pfaden früh. Durch das Herausstreichen eines Matches haben alle mit diesem unverträglichen Matches selber ein unverträgliches Match weniger und werden dann möglicherweise im nächsten Schritt ausgewählt. Es entstehen so

früher Situationen, in denen sowohl Herausstreichen als auch Festlegen zu einem Knotenmatch ohne Unverträglichkeiten führen, weil eines der unverträglichen Matches selbst nur das ausgewählte herausgestrichene Match als Unverträglichkeit hat. Dies kommt einer Festlegung gleich. Durch diese frühen Festlegungen wird die Menge auszuwählender Teillösungen möglichst klein gehalten.

Durch Maximumsauswahl werden durch eine Festlegung möglichst viele Knotenmatches ausgeschlossen. Der Informationsgewinn ist also groß. Außerdem wird die Auswahl breiter über alle Matches verteilt, da nach Herausstreichung die zum ausgewählten Match unverträglichen Matches eine Unverträglichkeit weniger haben und so wahrscheinlicher nicht ausgewählt werden.

Beide Strategien erscheinen im Normalfall geeignet, den Aufwand zum Finden der Lösungen zu reduzieren. Insgesamt betrachtet liefert der Algorithmus also die optimalen Ergebnisse und für den Normalfall auch in angemessener Zeit. Eine gute Laufzeit kann aber auf Grund der Natur des Problems nicht garantiert werden. Dieser Teil des Matchingverfahrens ist also hinsichtlich seiner strukturellen Komplexität kaum verbesserungsfähig und eine Verbesserung ist auch nicht unbedingt notwendig. Trotzdem werde ich ein alternatives Verfahren vorschlagen, welches einen anderen Teil des Graphmatchingproblems ausnutzt, um die Lösung zu finden.

Der Algorithmus nach Poole und Campbell verfolgt die grundlegende Strategie, von der Menge aller möglichen Matchkombinationen durch Weglassen einzelner Matches von oben herab auf die Lösung zuzugehen. Man könnte dies einen top-down oder destruktiven Ansatz nennen. Auch liegt der Fokus des Verfahrens auf den Knotenmatches, durch deren Ausschließen die Teillösung verändert wird. Die Relationenmatches werden nur in Abhängigkeit von den Knotenmatches entfernt und haben nur indirekt über die Gesamtähnlichkeitsberechnung eine Auswirkung auf den Prozess. Die für die Erkennung vor allem wichtige räumliche Struktur der Beschreibungen ist aber in den Relationen enthalten. Deswegen sollte bei der Suche nach alternativen Verfahren der Fokus mehr auf die Relationen gelegt werden.

Ein Vorteil des Verfahrens ist, dass durch die top-down-Herangehensweise einfach alle maximalen Lösung garantiert gefunden werden können. Für eine Herangehensweise, die die Lösung von unten durch Hinzufügen aufbaut, müssen alle konsistenten Lösungen untersucht werden, da lokale Minima in der Ähnlichkeit der Teillösungen nicht von globalen Minima zu unterscheiden sind und kein Abbruchkriterium für das Finden einer optimalen Lösung zur Verfügung steht. Andererseits hat eine konstruktive Lösung den Vorteil, dass nur lokal ein kleiner Ausschnitt des Graphen betrachtet werden muss, was möglicherweise für eine bessere Laufzeit ausgenutzt werden kann.

4.2.3 Alternative Lösung – Graphflooding

Ein Graphmatchingverfahren, das die Aspekte der Problemstellung ausnutzt, welche der Poole-Campbell-Algorithmus vernachlässigt, sollte die Lösung von unten her, also konstruktiv, möglichst zielgerichtet an Hand der durch die Relationen vorgegebenen Struktur aufbauen. Dabei soll jeweils nur ein kleiner lokaler Teil des Graphen betrachtet werden, wodurch die Laufzeit begrenzt werden kann, gleichzeitig aber keine Aussagen mehr über die Gesamtlösung getroffen werden können. Es wird dafür ein Algorithmus vorgeschlagen, der den Instruktionsgraphen Knoten für Knoten an Hand eines Graphflooding-Algorithmus matcht. Es werden für miteinander gematchte Knoten alle Relationen, die von diesen Knoten ausgehen, untersucht und Relationenmatches sowie Knotenmatches für die Endknoten der Relationen gebildet. Das Graphmatch wird also in einem Schritt um ein Relationenmatch und alle Knotenmatches der zu den beiden Relationen gehörenden Knoten erweitert, sofern die Knoten noch kein festgelegtes Match haben. Für die neu gefundenen Knotenmatches wird dann der Vorgang rekursiv wiederholt, bis alle Knoten gematcht sind. Das Matching orientiert sich so an der Relationenstruktur des Instruktionsgraphen und es wird sichergestellt, dass nur Relationen- und Knotenmatches aufgenommen werden, die durch die schon in das Graphmatch aufgenommenen Matches legitimiert sind. Es wird dabei davon ausgegangen, dass alle Knoten und Relationen des Instruktionsgraphen gematcht sein müssen.

Zhong et al. (2002) stellen einen Algorithmus vor, der nach dieser Methode vorgeht. Dieser Algortihmus berechnet jeweils die höchste Ähnlichkeit eines über eine Relation vom Ausgangspunkt ausgehenden Teilgraphen rekursiv. Allerdings hat er Probleme mit Zyklen in den Graphen, was in den Instruktionsgraphen häufig vorkommt (vergleiche Abschnitt 2.3.2), und kann nur jeweils das beste Match errechnen. Für die Zwecke des Teilgraphmatchings im Geometrischen Agenten ist der Algorithmus von Zhong et al. also ungeeignet und muss stark modifiziert werden.

Es müssen zunächst alle möglichen Lösungen untersucht werden und deswegen muss jede Instruktionsrelation mit allen Perzeptionsrelationen gematcht werden. Es entsteht dabei für jedes Relationenmatch eine neue Teillösungsvariante, wenn das Relationenmatch und die daraus zwangsläufig folgenden Knotenmatches *geeignet* sind. Geeignet sind die neuen Matches aber nur dann, wenn Knoten und Relationen in ihren Matches nicht exklusiv zueinander sind.

Definition 4.2.1
Ein Knoten- bzw. Relationenmatch ist geeignet, *wenn die beiden Knoten, bzw. Relationen nicht exklusiv zueinander sind.*

Die Auswahl des Ausgangspunktes ist für das Graphflooding von entscheidender Bedeutung. Hier können beispielsweise schon von vornherein bekannte Knotenmatches benutzt werden. Wenn der Instruktionsteilgraph den momentanen Standpunkt des Agenten beinhaltet, kann das als Grundvoraussetzung angenommene Knotenmatch (vgl. Bittkowski, 2005) auf diese Weise gut ausgenutzt werden. Für andere Fälle muss entweder auf schon von vorherigen Matchingdurchläufen bekannte Knotenmatches zurückgegriffen werden oder es können markante Landmarken mit wenigen, eindeutigen Matches benutzt werden. Im Folgenden wird also davon ausgegangen, dass ein Anfangsknotenmatch auf geeignete Weise bestimmbar ist. Zusätzlich können aber auch weitere bekannte Knotenmatches schon festgelegt werden, was die Laufzeit positiv beeinflusst, da der Suchraum eingeschränkt wird.

Entwurf des Algorithmus

Die Ausgangslage für jeden rekursiven Schritt des Algorithmus stellt sich also wie folgt dar: Es gibt ein bisher aufgebautes Teilgraphmatch g, in dem alle bis jetzt festgelegten Knoten- und Relationenmatches gespeichert sind und welches gerade um ein Knotenmatch (i, p) erweitert wurde. Für den Instruktionsknoten i des Matches werden nun alle Relationen, in denen i steht, mit den Relationen, in denen der Perzeptionsknoten p steht, verglichen und Matches gebildet. Dabei müssen für jede Instruktionsrelation alle Matchvarianten, die in den vorangegangenen Iterationen entstanden sind, einzeln betrachtet werden. Allerdings können von vornherein verschiedenstellige Relationen getrennt behandelt werden, so dass nur Relationenmatches mit gleicher Stelligkeit gebildet werden müssen. Der einfachste und in der Anwendung häufigste Fall sind zweistellige Relationen, da dort nur ein einziges neues Knotenmatch entstehen kann. Im Folgenden wird deshalb zunächst das Verfahren nur für zweistellige Relationen abgehandelt und später dann auf höherstellige Relationen verallgemeinert.

function floodGraph$((i, p), rm, g)$
  erweitere g um (i, p) und rm
  MATCHES $= \{g\}$
  INSTRUKTIONSRELATIONEN = alle Relationen mit i
  PERZEPTIONSRELATIONEN = alle Relationen mit p
  if i ist kein Endknoten im Graph **then**
    for all *iRelation* $\in$ INSTRUKTIONSRELATIONEN **do**
      i' = mit i in *iRelation* stehender Knoten
      MATCHES$'= \emptyset$
      for all $g' \in$ MATCHES **do**
        for all *pRelation* $\in$ PERZEPTIONSRELATIONEN **do**

```
                p' = mit p in pRelation stehender Knoten
                rm' = neues Match (iRelation, pRelation)
                (i', p') = neues Match aus Knoten i' und p'
                if (i', p') und rm' sind geeignet then
                  g'' = kopiere g'
                  NM=floodGraph((i', p'), rm', g'')
                  MATCHES' =MATCHES'∪ NM
                end if
              end for
            end for
            MATCHES=MATCHES'
          end for
        end if
        return MATCHES
      end function
```

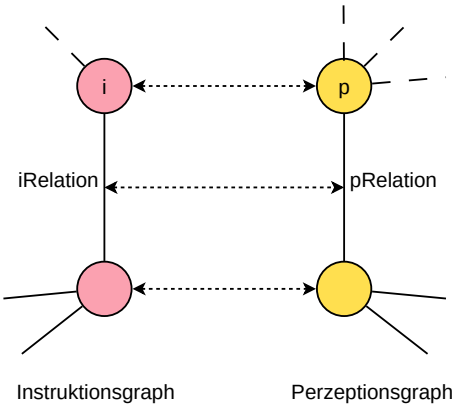
Dies ist das Grundgerüst des Algorithmus. Es entspricht einer Tiefensuche über dem Instruktionsgraph. Dafür wird für das über ein Relationenmatch gefundene Knotenmatch (i, p) über alle Relationen von i iteriert und für jede *iRelation* wird jede mögliche Kombination mit einer Relation *pRelation* von p überprüft. Ist das Relationenmatch und das Knotenmatch der am anderen Ende der Relationen hängenden Knoten geeignet, wird die Rekursion mit diesen Matches und einer Kopie des bisherigen Teilgraphmatches aufgerufen. Von allen Suchpfaden, die mit *iRelation* anfangen und durch die Instruktionsrelationen tieferer Rekursionsschritte entstehen, werden die Teilgraphmatches in MATCHES' gesammelt und dann in der nächsten Iteration für die nächste *iRelation* einzeln untersucht. Ist der Instruktionsknoten i ein *Endknoten* des gerade betrachteten Suchpfades, endet die Rekursion und nur das bisherige Teilgraphmatch g wird zurückgegeben.

Definition 4.2.2
Ein Instruktionsknoten i ist ein Endknoten *im gerade untersuchten Suchpfad, wenn alle iRelation* $\in$ INSTRUKTIONSRELATIONEN *schon gematcht sind.*

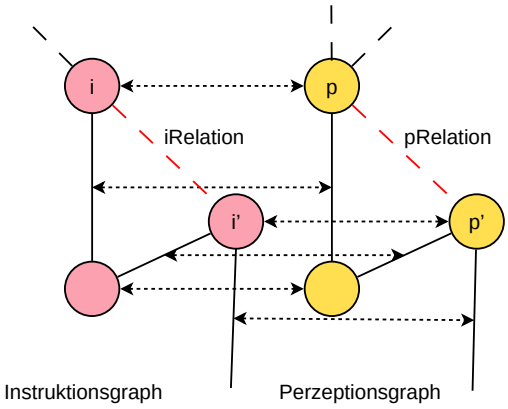
Zusätzlich zu diesem Grundprinzip müssen jetzt noch verschiedene Fälle abgefragt werden, die Abbruchkriterien unterschiedlicher Eingriffstiefe darstellen, weil i', p', *iRelation* oder *pRelation* oder mehrere dieser schon Teil eines anderen Matches in diesem Teilgraphmatch sind (siehe auch Abbildung 4.2.1). Ebenfalls muss darauf geachtet werden, dass kein Knotenmatch, das wegen eines nicht geeigneten Relationenmatch bereits abgelehnt wurde, über eine andere Relation Teil des Teilgraphmatches wird.

Wenn eine Relation *iRelation* oder *pRelation* schon Teil eines Relationenmatches in der Teillösung g ist (Fall 1, Abb. 4.2.1a), kann sie ignoriert werden, da kein neues Match mit dieser Relation möglich ist. Dies ist insbesondere für das Relationenmatch der Fall, das zu dem gerade untersuchten Knotenmatch (i, p) geführt hat.

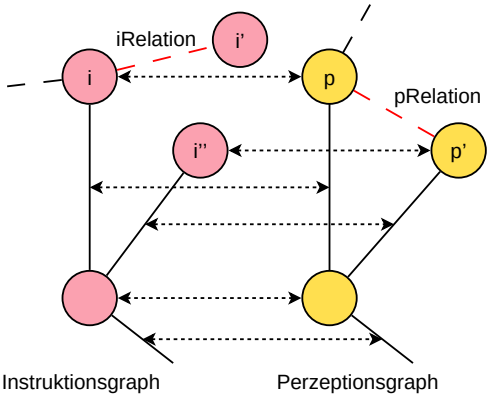
Wenn ein Knoten des Instruktionsgraphen i' schon Teil eines Knotenmatches (i', p') in der Teillösung g ist (Fall 2, Abb. 4.2.1b), die dorthin führende Relation *iRelation* jedoch noch nicht gematcht wurde, ist ein Zyklus im Instruktionsgraphen vorhanden. Für diesen Fall muss für *iRelation* ein geeignetes Relationenmatch mit einer Relation *pRelation* zwischen p und p'



(a) Fall 1



(b) Fall 2



(c) Fall 3

Abbildung 4.2.1: Drei mögliche Sonderfälle, die beim Durchgehen der Relationen auftreten können. Doppelpfeile zeigen Matches an. Durchgezogene Linien sind schon untersuchte Relationen.

gefunden werden. Nach Ergänzung des Relationenmatches muss g' der Menge MATCHES$'$ hinzugefügt werden. Gelingt es nicht, ein geeignetes Match zu finden, ist das derzeitige Graphmatch g' inkonsistent und muss verworfen werden.

Wenn ein Knoten des Perzeptionsgraphen p' schon Teil eines Knotenmatches (i'', p') in der Teillösung g ist, i', *iRelation* und *pRelation* jedoch nicht (Fall 3, Abb. 4.2.1b), kann p' nicht mehr mit i' gematcht werden und auch das Relationenmatch aus *iRelation* und *pRelation* kann nicht Teil der Lösung sein.

Wenn ein Knotenmatch (i, p) schon einmal abgelehnt wurde, als es über ein nicht geeignetes Relationenmatch $(iRelation', pRelation')$ untersucht wurde (Fall 4), darf es nicht mehr Teil eines Teilgraphmatch werden. Dies ist vor allem auch aus Symmetriegründen wichtig, denn es ist dieselbe Konstellation wie für den negativen Fall 2, nur dass zuerst festgestellt wird, dass es keine geeignete Relation gibt und erst dann das Match festgelegt werden soll.

```
function floodGraph((i,p), rm, g)
  erweitere g um (i,p) und rm
  MATCHES = {g}
  INSTRUKTIONSRELATIONEN = alle Relationen mit i
  PERZEPTIONSRELATIONEN = alle Relationen mit p
  if i ist kein Endknoten im Graph then
    for all iRelation ∈ INSTRUKTIONSRELATIONEN do
      MATCHES'= ∅
      for all g' ∈ MATCHES do
        if iRelation hat kein Match in g' then                ▷ Fall 1
          i' = mit i in iRelation stehender Knoten
          if i' hat kein Match in g' then                     ▷ Fall 2
            for all pRelation ∈ PERZEPTIONSRELATIONEN do
              if pRelation hat kein Match in g' then          ▷ Fall 1
                p' = mit p in pRelation stehender Knoten
                if p' = hat kein Match in g' then             ▷ Fall 3
                  rm' = (iRelation, pRelation)
                  (i', p') = neues Match aus Knoten i' und p'
                  if (i', p') noch nicht abgelehnt then       ▷ Fall 4
                    if (i', p') und rm' sind geeignet then
                      g'' = kopiere g'
                      NM=floodGraph((i', p'), rm', g'')
                      MATCHES'=MATCHES'∪ NM
                    end if
```

```
                    end if
                  end if
                end if
              end for
            else                                          ▷ Fall 2
              (i', p') ist Knotenmatch ∈ g'
              pRelation ist Relation zwischen p und p'
              if pRelation existiert then
                if rm' = (iRelation, pRelation) ist geeignet then
                  erweitere g' um rm'
                  MATCHES'=MATCHES' ⋃ {g'}
                end if
              end if
            end if
          end if
        end for
        MATCHES=MATCHES'
      end for
    end if
    return MATCHES
  end function
```

Im Fall 1, 3 und 4 sowie wenn die gebildeten Relationen und Knotenmatches nicht geeignet sind, wird die *floodGraph*-Methode nicht mit einer neu generierten zu erweiternden Teillösung aufgerufen und die Anzahl der Matches in MATCHES′ erhöht sich nicht. Wird im Fall 2 ein geeignetes Relationenmatch gefunden, wird das Match g' erweitert und die Anzahl der Matches in MATCHES′ erhöht sich um eins. Gibt es hingegen kein Relationenmatch, wird die Teillösung verworfen und die Anzahl der MATCHES erhöht sich nicht. Für alle vier Sonderfälle und für den Fall der Nichteignung wird die Rekursion also für die derzeitige *iRelation* nicht fortgeführt.

Bestimmung der Ergebnismenge

Da der Algorithmus sehr komplex ist und somit nicht leicht zu erkennen ist, welche Teilgraphmatches zurückgegeben werden, muss dies hier näher ausgeführt werden. Dafür ist zuerst zu klären, ob der Algorithmus überhaupt terminiert.

Theorem 4.2.3
Der Algorithmus terminiert.

Zunächst ist festzuhalten, dass die Anzahl der Relationen und Knoten und damit auch der möglichen Relationen- und Knotenmatches endlich ist. Die Menge von geeigneten Relationenmatches, beziehungsweise Knotenmatches, ist also ebenfalls endlich. Teilgraphmatches müssen aus geeigneten Relationenmatches und Knotenmatches zusammengesetzt sein. Dabei kann keine Relation und kein Knoten in zwei Matches gleichzeitig vorkommen, so dass es auch hier endlich viele Möglichkeiten gibt.

Jedesmal wenn die Funktion rekursiv aufgerufen wird, wird ein Teilgraphmatch g um ein Relationenmatch $(iRelation, pRelation)$ und ein Knotenmatch (i, p) erweitert und für g verringert sich die Anzahl der noch nicht gematchten Relationen. Es gilt für die beiden Relationen $iRelation$ und $pRelation$ von diesem Zeitpunkt an Fall 1. Weiterhin gilt für alle potentiellen Matches $(iRelation', pRelation')$ Fall 2 und für alle potentiellen Relationenmatches $(iRelation'', pRelation')$ Fall 3, wenn $pRelation'$ eine Relation, die p' als Knoten hat, $iRelation'$ eine Relation, die i' als Knoten hat, und $iRelation''$ eine Relation, die i' nicht als Knoten hat, sind. Die Anzahl der geeigneten Relationenmatches, für die keiner der drei Fälle gilt, verringert sich also mit jedem Aufruf um mindestens eins, was bedeutet, dass pro Teilgraphmatch die rekursive Funktion maximal so oft aufgerufen werden kann, wie es geeignete Relationenmatches gibt. Da die Menge der möglichen Teilgraphmatches und Relationenmatches endlich ist, ist auch die Anzahl der rekursiven Aufrufe endlich und der Algorithmus muss terminieren.

Es bleibt nun zu klären, ob der Algorithmus alle graphkonformen, konsistenten, vollständig zugeordneten Teilgraphmatches findet. Zunächst werden die drei Begriffe definiert.

Definition 4.2.4
Ein Teilgraphmatch ist graphkonform*, wenn für jedes Relationenmatch $(iRelation, pRelation)$ die Knoten i, i' der $iRelation$ mit den Knoten p, p' der $pRelation$ Knotenmatches bilden.*

Die Notwendigkeit für Graphkonformität im Teilgraphmatch ergibt sich direkt aus der Beziehung zwischen Knoten und Relationen in einem Graph (siehe auch Definition 3.4.1). Wenn eine Relation mit einer anderen gematcht ist, müssen auch die Knoten der Relation zueinander passend sein, sonst ist einer der beiden gematchten Teilgraphen kein Graph, denn einer Kante fehlt ein Knoten, und das Teilgraphmatch ist damit irrelevant für die Lösung.

Definition 4.2.5
Ein Teilgraphmatch ist konsistent*, wenn jedes der Knoten- und Relationenmatches geeignet ist und keine Relation und kein Knoten doppelt gematcht wurde.*

Definition 4.2.6
Ein Teilgraphmatch ist vollständig zugeordnet*, wenn jedem Knoten des Instruktionsgraphen genau ein Knoten des Perzeptionsgraphen und jeder Relation des Instruktionsgraphen genau eine Relation des Perzeptionsgraphen zugeordnet ist.*

Nur wenn der gesamte Instruktionsgraph gematcht wurde, ist die Lösung vollständig. Partiell gematchte Graphen sollen zunächst nicht als Lösung betrachtet werden. Das Teilgraphmatching kann nicht darüber entscheiden, ob das partielle Teilgraphmatch für die Navigation relevante Informationen aufweist. Grundsätzlich ist aber durch Änderung der Kriterien zum Hinzufügen der noch nicht vollständigen Teillösungen zur Rückgabemenge auch die Suche nach partiellen Matches möglich.

Theorem 4.2.7
Der Algorithmus gibt genau alle graphkonformen, konsistenten, vollständig zugeordneten Teilgraphmatches zurück, die nach der Einschränkung durch die initiale Festlegung von Knotenmatches möglich sind.

Beweis: Zunächst wird bewiesen, dass jedes Teilgraphmatch, das im Algorithmus erzeugt wird, immer graphkonform ist. Das Match mit der Initialbelegung enthält nur Knotenmatches und ist deswegen graphkonform. Matches werden nur erweitert oder kopiert. Kopien von graphkonformen Matches sind immer graphkonform. Wenn ein Teilgraphmatch um ein Relationenmatch erweitert wird, sind entweder schon beide nötigen Knotenmatches vorhanden oder eins ist vorhanden und das noch fehlende wird ebenfalls hinzugefügt. Teilgraphmatches sind im Algorithmus folglich immer graphkonform, also auch die, welche zurückgegeben werden. Da alle behandelten Teilgraphmatches graphkonform sind, wird auf die explizite Erwähnung im Folgenden verzichtet.

Weiterhin ist zu zeigen, dass kein Teilgraphmatch zurückgegeben wird, das nicht konsistent oder nicht vollständig zugeordnet ist. Der Beweis erfolgt durch Widerspruch.

Wird ein nicht konsistentes Teilgraphmatch g zurückgegeben, so hat g ein nicht geeignetes Match oder es gibt einen Knoten k oder eine Relation r, der oder die mehrfach gematcht wurde. Für nicht geeignete Matches müsste die $floodGraph$-Methode mit einem nicht geeigneten Match aufgerufen oder beim Abbruchkriterium Fall 2 ein nichtgeeignetes Relationenmatch zum Match hinzugefügt worden sein. Beides ist nicht möglich, da vorher auf Eignung der Matches geprüft wird. Für Mehrfachmatching müsste ein Match mit einem i', p', $iRelation$ oder $pRelation$ erstellt worden sein, für das schon ein Match im Teilgraphmatch vorhanden ist. $iRelation$ wird auf

Fall 1 überprüft, kann also nicht zweimal gematcht werden. p' wird immer vor dem Hinzufügen auf Fall 3 überprüft und *pRelation* auf Fall 1. i' wird in Fall 2 überprüft. Wenn schon ein Match (i', p') vorhanden ist, wird nur ein Relationenmatch $(iRelation, pRelation)$ erstellt, für das *pRelation* noch nicht gematcht sein kann, ohne dass p' ebenfalls ein anderes Match als (i', p') hat, da sonst die Graphkonformität verletzt sein würde. p' wurde aber schon oben ausgeschlossen. Es gibt also keine Möglichkeit, ein zweites Match für i', p', *iRelation* oder *pRelation* hinzuzufügen und somit kann der Algorithmus keine inkonsistenten Teilgraphenmatches zurückgeben.

Würde der Algorithmus ein noch nicht vollständig zugeordnetes Match zurückgeben, müsste irgendwann in einem Aufruf der $floodGraph$-Methode mit dem Knotenmatch (i, p) ein Teilgraphmatch zurückgegeben worden sein, für das mindestens eine Instruktionsrelation von i oder ein darüber erreichbarer Instruktionsknoten nicht gematcht wurde. Da alle Teilgraphmatches im Algorithmus graphkonform sind und der Instruktionsgraph zusammenhängend ist, reicht es aus, nur die Relationen zu betrachten, da, sollte ein Knoten nicht gematcht sein, auch alle dazugehörigen Relationen nicht gematcht sein können. Wenn also ein solches unvollständiges Match zurückgegeben wird, muss es in MATCHES gewesen sein. Also war i entweder ein Endknoten des Suchpfads oder das Match wurde in der Iteration über den Instruktionsrelationen jedesmal zu MATCHES$'$ hinzugefügt. Wenn i ein Endknoten im Suchpfad ist, kann es jedoch keine nicht gematchte Instruktionsrelation geben, denn sonst ließe sich die Suche um diese fortführen und i wäre kein Endknoten eines Suchpfades. Wenn das unvollständige Teilgraphmatch am Ende der for-Schleife in MATCHES enthalten ist, ist es in jeder Iteration zu MATCHES$'$ hinzugefügt worden, die zur neuen Menge MATCHES für die nächste Iteration wird. Während der Iteration wird ein Teilgraphmatch aber nur zu MATCHES$'$ hinzugefügt, wenn für die gerade betrachtete Relation *iRelation* über den Aufruf der Methode $floodGraph$ oder über einfache Erweiterung im Fall 2 ein Relationenmatch hinzugefügt wird. Da über alle Instruktionsrelationen iteriert wird, sind auch alle Instruktionsrelationen gematcht. Es kann also vom Algorithmus kein unvollständiges Teilgraphenmatch zurückgegeben werden.

Es bleibt also zu zeigen, dass alle gewünschten Matches zurückgegeben werden, um das Theorem 4.2.7 zu beweisen.

Mit Hilfe der Graphkonformität kann man die duale Beziehung zwischen Knoten und Relationen in den Graphen als Folge aus abwechselnden Knoten und Relationen schreiben $k\ r\ k'\ r' \ldots$. Ein Pfad im Graph von einem Knoten k_1 zu einem Knoten k_n lässt sich so als alternierende Folge mit k_1 am Anfang und k_n am Ende $k_1\ r_1\ k_2\ r_2 \ldots r_{n-1}\ k_n$ beschreiben. Für jeden solchen Pfad im Instruktionsgraphen $i_1\ iRelation_1 \ldots iRelation_{n-1}\ i_n$ gibt es für ein vollständig zugeordnetes, konsistentes Teilgraphenmatch genau einen ensprechenden

Pfad im Perzeptionsgraphen $p_1\ pRelation_1 \dots pRelation_{n-1}\ p_n$, da genau eine Entsprechung für jeden Knoten i_j und jede Relation $iRelation_j$ durch ein geeignetes Match im Teilgraphmatch festgelegt worden ist. Dies kann man zusammenfassen zu einer Folge aus geeigneten Knoten- und Relationenmatches $(i_1, p_1)(iRelation_1, pRelation_1) \dots (iRelation_{n-1}, pRelation_{n-1})(i_n, p_n)$. Umgekehrt gilt aber auch: Gibt es keine Folge aus geeigneten Knoten- und Relationenmatches für jeden Pfad im Instruktionsgraphen, kann das Teilgraphmatch nicht vollständig zugeordnet und konsistent sein, weil es mindestens für einen Knoten oder eine Relation, die Teil des Pfades ist, kein geeignetes Match im Teilgraphmatch gibt.

Man kann also festhalten:

Lemma 4.2.8
Ein Teilgraphmatch g' ist genau dann konsistent und vollständig zugeordnet, wenn es für jeden Pfad im Instruktionsgraphen genau eine Folge aus geeigneten Matches gibt, die den jeweiligen Knoten und Relationen Knoten und Relationen der Perzeption zuordnen.

Der Grundgedanke des Algorithmus ist es, alle möglichen Pfade im Instruktionsgraphen abzugehen. Um zu zeigen, dass er genau alle möglichen Matches, findet wird zunächst argumentiert, dass der Algorithmus ohne jegliche Abbruchkriterien jeden beliebigen Pfad gehen kann, und dann ausgeführt, warum die Abbruchkriterien genau die Teilmatches ausschließen, für die es einen Pfad gibt, der keine eindeutige Folge von geeigneten Matches hat. Um einen beliebigen Pfad w von i nach i' im Instruktionsgraphen zu untersuchen, kann der Algorithmus zunächst von seinem Anfangspunkt i_{start}, der aus dem initialen Knotenmatch resultiert, bis zum Anfangsknoten des Pfades i gehen. Dies ist immer möglich, da der Instruktionsgraph zusammenhängt. D. h. es gibt einen Pfad w_{start} von i_{start} nach i. Da immer alle Relationen, die an einem Knoten hängen, untersucht werden, also auch diejenige, die von i_{start} zum nächsten Knoten von w_{start} führt, die zweite Relation von w_{start} am zweiten Knoten von w_{start} und so weiter, wird dieser Pfad untersucht. Gleiches gilt auch für den Pfad w. Beim Abhandeln einer Instruktionsrelation für einen Pfad werden dabei Teilgraphmatches für alle möglichen Kombinationen erstellt und einzeln weiter untersucht. Zu beachten ist weiterhin, dass Pfade im Instruktionsgraphen nicht in einem Stück hintereinander oder Teile des Pfades nicht in einer bestimmten Reihenfolge behandelt werden müssen. Deswegen ist es nicht von Bedeutung, ob w_{start} und w in Teilen gleich sind oder nicht. Es können also prinzipiell alle gesuchten Teilmatches gefunden werden.

Theorem 4.2.9
Es werden im Algorithmus in jedem Schritt durch Abbruchkriterien nur Teilgraphmatches ausgeschlossen, die für einen Pfad keine Folge aus geeigneten Matches bilden können und deswegen nicht konsistent und vollständig zugeordnet sind oder durch die Initialbelegung ausgeschlossen werden.

Der Beweis wird per Induktion geführt. Es wird angenommen, dass am Anfang jeden Schrittes der Rekursion das Theorem nicht verletzt wurde.

Der Algorithmus wird mit einem einzigen unvollständigen Teilgraphmatch begonnen, das genau die Knotenmatches beinhaltet, die durch die Initialbelegung vorgegeben sind. Dadurch werden genau die Teilgraphmatches ausgeschlossen, die durch die Initialbelegung ausgeschlossen werden, und die Annahme gilt.

Wenn die Annahme zu Anfang eines Schrittes gilt, wenn das Knotenmatch (i, p) hinzugefügt wird, muss sie auch am Ende des Schrittes gelten, wenn keines der Abbruchkriterien gesuchte Teilgraphmatches ausschließt. Es gibt fünf Fälle, die verhindern, dass eine Teilgraphmatchvariante in die Menge MATCHES aufgenommen wird, und diese werden in jeder Iteration geprüft.

Der Normalfall für den Abbruch ist, dass für eine Instruktionsrelation und den daranhängenden Knoten Matches hinzugefügt werden sollen, jedoch das Knotenmatch (i', p') oder das Relationenmatch $(iRelation, pRelation)$ oder beide nicht geeignet sind. Dadurch werden nur Teilgraphmatches ausgeschlossen, die ein ungeeignetes Match beinhalten und deswegen eine Folge aus Matches zu einem Pfad mit i' als Endpunkt hätten, von denen eines ungeeignet ist.

Der Abbruch beim Eintritt von Fall 1 geschieht, weil eine untersuchte Relation *relation* schon mit einem Relationenmatch vertreten ist. Es werden folglich nur Teilgraphmatches ausgeschlossen, die *relation* in zwei Matches beinhalten und nicht konistent sein können. Gleichzeitig wird Mehrfachhinzufügen von Relationenmatches verhindert, wodurch aber keine Teilgraphmatches ausgeschlossen werden.

Der Abbruch beim Eintritt von Fall 2 und 4 geschieht, wenn zwischen zwei Knotenmatches (i, p) und (i', p') kein geeignetes Relationenmatch für die $iRelation$ zwischen i und i' gefunden wurde. Es werden nur Teilgraphmatches ausgeschlossen, die für den Pfad $i\ iRelation\ i'$ keine Folge von Matches haben.

Der Abbruch beim Eintritt von Fall 3 geschieht, wenn für einen Knoten des Perzeptionsgraphen p ein zweites Knotenmatch zum Teilgraphmatch hinzugefügt werden würde. Es werden dadurch also nur nicht konsistente Teilgraphmatches verhindert.

Für alle Abbruchfälle bleibt also die Gültigkeit der Annahme unverändert und die Annahme gilt auch zum Anfang der nächsten Schritte tieferer Rekur-

sionsstufen und am Ende des aktuellen Schritts. Folglich gilt auch Theorem 4.2.9.

Wenn der Algorithmus aber nur Teilgraphmatches ausschließt, die nicht konsistent und vollständig zugeordnet sind und ansonsten alle Teilgraphmatches findet, findet er alle vollständig zugeordneten und konsistenten Teilgraphmatches. Also werden alle und nur alle graphkonformen, konsistenten, vollständig zugeordneten Graphmatches gefunden.

Behandlung von komplexeren Graphen

Es ist also ein Algorithmus für den einfachsten denkbaren Fall gefunden. Es können CRIL-Graphen mit zweistelligen Relationen und jeweils nur einer Relation zwischen zwei Knoten miteinander gematcht werden. Im Anwendungsfall tauchen jedoch Graphen auf, die auch Relationen mit höherer Stelligkeit als zwei und mehrere Relationen zwischen denselben Knoten besitzen. Aber auch diese Fälle können mit einigen Veränderungen nach demselben Grundprinzip behandelt werden.

Bei Graphen, die auch Relationen mit höherer Stelligkeit als zwei besitzen, können die Relationen verschiedener Stelligkeit wie oben bereits erwähnt getrennt behandelt werden, da verschiedenstellige Relationen nicht zu einem geeigneten Relationenmatch gematcht werden können und mindestens ein Knoten der höherstelligen Relation keinen Matchpartner hätte, was die Graphkonformität verletzen würde. Die Kombinationsmöglichkeiten werden so von vornherein eingeschränkt. Dass auch drei- und höherstellige Relationen in gleicher Weise wie oben für zweistellige Relationen dargestellt mit allen anderen Relationen gleicher Stelligkeit gematcht werden können, ist leicht ersichtlich, wenn man bedenkt, dass eine n-stellige Relation durch Reifikation der Relation als Knoten in n zweistellige Relationen aufteilbar ist. Sei der Algorithmus in einem Schritt, wo das Knotenmatch (i, p) gerade hinzugefügt wurde, gerade die n-stelligen Relationen behandelt werden, und die n-stellige Instruktionsrelation *iRelation* sei als Knoten r reifiziert. Nach dem Prinzip des Algorithmus für zweistellige Relationen wird ein Relationenmatch für die Relation zwischen i und r und unter den reifizierten n-stelligen Relationen des Perzeptionsgraphen ein Knotenmatch für r gesucht (Vergleiche Abbildung 4.2.2). Mit den beiden Matches wird die nächste Rekursionsstufe begonnen und nun müssen für die $n-1$ noch nicht gematchten Knoten zusammen mit den zweistelligen Relationen, über die sie an r hängen, wiederum Matches mit den ebenfalls $n-1$ ungematchten Knoten, die mit dem Matchpartner von r verbunden sind, gefunden werden und zwar in allen möglichen Kombinationen $((n-1)!)$. Das Matching von zwei n-stelligen Relationen resultiert also in Rekursionsaufrufen für jeden noch nicht gematchten Knoten

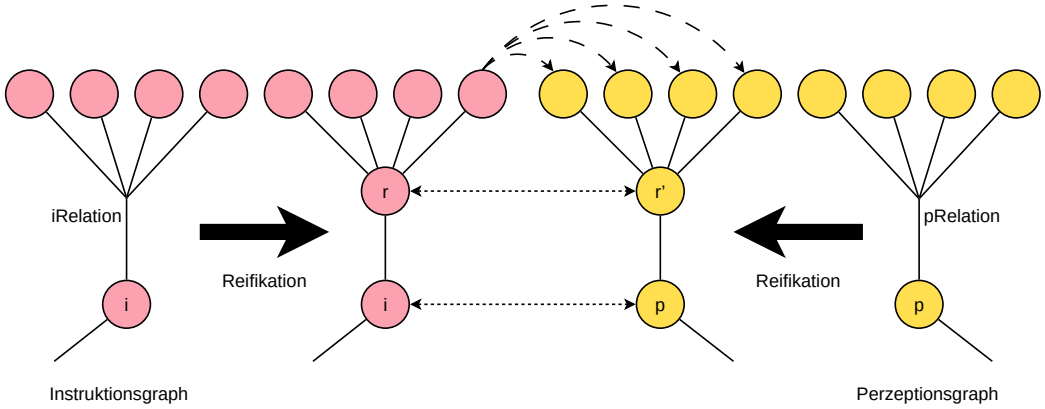


Abbildung 4.2.2: Veranschaulichung der Reifikation am Beispiel einer fünfstelligen Relation.

für alle Kombinationen, die nur geeignete Matches beinhalten. Auch die Abbruchkriterien können analog zu den zweistelligen Relationen übernommen werden. Nur Fall 2 wird dadurch komplizierter, dass mehr als einer der neu hinzukommenden Knoten, jedoch nicht unbedingt alle, schon gematcht sein können. Es können also auch Graphen mit n-stelligen Relationen nach dem Prinzip des Graphflooding gematcht werden, indem über Reifikation motivierte Anpassungen vorgenommen werden.

Bei Graphen, die mehrere Relationen zwischen denselben Knoten haben, reicht auch schon der oben vorgestellte Algorithmus aus, um alle gesuchten Teilgraphmatches zu finden. Wird im Rekursionsschritt eine Instruktionsrelation $iRelation$ untersucht, die denselben Endknoten hat wie eine schon zuvor im gleichen Schritt untersuchte Relation $iRelation'$, greift entweder Fall 2 oder Fall 3. Allerdings wurde zu diesem Zeitpunkt wegen der Tiefensuche schon der gesamte Suchpfad für $iRelation'$ gematcht. Es wäre sinnvoll, schon vorher abzubrechen. Sind zwischen zwei Knoten im Instruktionsgraphen mehrere Relationen derselben Stelligkeit vorhanden, so muss für jede Instruktionsrelation ein geeignetes Relationenmatch mit den Relationen zwischen den gematchten Knoten im Perzeptionsgraphen gefunden werden. Dabei darf aus Konsistenzgründen keine Perzeptionsrelation mehrfach verwendet werden. Es ist jedoch nicht nötig, alle möglichen Kombinationen in einzelnen Teilgraphmatches hinzuzufügen. Verschiedene Kombinationen haben nur über die Ähnlichkeit einen Einfluss auf das Teilgraphmatch und so reicht es aus, nur die beste Kombination zu finden. Dies entspricht dem N×M-Matchingproblem, das schon in Abschnitt 4.1 beim Knotenmatching behandelt wurde, und lässt sich leicht mit den dort vorgestellten Verfahren lösen. Wenn also jedesmal, wenn es ein geeignetes Relationenmatch $(iRelation, pRelation)$ zwischen zwei

geeigneten Knotenmatches (i, p) und (i', p') gibt, gleich alle Instruktionsrelationen zwischen i und i' gematcht und zum Teilgraphmatch hinzugefügt werden, lässt sich die Suche bei Nichteignung früher abbrechen und nur auf die besten möglichen Matchvarianten einengen.

Evaluation des Graphflooding-Algorithmus

Der vorgestellte Algorithmus ist also grundsätzlich geeignet, um alle richtigen Lösungen zu finden, aber wie sich in den Überlegungen zum Behandeln mehrstelliger Relationen schon andeutet, bleibt die Komplexität ein Problem. Für jeden Schritt müssen alle Instruktionsrelationen des Instruktionsknotens mit den Perzeptionsrelationen des gematchten Knotens abgeglichen werden. Bei n Perzeptionsrelationen und m Instruktionsrelationen ergibt das $\frac{n!}{(n-m)!}$ Möglichkeiten, wobei angenommen wird, dass $n \geq m$ ist, da sonst nicht jede Instruktionsrelation gematcht werden kann und die betrachtete Teillösung verworfen werden muss. Alleine dadurch wird klar, dass die Komplexität für den schlimmsten Fall nicht polynomiell bleiben kann. Dennoch bleiben Vorteile, die durch verschiedene Heuristiken noch besser ausgenutzt werden können.

Da der Algorithmus Matches eng an der Struktur des Instruktionsgraphen aufbaut, hängt der Aufwand stärker von diesem ab als vom Perzeptionsgraphen, denn es müssen nur Knoten und Relationen des Instruktionsgraphen gematcht werden. Asymmetrische Problemstellungen können so mit geringem Aufwand behandelt werden. Dies wirkt sich positiv aus, weil der Instruktionsgraph im Normalfall kleiner ist als der Perzeptionsgraph. Sollte wider Erwarten dennoch der Perzeptionsgraph kleiner sein, kann dies frühzeitig erkannt werden, sobald lokale strukturelle Unterschiede ein weiteres Matching verhindern. Die Laufzeit des Algorithmus hängt also immer stärker vom kleineren der Graphen ab als vom größeren. Allgemein werden auch bei gleich großen Graphen viele Knoten- und Relationenmatches, die für sich genommen geeignet wären, schon durch die Graphkonformität ausgeschlossen, weil Vorwissen und Graphstrukturen gut zur Einschränkung ausgenutzt werden können.

Der Fokus des Algorithmus liegt auf den Relationen, die von jedem gematchten Knotenpaar ausgehen. Deren Anzahl wirkt sich maßgeblich auf die kombinatorische Explosion der Teillösungen aus, da für jede Kombination aus Instruktions- und Perzeptionsrelationen eine neue Teilgraphmatchvariante erstellt werden muss. Dies geschieht durch Kopieren der Teillösung und ist deswegen aufwendig. Man kann allerdings argumentieren, dass für den Anwendungsfall die Anzahl der Relationen und auch der Kombinationen, die zu Kopien des Matches führen, beherrschbar bleibt. Wie in Abschnitt 2.3.2 dargestellt sind die Instruktionsgraphen normalerweise klein und haben eine

geringe Anzahl Knoten. Dies begrenzt effektiv auch die Anzahl der Instruktionsrelationen. Da es beliebig viele Relationen zwischen zwei Knoten geben kann, kann es zwar durch die Knoten unbegrenzt viele Relationen geben, allerdings können wie oben beschrieben Relationen zwischen demselben Knotenpaar zusammen behandelt werden, woraus nur eine Teilgraphmatchvariante resultiert. Sie wirken sich also wie eine einzige komplexe Relation aus und die Anzahl dieser komplexen Relationen ist durch die Knoten begrenzt. Hinzu kommt, dass ein großer Teil der Knoten auf Grund der in der Modellierung vorgegebenen Strukturen nur mit wenigen anderen Knoten in Relation steht. Es wird also die durch die prinzipiell geringe Anzahl von Instruktionsknoten begrenzte Anzahl der Instruktionsrelationen mit der durch die Anzahl der Perzeptionsknoten begrenzte Anzahl der Perzeptionsrelationen kombiniert. Aber es werden nur für die Kombinationen, die nicht durch die Abbruchkriterien verworfen werden, wirklich Kopien der Teillösung erstellt und die rekursive Funktion aufgerufen.

Die Abbruchkriterien schränken die Kombinationsmöglichkeiten stark ein. So sind die Matches für alle Instruktionsrelationen schon festgelegt, für die Fall 1 oder Fall 2 gilt; bei Fall 2 muss nur noch überprüft werden, ob es diese Matches auf Basis der vorhandenen Relationen und der Eignung der Relationenmatches in einer konsistenten Lösung geben darf. Fall 3 und 4 schließen eine Reihe von Kombinationen aus, die zu inkonsistenten bzw. nicht vollständigen Matches führen würden, sobald dies erkennbar ist. Die Eignungsprüfung von Relations- und Knotenmatches schränkt die möglichen Kombinationen weiter ein. Für jede Instruktionsrelation werden nur so oft Kopien der bisherigen Teillösung erstellt, wie es Perzeptionsrelationen gibt, aus denen ein geeignetes Relationenmatch und ein geeignetes Knotenmatch resultiert. Für jede Instruktionsrelation $iRelation_k$ gibt es also eine Menge $\textsc{PerzeptionsRelationen}_k \in \textsc{PerzeptionsRelationen}$, in der alle Perzeptionsrelationen sind, die geeignete Matches bilden können. Sind zwei Instruktionsrelationen $iRelation_1$ und $iRelation_2$ ähnlich, so sind die Mengen $\textsc{PerzeptionsRelationen}_1$ und $\textsc{PerzeptionsRelationen}_2$ gleich oder zumindest in großen Teilen gleich. Sind $iRelation_1$ und $iRelation_2$ nicht ähnlich oder sogar exklusiv zueinander, so sind $\textsc{PerzeptionsRelationen}_1$ und $\textsc{PerzeptionsRelationen}_2$ disjunkt oder zumindest ist die Schnittmenge nur klein. So lassen sich die Instruktionsrelationen in Gruppen einteilen, die eine Quasipartition bilden, und das kombinatorische Problem kann weitestgehend in jeder dieser Gruppen seperat gelöst werden. Diese Seperierung hängt entscheidend von der Modellierung der Taxonomie ab. Je diversifizierter die Informationen über die Domäne sind, die in der Taxonomie kodiert sind, desto besser kann der Algorithmus die Suche einengen.

Als Beispiel stelle man sich eine Problemstellung mit vier Instruktionsre-

lationen und neun Perzeptionsrelationen vor. Zwei der Instruktionsrelationen bilden eine Gruppe und können nur mit drei (für beide dieselben drei) der Perzeptionsrelationen geeignete Matches bilden, die anderen zwei können mit diesen drei Perzeptionsrelationen keine geeigneten Matches bilden, sondern bilden mit den verbleibenden sechs Perzeptionsrelationen Matches. Anstelle der $9 \times 8 \times 7 \times 6 = 3024$ Möglichkeiten müssen nur $(3 \times 2)\,(6 \times 5) = 180$ Möglichkeiten untersucht werden. Man sieht, dass sich die Kombinationen durch solche Konstellationen, die im Anwendungsfall häufig auftauchen, stark reduzieren. Was ebenfalls auffällt ist, dass die einzelnen Instruktionsrelationen nicht gleich sind, denn einige erzeugen weniger Kombinationen als andere. Insbesondere gilt dies für Instruktionsrelationen, die nur ein einziges geeignetes Match haben und auch für solche, die gar nicht gematcht werden können.

Dies ist ein Ansatzpunkt für eine Heuristik. Es ist sinnvoll, Instruktionsrelationen früher im Rekursionsschritt zu behandeln, je geringer die Zahl ihrer möglichen Matches ist. Werden Relationen, die überhaupt keine geeigneten Matches haben, zuerst behandelt, wird abgebrochen, bevor es durch geeignete Matches anderer Instruktionsrelationen zu Kopien und einen Abstieg in tiefere Rekursionsschritte kommt, deren Teilergebnisse am Ende ohnehin wieder verworfen werden müssen. Auch Relationen mit nur einem Match sollten möglichst früh im Schritt behandelt werden, da für sie zwar nur eine neue Kopie erstellt werden muss, dies aber für jedes schon bis dahin erstellte Teilgraphenmatch getan wird. Dieselbe Argumentation gilt generell für Relationen mit wenigen Kombinationsmöglichkeiten. Zwar können Lösungsvarianten in tieferen Rekursionsschritten verworfen oder stark vervielfältigt werden, so dass mit lokalen Informationen keine Aussagen darüber getroffen werden können, wieviele Matches aus den tieferen Rekursionsschritten zurückgegeben werden, aber dennoch erscheint es sinnvoll, auf diese Weise die Anzahl der Kopien vor einem Abbruch möglichst klein zu halten.

Ein weiterer Grund, möglichst mit den Relationen mit niedriger Anzahl möglicher Matches zu beginnen, sind Zyklen im Graphen. Viele der behandelten Instruktionsgraphen haben Zyklen und die Behandlung der letzten Relation, die den Zyklus schließt, ist mit wenig Aufwand verbunden, weil sie durch Fall 2 geschieht und dabei keine Kopie des gegenwärtigen Teilgraphmatches erstellt wird und keine Rekursion nötig ist. Wenn nun Relationen mit niedriger Matchzahl zuerst behandelt werden, hat die Relation, auf welcher der Algorithmus zu einem Knoten zurückkehrt, immer eine höhere Anzahl möglicher Matches als die, auf der er ihn verlassen hat. Es werden also mehr rekursive Aufrufe und Kopien der Teilgraphmatches vermieden. Da das Schließen von Zyklen eine berechnungstechnisch günstige Möglichkeit ist, Lösungen auszuschließen, sollten zudem Relationen, für die Fall 2 gilt, eine hohe Priorität haben und möglichst vor allen anderen behandelt werden.

Ein weiterer Ansatzpunkt für eine Heuristik zur Sortierung der Instruktionsrelationen wäre der Aufwand, der im nächsttieferen Rekursionsschritt mit den Knotenmatches nötig sein wird. Dieser kann über die Anzahl der Instruktionsrelationen und das Verhältnis von Perzeptionsrelationen zu Instruktionsrelationen für jedes mögliche Knotenmatch grob abgeschätzt werden. Je mehr Instruktionsrelationen es überhaupt gibt, desto stärker verzweigt der Suchbaum, was mehr Aufwand für die Abarbeitung bedeutet und je mehr Perzeptionsrelationen es im Vergleich zu den Instruktionen gibt, desto größer ist die Anzahl an Kombinationen, da für eine Instruktionsrelation mehr potentielle Matchpartner vorhanden sind. Da jedoch dieser Aufwand auch abhängig von den Perzeptionsrelationen und somit für jedes Knotenmatch verschieden ist, müssen für eine Bewertung der Relation die Ergebnisse der einzelnen Matches geeignet zusammengefasst werden, was beispielsweise durch eine Maximumsbildung möglich wäre. Alles in allem erscheint aber die heuristische Aussage, die getroffen werden kann, als unsicher, weswegen auf eine weitere Gewichtung der Relationen durch dieses Kriterium verzichtet wird. Allerdings entsteht durch die Betrachtung des heuristischen Kriteriums auch ein weiteres Eignungskriterium für Knotenmatches. Ein Knotenmatch kann nur zu einer vollständigen Zuordnung führen, wenn die Anzahl der Perzeptionsrelationen größer oder gleich derjenigen der Instruktionsrelationen ist. Dies kann während des Sortiervorgangs und bei Erstellung des Knotenmatches überprüft werden und es können so weitere Matches von vornherein ausgeschlossen werden.

Als Ergebnis der Komplexitätsbetrachtung kann man also zusammenfassen, dass der Algorithmus lokal potentiell sehr viele Kombinationen untersuchen muss, dies aber in der Praxis durch die Problemstellung, in der nur relativ kleine Anzahlen an Relationen vorkommen, beschränkt wird und viele Teillösungen auch schon in den nächsten Schritten verworfen werden. Durch geeignete Heuristiken und Abbruchkriterien lässt sich sicherstellen, dass dieses Ablehnen von Teillösungen in der Regel möglichst früh stattfindet.

4.2.4 Vergleich des Graphflooding-Algorithmus mit dem Poole-Campbell-Algorithmus

Wenn man nun den neuen Algorithmus mit dem Verfahren nach Poole und Campbell vergleicht, gibt es eine Reihe von Vor- und Nachteilen. Im Graphflooding werden nur Teillösungen betrachtet, die weniger Matches enthalten als ein vollständig zugeordnetes Teilgraphmatch und konsistent sind, während im Poole-Campbell-Algorithmus nur Teillösungen betrachtet werden, die nicht konsistent sind und mehr Matches enthalten als ein vollständig zugeordnetes Teilgraphmatch. Graphkonformität bleibt bei beiden Algorith-

men ständig erhalten, wobei beim Poole-Campbell-Algorithmus mit einem Knotenmatch alle dann Graphkonformität verletzenden Relationenmatches entfernt werden, während beim Graphflooding nur Relationenmatches hinzugefügt werden, wenn die entsprechenden Knotenmatches auch in der Teillösung vorhanden sind. Kopien der Teillösung werden bei beiden Verfahren nur erstellt, wenn sich die Knotenmatches verändern, fehlende Relationen werden beim Graphflooding durch Fall 2 relativ effizient ohne Kopie behandelt. Für eine vollständig zugeordnete Lösung müssen soviele Knotenmatches endgültig festgelegt, bzw. über eine Relation hinzugefügt werden, wie es Knoten im Instruktionsgraphen gibt. Sind es n Knoten, so genügen für eine vollständige Lösung durch das Graphfloodmatching auch genau n Kopien, da nur dann eine Kopie erstellt wird, wenn ein Knotenmatch zur Lösung hinzugefügt wird. Beim Poole-Campbell-Matching sind es n oder mehr Kopien, weil auch eine Kopie erstellt wird, wenn durch das Auswahlverfahren Knotenmatches ausgewählt werden, die nicht in der Lösung vorhanden sind. Auch bei der Größe der Teilmatches ist das Graphflooding im Vorteil, denn der Poole-Campbell-Algorithmus besitzt wie oben dargestellt generell größere Teillösungen und der einzelne Kopiervorgang ist damit aufwendiger. Andererseits hat der Graphflooding-Algorithmus den Nachteil, dass er erst terminiert, wenn alle vollständig zugeordneten Teilgraphmatches gefunden sind, während im anderen Verfahren eine der besten Lösungen sicher zuerst gefunden wird und nach dem Finden aller besten Lösungen abgebrochen werden kann, so dass insgesamt wieder weniger Kopien nötig sein könnten.

Für die Frage, wieviele Kopien insgesamt in einem Aufruf des Algorithmus anfallen, müssen jedoch nicht nur die Kopien berücksichtigt werden, die zu den Lösungen geführt haben, sondern auch die, die für wieder verworfene Teilgraphmatchvarianten benutzt wurden. Wenn man nur die für Kopien maßgeblichen unterschiedlichen Kombinationen von Knotenmatches betrachtet, gibt es weit mehr inkonsistente Teillösungen, die größer als vollständige Lösungen sind, als konsistente Teillösungen, die kleiner sind. Beide Algorithmen schränken jedoch ihren jeweiligen Suchraum über strukturelle Ansätze und Heuristiken ein, so dass niemals alle diese Teillösungen im Algorithmus untersucht werden müssen. Für den Poole-Campbell-Algorithmus besteht die strukturelle Einschränkung darin, dass ein Binärbaum aus Teillösungen verwendet wird und für jede Teillösung nur zwei Folgezustände möglich sind. Dabei wird ein ausgewähltes Knotenmatch festgelegt oder entfernt und so kann es tiefer im Ast nur noch Teillösungen geben, für die dieses Match schon bearbeitet wurde. Im allerersten Schritt werden alle Teilgraphenlösungen von der Betrachtung ausgeschlossen, für die nur ein Knotenmatch herausgestrichen oder festgelegt ist, welches nicht gleich dem durch die Heuristik gewählten ist. Der Graphflooding-Algorithmus schränkt auf ähnliche Weise

die Matches über die Strukur ein. Auch bei ihm werden Knotenmatches entweder als Teil der Lösung angenommen oder implizit abgelehnt und somit Teillösungsvarianten, für die dieses Knotenmatch nicht behandelt ist, von der Betrachtung ausgeschlossen. Zusätzlich werden jedoch schon alle Knotenmatches, die über die Struktur der beiden Graphen nicht möglich sind, von der Auswahl durch Heuristiken ausgeschlossen, was einen weiteren Vorteil in der Anzahl der untersuchten Teilgraphmatches bringt. Das Verwerfen bzw. Nichtweiterverfolgen solcher graphstrukturell nicht passenden Matches geschieht im Poole-Campbell-Algorithmus nur indirekt durch die Ähnlichkeitsbegrenzungen, weil durch das Wegfallen von Relationenmatches bei Herausstreichen strukturell zentraler Knotenmatches die Ähnlichkeit stark fällt. Für die Heuristik zur Auswahl von zu betrachtenden Matches wird bei beiden Algorithmen auf die Anzahl der gerade möglichen Matches zurückgegriffen.

Abschließend zusammengefasst ist die Stärke des Poole-Campbell-Algorithmus eindeutig die Betrachtung der Gesamtähnlichkeit, weil dadurch der Algorithmus schnell abgebrochen werden kann, sobald die erste Lösung gefunden wurde. Andererseits ist der Algorithmus nur durch die Ähnlichkeit zielgerichtet. Das Graphflood-Matching hingegen ist wesentlich zielgerichteter auf die Lösung und kann potentiell durch Informationen aus der Graphstruktur viele Kombinationen ausschließen. Allerdings bringt dies den Nachteil mit sich, dass keine Betrachtungen über die globale Ähnlichkeit angestellt werden können. Möglicherweise wäre eine Kombination aus beiden Verfahren gewinnbringend, worauf ich in Abschnitt 5.3 noch weiter eingehe.

4.3 Vorverarbeitung – Auswählen von Teilgraphen

Unter Vorverarbeitung wird hier das Vorbereiten der Eingabedaten des Matchings, die Auswahl der zu matchenden Teile und notwendige Erweiterungen der Graphen durch Schlüsse zusammengefasst. Ziel des Matchingverfahrens ist, eine Entität in der Welt zu identifizieren. Eingabedaten sind dabei der Instruktionsgraph, der Perzeptionsgraph und der CRIL-Knoten im Instruktionsgraph, der das Ziel darstellt. Als Ausgabe wird eine Menge von Matches der beiden Graphen, beziehungsweise Teilstücken dieser Graphen, erwartet, in denen das Ziel in der Perzeption identifiziert wurde.

Um dies zu erreichen, ist es zunächst sinnvoll festzulegen, welche Teile des Instruktionsgraphen als Eingabe für das Graphmatching benutzt werden sollen. Der Instruktionsgraph enthält die Beschreibungen für alle Aktionen der Routenbeschreibung, für die Identifikation des Ziels ist jedoch nur der Teil relevant, der die aktuelle Aktion beschreibt.

Der Perzeptionsgraph enthält zu Beginn des Matchingvorgangs alle Land-

marken sowie die über Tracks vom eigenen Standpunkt ausgehenden Pfade. Landmarken sind aber noch nicht über Regionen in Beziehung gesetzt. Sind im für die aktuelle Aktion relevanten Teilgraphen der Instruktion Regionen enthalten, so muss der Perzeptionsgraph mit geeigneten Regionen erweitert werden, bevor ein Match gesucht werden kann.

4.3.1 Problemstellung der Vorverarbeitung

Das Matchen von Teilgraphen ist komplex, wie in Abschnitt 4.2 dargelegt. Es ist deswegen sinnvoll, die zu matchenden Graphen möglichst klein zu halten.

Zum einen kann das erreicht werden, indem der Perzeptionsgraph nicht um alle möglichen Regionen bei allen Landmarken erweitert wird, sondern nur Regionen ergänzt werden, die auch im Instruktionsgraphen vorkommen und deren Referenzobjekte eine hohe Ähnlichkeit zu den Referenzobjekten der instruierten Regionen haben. Dies entspricht einer Filterung aller möglichen Regionen nach räumlicher Relation, in der sie stehen, und einfacher Knotenähnlichkeit der Referenzobjekte und kann die Größe und Verzweigung des Perzeptionsgraphen stark reduzieren.

Eine zweite Möglichkeit, die mit der ersten kombiniert werden kann, ist das Aufteilen von Instruktionsteilgraphen mit mehreren Regionen in kleinere Einheiten, die nur eine Region haben (vergleiche Abbildung 4.3.1). Diese können dann einzeln gematcht und danach wieder zusammengefügt werden.

Die Kombination aus beiden Möglichkeiten bringt den Vorteil, dass für

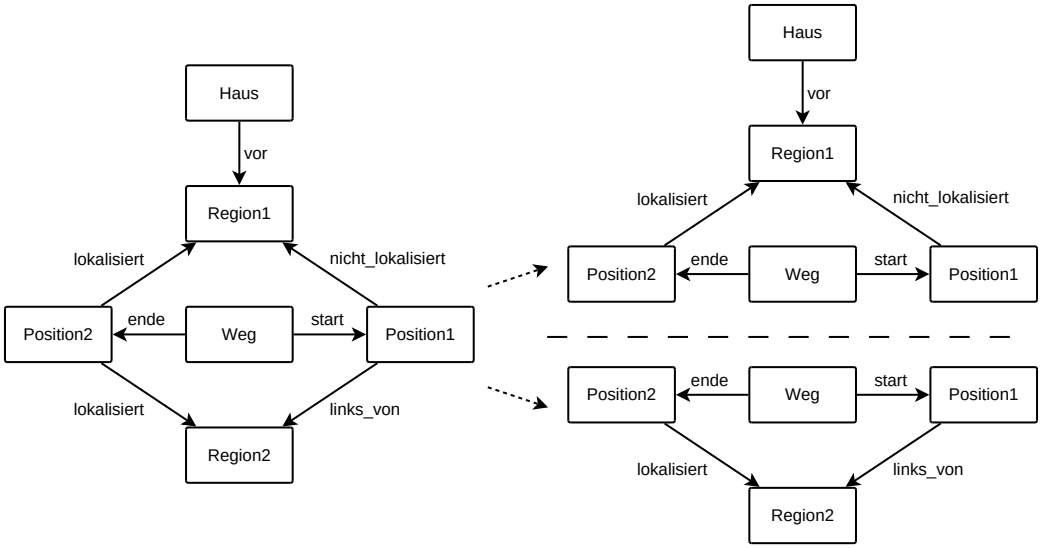


Abbildung 4.3.1: Ein Instruktionsteilgraph, der einen Weg durch zwei unterschiedliche Regionen beschreibt, aufgeteilt in zwei Graphen mit jeweils einer Region

jeden Matchingvorgang der Perzeptionsgraph nur um die durch die räumliche Relation möglichen Matches für eine einzige Region erweitert werden muss. Selbst wenn also, wie im Beispiel von Abbildung 4.3.1 dargestellt, nur eine Region und ihre Referenzobjekte nicht in dem zu matchenden Instruktionsgraphen enthalten sind, können dadurch im Perzeptionsgraphen die Hälfte der Regionen und die Lokalisationsrelationen für diese wegfallen.

Nach diesem Prinzip kann versucht werden, in der Vorverarbeitung des Matchings den Berechnungsaufwand durch ein Divide-and-Conquer-Verfahren zu reduzieren. Bei Helwich (2003) wird dies ebenfalls vorgeschlagen, jedoch nicht abschließend geklärt, wie mit mehreren, unterschiedlichen und sich möglicherweise widersprechenden Ergebnissen umgegangen werden kann. Zu Beginn dieser Diplomarbeit ist also zwar eine Aufteilung in mehrere Regionsteilgraphen möglich, das Zusammenfügen der einzeln gematchten Regionsteilgraphen jedoch noch nicht implementiert. Das Matching befindet sich also in einem Zustand, in dem nur Teilgraphen der Instruktion verarbeitet werden können, die durch eine einzige Region beschrieben werden. Dies limitiert die Expressivität der Routenbeschreibungen unnötig. Im Projekt 'Graphische Instruktion künstlicher Agenten' vom Wintersemester 2008/2009 wurde z. B. erkannt, dass für die Verarbeitung von Skizzen zur Routeninstruktion eine Beschreibung durch mehrere Regionen gleichzeitig sinnvoll wäre, da so räumliche Konstellationen, die in graphischen Darstellungen stärker zu Tage treten als in sprachlicher Beschreibung, besser übertragen werden könnten. Dies deckt sich mit den Ergebnissen von Skubic et al. (2004), wo ebenfalls mehrere Landmarken mit dazugehörigen räumlichen Relationen in den einzelnen Navigationsschritten benutzt werden. Die Vorverarbeitung sollte also auf einen Stand gebracht werden, in dem das Matching auch Instruktionsgraphen mit mehreren Regionen verarbeiten kann.

Die einfachste Lösung für das Problem, regionenspezifische Teilgraphmatches zu einem Gesamtmatch zusammenzufügen, ist, die Aufteilung in Regionenteilgraphen und sukzessive Abarbeitung ganz aufzugeben und einfach den gesamten Teilgraph, der das Ziel des Matching beschreibt, zu matchen. Wie man auch am Beispiel aus Abbildung 4.3.1 nachvollziehen kann, überschneiden sich die einzelnen Regionenteilgraphen für den Instruktionsgraphen oft stark. Dadurch muss viel Matchingarbeit für jeden einzelnen Regionenteilgraphen wiederholt werden und so wird der Vorteil, der durch die kleineren Graphen entsteht, zumindest teilweise wieder zunichte gemacht. Ebenfalls zu beachten ist, dass größere Instruktionsgraphen grundsätzlich zwar zu höheren Berechnungszeiten führen können, gleichzeitig aber auch mehr Informationen enthalten, die die Suche einschränken, und so weniger Matches als Ergebnis gefunden werden. Eine Aufteilung muss also nicht grundsätzlich von Vorteil sein. Insbesondere ist auch zu bedenken, dass durch den neu vorgeschlagenen

Algorithmus für das Teilgraphmatching die Größe der Instruktionsgraphen eine höhere Bedeutung erhält, als das für den alten Algorithmus der Fall ist, und so der Vorteil von kleineren Perzeptionsgraphen an Bedeutung verliert.

4.3.2 Vorverarbeitungsschritte bis zum Matchen

Für die Vorbereitung eines Matchingvorgangs, egal ob für einzelne Regionen oder den alle Regionen enthaltenden Teilgraphen, muss der vom Matching zu verarbeitende Teilgraph aus dem Gesamtinstruktionsgraphen bestimmt werden. Dies lässt sich über einfaches Graphflooding realisieren. Da die Beschreibung des Ziels eines Matchingvorgangs als Zusammenhangskomponente im Instruktionsgraphen vorliegt, genügt es, für den alle Regionen enthaltenden Teilgraph diese Zusammenhangskomponente ausgehend vom Ziel des Matchings zu bestimmen. Für die Regionenteilgraphen müssen alle für das aktuelle Ziel relevanten Regionen in dieser Zusammenhangskomponente bestimmt werden und dann ausgehend von diesen Regionen jeweils die Referenzknoten und durch die Region lokalisierten Knoten. Dabei ist allerdings zu beachten, dass die Zusammenhangskomponenten nicht nur die für das Ziel relevanten Informationen enthalten müssen. Wenn dieselbe Landmarke in der Routeninstrukion für mehrere Routenabschnitte benutzt wird, so gibt es für sie auch nur einen Knoten im Instruktionsgraphen. An diesem hängen jedoch die beschreibenden Teilgraphen für die Routenabschnitte, so dass in so einer Zusammenhangskomponente Informationen mit unterschiedlichen Voraussetzungen hinsichtlich zeitlicher Verankerung in der Routeninstruktion und der Position und Orientierung des Agenten vorliegen (vergleiche Abbildung 4.3.2). Es muss darauf geachtet werden, dass nur Instruktionsinformationen, die für das aktuelle Matching relevant sind, im extrahierten Teilgraphen bleiben. Dies kann gewährleistet werden, wenn für die Komplettierung der beschreibenden Teilgraphen räumliche Relationen nur in der Richtung von der Region zur Landmarke verfolgt werden und nicht umgekehrt.

Für die Weiterverarbeitung eines relevanten Teilgraphen, müssen auf dessen Basis der Perzeptionsgraph um Regionen erweitert werden. Für jede Region im zu matchenden Instruktionsteilgraphen werden potentielle Knotenmatches zu den Referenzknoten der Region gebildet. Ist die Ähnlichkeit eines Knotenmatches groß genug, wird in der Perzeption eine Region in derselben räumlichen Relation zum potentiellen Matchpartner des Referenzknoten hinzugefügt, falls diese noch nicht vorhanden ist. Kommen in der Beschreibung des aktuellen Ziels mehrere Regionen derselben räumlichen Relation vor, ist auch hier die Laufzeitverbesserung durch das Aufteilen in mehrere Regionenteilgraphen fragwürdig, da beim Gesamtteilgraphen nur eine Region hinzugefügt werden muss, während diese Region für die Regionenteilgraphen

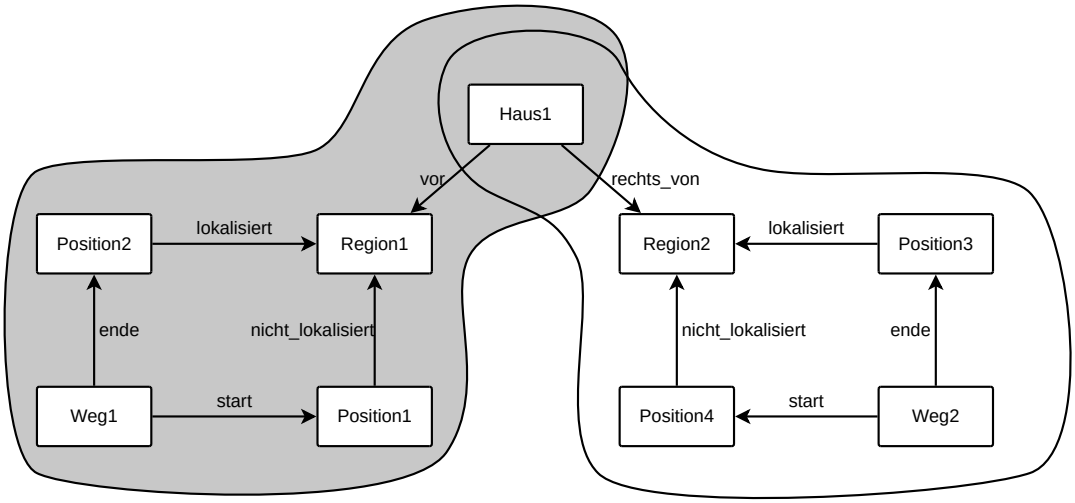


Abbildung 4.3.2: Zwei Beschreibungen von Routenabschnitten teilen sich dieselbe Landmarke *Haus1* und bilden deswegen eine Zusammenhangskomponente im Graphen. Beide Beschreibungen sind jedoch zeitlich und deswegen auch potentiell räumlich voneinander getrennt und dürfen nicht zusammen behandelt werden.

potentiell mehrfach hinzugefügt wird und der Größenunterschied, der den Vorteil der Aufteilung mit sich bringt, in diesem Fall auch für die Perzeptionsgraphen geringer wird.

Obwohl also begründete Zweifel bestehen, dass das Aufteilen in Regionenteilgraphen für alle Fälle eine Verbesserung mit sich bringt, kann das Verfahren erst abschließend bewertet werden, wenn ein Verfahren zum Zusammenführen der Teilmatches gefunden wurde.

4.3.3 Zusammenfassen von Regionenteilgraphen

Werden die Regionenteilgraphen einzeln gematcht, so entsteht für jeden Teilgraphen eine eigene Lösungsmenge an Matches. Um aus diesen Lösungsmengen Gesamtmatches zu erstellen, ist zunächst zu überlegen, wie zwei Teilgraphmatches sich zu einem zusammenfassen lassen.

Teilgraphmatches bestehen immer aus mehreren Komponenten. Zum einen sind Relationen- und Knotenmatches gespeichert. Zum anderen ist gespeichert, aus welchen beiden Graphen das Match entstanden ist. Dies ist notwendig, um die Ähnlichkeit, die sich aus den einzelnen Knoten- und Relationenmatches ergibt, durch die Größe der Graphen zu normieren. Wenn nun zwei Regionenteilgraphmatches zu einem Match kombiniert werden sollen, müssen auch diese Graphen wieder vereinigt werden, da sich die zusammenge-

fassten Relationen- und Knotenmatches auf den größeren Graphen beziehen. Dabei können sich die beiden Instruktions- oder Perzeptionsgraphen überlappen. Für diese Überlappungsregionen kann es in beiden Matches Knoten- und Graphmatches geben und diese sind dafür entscheidend, ob die beiden Teilgraphmatches zusammenpassen. Um Konsistenz des zusammengefassten Teilgraphmatches zu gewährleisten, müssen also diese Knoten- und Relationenmatches überprüft werden, ob in beiden zu kombinierenden Teilgraphmatches das gleiche Match auftaucht. Da die Bestimmung der Überlappungsregionen einigen Aufwand erfordert, ist es sinnvoller, einfach alle Matches zu prüfen, ob ein Knoten oder eine Relation denselben Matchpartner hat, wenn dieser oder diese in beiden Matches gematcht wurde. Dann sind die beiden Teilgraphmatches *zusammenfassbar*.

Definition 4.3.1
Zwei Teilgraphmatches M_1, M_2 sind zusammenfassbar*, wenn für alle Knotenmatches und Relationenmatches $(i_1, p_1) \in M_1$ gilt:*

$$\neg\exists (i_1, p_2) \in M_2 \text{ mit } p_2 \neq p_1$$
$$\neg\exists (i_2, p_1) \in M_2 \text{ mit } i_2 \neq i_1$$

Das Zusammenfassen selbst ist dann eine Reihe trivialer Vereinigungsoperationen über den beiden Mengen jeweils der Knotenmatches, Relationenmatches, Instruktionsknoten, Instruktionsrelationen, Perzeptionsknoten und Perzeptionsrelationen. Da aus der Zusammenfassung von zwei Teilgraphmatches wieder ein Teilgraphmatch entsteht, ist es problemlos möglich, auch mehrere Teilgraphmatches zu einem zusammenzufassen, indem man jedes weitere immer wieder mit dem Ergebnis weiter zusammenfasst. Wenn die Teilgraphmatches dabei paarweise zusammenfassbar sind, ist das resultierende Teilgraphmatch konsistent, wenn die einzelnen Teilgraphmatches vorher konsistent waren.

Mit dieser Vorüberlegung zur Zusammenfassung von Teilgraphmatches kann ein Algorithmus entwickelt werden, der alle zusammengefassten Teilgraphmatches aus den Matches der einzelnen Regionenteilgraphen erstellt. Für jeden Regionenteilgraphen R_k gibt es eine Menge $\textsc{Matches}_k$. Für jeden Regionenteilgraph muss genau ein Match zu einem vollständigen Gesamtmatch beigetragen haben, und so besteht ein vollständig zusammengefasstes Match für n Regionenteilgraphen aus einer Folge $m_1 m_2 m_3 \ldots m_n$ bei der m_k jeweils $\in \textsc{Matches}_k$ und alle einzelnen Matches paarweise zusammenfassbar sind. Es müssen also alle Kombinationen durchprobiert werden, wobei eine Prüfung auf Zusammenfassbarkeit als Abbruchkriterium benutzt werden kann. Die Prüfung aller Kombinationen ist nötig, da die Matches sich in

der Ähnlichkeit nicht unterscheiden müssen und so kein Kriterium für eine bevorzugte Behandlung nur eines Teils der Matches besteht.

function $fasseRegionsGraphenZusammen(\text{MATCHES}_1, \ldots \text{MATCHES}_n)$
ZMATCHES=MATCHES_1
for $k = 2$ to n **do**
  ZMATCHES′= $\emptyset$
  for all $m \in \text{MATCHES}_k$ **do**
    for all $zm \in$ ZMATCHES **do**
      if zm und m sind zusammenfassbar **then**
        ZMATCHES′=ZMATCHES′$\bigcup fasseZusammen(zm, m)$
      end if
    end for
  end for
  ZMATCHES=ZMATCHES′
end for
return ZMATCHES

Dieser Algorithmus berechnet alle konsistenten Teilgraphmatches, die aus allen Regionenteilgraphen zusammengefasst sind. Die Komplexität dafür liegt grundsätzlich bei $O(|\text{MATCHES}_1| \cdot |\text{MATCHES}_2| \ldots \cdot |\text{MATCHES}_n|)$. Es werden jedoch viele Kombinationen ausgeschlossen, da die paarweise Zusammenfassbarkeit nötig ist. Durch geringfügige Änderungen im Algorithmus, wenn ZMATCHES′ zu ZMATCHES hinzugefügt wird, anstatt es zu ersetzen und zusätzlich noch jedes m zu ZMATCHES′ hinzugefügt wird, können auch alle nicht vollständigen möglichen konsistenten, zusammengefassten Teilgraphmatches berechnet werden. Allerdings hat diese Variante des Algorithmus eine noch höhere Komplexität und damit potentiell längere Laufzeit.

4.3.4 Evaluation

Angesichts der hohen Komplexität, die im exponentiellen Bereich in Abhängigkeit von der Anzahl der Regionenteilgraphen liegt, erscheint die Aufteilung in Regionenteilgraphen zur Reduzierung der Laufzeit des Matchings nicht sinnvoll. Das Verfahren sollte gerade für hohe Zahlen an Regionen einen Vorteil bringen, da erst dann die Instruktionsteilgraphen wesentlich kleiner werden als der Gesamtteilgraph. Während des Teilgraphmatchings müssen für den zusammenhängenden Gesamtgraphen dieselben Kombinationen überprüft werden wie im Nachhinein beim Zusammenfassen. Es scheint, als würde dieser Komplexitätsfaktor nur aus dem Teilgraphmatching in das Zusammenfassen von Matches verlagert. Dies rechtfertigt den Mehraufwand, der durch

die Überschneidung der Teilgraphen entsteht, nicht, zumal im Teilgraphmatching Konflikte, die bei Aufteilung in Regionenteilgraphen erst nach dem Matchen in der Zusammenfassung auftauchen, durch heuristische Auswahl möglicherweise früh ausgenutzt werden können, um Berechnungsaufwand zu vermeiden.

Ein Vorteil, den das Aufteilen bieten könnte, ist, dass mit dem abgewandelten Zusammenfassungs-Algorithmus auch partielle Matches gefunden werden können, diese partiellen Matches jedoch aus gematchten Regionenteilgraphen bestehen und so durch die Region einen relevanten Informationsgehalt haben. Da die abgewandelte Form des Algorithmus jedoch noch höhere Komplexität hat, sollte das gesamte Vefahren höchstens als Alternative eingesetzt werden, falls mit anderen Ansätzen keine vollständigen Matches gefunden wurden.

5. FAZIT UND AUSBLICK

Ziel dieser Diplomarbeit war es, das Matching des Geometrischen Agenten auf Probleme zu überprüfen, die seine Anwendung für bestimmte Problemstellungen verhindern oder zumindest erschweren, und für diese Probleme so weit möglich Lösungen zu finden. Für die Analyse konnten drei Hauptkomponenten identifiziert werden, Vorverarbeitung, Teilgraphmatching und Knotenmatching, die einzeln untersucht wurden. Wie bei den meisten Verfahren, die kognitive Fähigkeiten simulieren, ist das größte Problem die Komplexität der eingesetzten Algorithmen und die daraus resultierende Laufzeit für große Problemstellungen. Dementsprechend taucht diese Schwierigkeit auch in allen drei Teilgebieten auf und war in der Vorverarbeitung für ein weiteres Problem verantwortlich.

5.1 Vorverarbeitung

Um die Laufzeit einzuschränken, wurde in der Vorverarbeitung nicht der gesamte Instruktionsausschnitt bearbeitet, der das Ziel des Matchings beschreibt. Stattdessen wurde versucht, den Graphen auf Regionenteilgraphen einzuschränken. Daraus ergab sich der Umstand, dass das Matching nur Instruktionsausschnitte mit einer einzigen Region verarbeiten konnte. Die Idee der Aufteilung in Teilgraphen und einzelnes Matchen dieser nach dem Divide-and-Conquer-Prinzip erschien zunächst sinnvoll, um möglichst geringe Laufzeit zu erreichen. Nach der Ausarbeitung eines Verfahrens zum Zusammenfügen der einzelnen Teilgraphmatches habe ich jedoch diesen Ansatz wieder verworfen, weil die Komplexität des Zusammenfügens den Vorteil der Behandlung kleinerer Teilgraphen aufwiegt. Eine Bestimmung des gesamten das Ziel des Matchings beschreibenden Teilgraphen und einmaliges Teilgraphmatchen halte ich für sinnvoller, da so Synergieeffekte, die durch die zusätzlichen Informationen der größeren Graphen entstehen, im Teilgraphmatching ausgenutzt werden können. Mehrere Regionen in dem zu untersuchenden Teilgraphen können aber von beiden vorgestellten Verfahren jetzt behandelt werden.

5.2 Knotenmatching

Beim Knotenmatching war das Problem der Komplexität der Algorithmen und die daraus resultierende Beschränkung auf kleine Problemstellungen besonders ausgeprägt. Es konnten nur Knoten mit relativ wenigen Attributen überhaupt adäquat behandelt werden. Dies ist darauf zurückzuführen, dass der verwendete Algorithmus zwar das gestellte NxM-Matchingproblem lösen konnte, eigentlich jedoch für eine Oberklasse dieses Spezialfalls, Maximum Subgraph Matching, verwendet wird, zu der auch das Teilgraphmatching gehört, welches NP-vollständig ist. Zusätzlich waren auch die im Algorithmus verwendeten Auswahlverfahren und Heuristiken ungeeignet. Die in Abschnitt 4.1.2 vorgestellten Verfahren lösen dagegen das spezifische Problem und kommen deswegen mit polynomiellem Zeitaufwand aus. Das MWBG-Matching benutzt das gleiche Optimalitätskriterium wie das alte Verfahren und ist deswegen ohne Zweifel ein vollständiger Ersatz. Das Stable-Marriage-Matching ist komplexitätstechnisch gesehen sogar noch besser, hat jedoch ein anderes Optimalitätskriterium und findet deswegen in einigen Fällen andere Zuordnungen der Prädikate der beiden zu matchenden Knoten. Da jedoch nur Ähnlichkeiten miteinander verglichen werden, die durch dasselbe Verfahren berechnet wurden, halte ich grundsätzlich das Stable-Marriage-Matching für am besten geeignet, das Knotenmatching im Geometrischen Agenten zu übernehmen. Durch Verwendung der neuen Matchingverfahren können jetzt auch Knoten mit wesentlich mehr Eigenschaften als vorher in den CRIL-Repräsentationen verwendet werden und, da die Ähnlichkeitsberechnung von Knoten ein oft verwendeter Unterschritt des Teilgraphmatching und der Vorverarbeitung ist, hat das Matching dadurch eine günstigere Laufzeit.

5.3 Graphmatching

Für das Graphmatching besteht die Schwierigkeit, dass das zu lösende Problem grundsätzlich NP-vollständig ist. Sowohl der von Helwich (2003) eingeführte Poole-Campbell-Algorithmus als auch der neu vorgeschlagene Graphflooding-Algorithmus, der Ähnlichkeit zu dem in Zhong et al. (2002) verwendeten Algorithmus aufweist, haben keine polynomielle Worst-Case-Komplexität und hängen in ihrer Laufzeit stark von eingesetzten Heuristiken und der Gutartigkeit der zu matchenden Graphen ab. Beide untersuchten Algorithmen bewerte ich als in etwa gleich geeignet, die benötigten Graphmatches zu finden. Aufgrund der unterschiedlichen Ansätze hat jedoch jeder Algorithmus seine eigenen Vor- und Nachteile. Der Poole-Campbell-Algorithmus selektiert über die Gesamtähnlichkeit der Teilgraphmatches dasjenige, für

das es sich lohnt es weiterzuverarbeiten, und findet so nur die besten Matches, während der Graphflooding-Algorithmus, blind für die Gesamtähnlichkeit, alle Teilgraphlösungen untersucht, die strukturell passend sind. Dies betrifft nur die Selektion eines Teilgraphmatches aus den vorhandenen. Der Graphflooding-Algorithmus hingegen muss nur aus einer durch die Struktur eingeschränkten Anzahl an Knotenmatches das nächste wählen und steuert an Hand des Instruktionsgraphen so zielstrebig auf die Lösungen zu, während der Poole-Campbell-Algorithmus immer aus allen Knotenmatches ohne Informationen über die gesuchten Graphen auswählt. Dies betrifft nur die Auswahl eines zu überprüfenden Knotenmatches für ein Teilgraphmatch.

Ein Algorithmus, der beide Vorteile verbindet, könnte also die Selektion von Graphmatches über die Ähnlichkeit aus dem Poole-Campbell-Algorithmus übernehmen und die herauszustreichenden Matches über Graphflooding bestimmen. Dazu müssten in den Teillösungen sowohl die Informationen über alle Knoten- und Relationenmatches, die für die Ähnlichkeitsberechnung nötig sind, als auch die aktuelle Position sowie Informationen über die Suchrichtung im dominanten Graphen abgespeichert werden. Diese Grundidee halte ich für einen Ansatz, der grundsätzlich bessere Laufzeit bietet als einer der beiden anderen Algorithmen, und deswegen wäre es interessant, ihn weiter zu verfolgen.

LITERATURVERZEICHNIS

Bittkowski, N. (2005). *Aktionsplanung und -steuerung unter Unsicherheit bei der Navigation eines Geometrischen Agenten mit Hilfe von Wegbeschreibungen.* Diplomarbeit, Universität Hamburg.

Bosch, T. (2004). *Bestimmung von Position und Sicht des imaginären Wanderers in Routenbeschreibungen.* Diplomarbeit, Universität Hamburg.

Burkard, R. E. (1999). Linear assignment problems and extensions. *Handbook of Combinatorial Optimization*, 4(1):221–300.

Gale, D. und Shapley, L. S. (1962). College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69, No. 1:9–15.

Habel, C. (1988). Prozedurale Aspekte der Wegplanung und Wegbeschreibung. *Sprache in Mensch und Computer*, 107–133.

Helwich, J. (2003). *Graphenbasierte Navigation eines Geometrischen Agenten: Integration von Perzeption und Instruktion.* Diplomarbeit, Universität Hamburg.

Karp, R. M. (1980). An Algorithm to Solve the m x n Assignment Problem in Expected Time O(mn log n)*. *Networks*, 10:143–162.

Klein, W. (1979). Wegauskünfte. *Zeitschrift für Literaturwissenschaft und Linguistik*, 33:9–57.

Kuhn, H. W. (2005). The Hungarian method for the assignment problem. *Naval Research Logistics*, 52(1):7–21.
URL `http://dx.doi.org/10.1002/nav.20053`

Lawler, E. L. (1963). The Quadratic Assignment Problem. *Management Science*, 9(4):586–599.
URL `http://www.jstor.org/stable/2627364`

Lewis, J. M. (1978). On the complexity of the Maximum Subgraph Problem. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, 265–274. ACM, San Diego, California, United States.

Maass, W. (1993). A cognitive model for the process of multimodal, incremental route descriptions. In Frank, A. U. und Campari, I. (Hg.), *Spatial Information Theory A Theoretical Basis for GIS*, 1–13. Springer, Berlin/Heidelberg.
URL http://dx.doi.org/10.1007/3-540-57207-4_1

Marie, A. und Gal, A. (2007). On the Stable Marriage of Maximum Weight Royal Couples. In *Proceedings of AAAI Workshop on Information Integration on the Web (II-Web'07), Vancouver, BC, Canada*.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.
URL http://www.jstor.org/stable/2098689

Poole, J. und Campbell, J. A. (1995). A novel algorithm for matching conceptual and related graphs. In Ellis, G.; Levinson, R.; Rich, W. und Sowa, J. F. (Hg.), *Conceptual Structures: Applications, Implementation and Theory*, 293–307. Springer, Berlin.
URL http://dx.doi.org/10.1007/3-540-60161-9_45

Russell, S. und Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2. Aufl.

Skubic, M.; Blisard, S.; Bailey, C.; Adams, J. und Matsakis, P. (2004). Qualitative analysis of sketched route maps: translating a sketch into linguistic descriptions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(2):1275–1282.

Tschander, L.; Schmidtke, H.; Eschenbach, C.; Habel, C. und Kulik, L. (2003). A Geometric Agent Following Route Instructions. In Freska, C.; Bauer, W.; Habel, C. und Wender, K. F. (Hg.), *Spatial Cognition III*, 89–111. Springer, Berlin/Heidelberg.

Tversky, B. und Lee, P. (1999). Pictorial and Verbal Tools for Conveying Routes. In Freksa, C. und Mark, D. (Hg.), *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*, 752. Springer, Berlin/Heidelberg.
URL http://dx.doi.org/10.1007/3-540-48384-5_4

Zhong, J.; Zhu, H.; Li, J. und Yu, Y. (2002). Conceptual Graph Matching for Semantic Search. In Priss, U. (Hg.), *Conceptual Structures: Integration and Interfaces*, 92–106. Springer, Berlin.
URL http://dx.doi.org/10.1007/3-540-45483-7_8

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.

Ich bin mit einer Einstellung in den Bestand der Bibliothek des Departments Informatik einverstanden.

Hamburg, den 26. November 2009

Lars Müllerchen