TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik

Diplomarbeit

zum Thema

Erweiterung des RNN-Modelles um SHRUTI-Konzepte: Vom aussagenlogischen zum Schließen über prädikatenlogischen Programmen.

von
Jörg Wunderlich
geboren am 30.03.1973 in Dresden

eingereicht am 30. September 1998

Betreuer und verantwortlicher Hochschullehrer: Prof. Dr. Steffen Hölldobler

Inhaltsverzeichnis

Vorwort

Nach mehreren hundert Stunden Arbeit wollte ich es mir nicht nehmen lassen, am Anfang der Arbeit wenigstens ein kleines Dankeschön an alle zu richten, die mich während meines Studiums und insbesondere während der Diplomarbeit unterstützt haben und mir stets mit Rat und Tat zur Seite standen.

Mein erster Dank geht an meine Eltern, die mich während meiner gesamten Studienzeit in allen Belangen liebevoll unterstützten und mir viele Lasten abnahmen. Ohne sie hätte ich das Studium wahrscheinlich nicht in dieser Form vollbringen können.

Die Anregungen zum Thema dieser Arbeit verdanke ich Steffen Hölldobler und Yvonne Kalinke. Sie weckten mein Interesse dafür, konnektionistische Modelle und die symbolische Verarbeitung von Wissen zu verbinden und gaben mir wertvolle Ratschläge während der Durchführung meiner Arbeit.

Für viele fruchtbare Diskussionen und Anregungen, sowie die mehrfache ausführliche Lektüre des Manuskriptes bin ich besonders Enno Sandner und Frank Wehner zu Dank verpflichtet. Weitere Unterstützung erhielt ich auch von André Standke, Yves Martin und Helko Lehmann.

Zum Schluß möchte ich all denen danken, die mich im Studium eine großartige Zeit erleben ließen, sei es nun, daß sie gemütlich mit mir beim Bier gesessen und gelacht haben, mir beim Sport tolle Trainingskameraden waren oder einfach da waren, wenn ich sie brauchte, kurz und gut, all meinen Freunden.

1. Einleitung

Das grundlegende Ziel der Wissensverarbeitung ist von jeher, neues Wissen aus einem vorhandenen Erfahrungsschatz automatisiert folgern zu können und dies mindestens so schnell zu tun, wie das der Mensch selbst im Alltagsschließen realisiert. Diese Bestrebungen scheitern jedoch immer wieder an der Mächtigkeit prädikatenlogischer Problemstellungen. Zusammenhänge, die der Mensch in kürzester Zeit lösen kann, verbrauchen mit maschinellen Methoden unverhältnismäßig viel mehr Zeit oder Platz. Und auch das menschliche Gehirn hat nur begrenzte Ressourcen. Dieses Paradoxon der Künstlichen Intelligenz beschäftigt eine Vielzahl von Forschern auf der ganzen Welt. Doch worin liegen seine Ursachen? Ist der Grund dafür, daß wir im menschlichen Gehirn nicht mit einem so künstlichen Instrument wie der Prädikatenlogik arbeiten, oder kann man die Probleme, die wirklich effektiv lösbar sind, auf eine Logik reduzieren, die auch maschinell mit sinnvollem Aufwand realisiert werden könnte?

Die Anforderungen an eine Inferenzmethode beschreibt Bibel ([Bib91]) vor diesem Hintergrund in der Adäquatheitseigenschaft. In seinem Verständnis ist eine Schlußfolgerung adäquat, wenn die Zeit zum Lösen des Problems tatsächlich abhängig ist von der Schwere des Problems. Je leichter das Problem, desto geringer der nötige Zeitaufwand. In dem Zusammenhang stellt sich sofort die Frage, wie die Schwere eines Problems zu beurteilen ist. Aus Sicht der symbolischen Logiken kann man sie vielleicht an der Verkettung des Regelwissens messen, das zur Folgerung führt, aber ob das der tatsächlichen menschlichen Skala entspricht ist ungeklärt. Für das Erreichen wirklich adäquater Folgerungsmethoden gibt es deshalb viele Ansätze, mit konnektionistischen Modellen nach dem Vorbild Gehirn symbolische Zusammenhänge darzustellen und zu verarbeiten. Auch die Komplexitätstheorie (siehe [GR88],[KR90]) liefert dafür entsprechende Beurteilungskriterien. Ein paralleler Algorithmus wird als optimal bezeichnet, wenn das Produkt aus Anzahl verwendeter Prozessoren p und paralleler Zeit t linear zur Schwere des Problems ist. Optimalität kann aber auch bedeuten, daß $p \times t$ gleich der Berechnungszeit des schnellsten bekannten sequentiellen Algorithmus ist. Ein effizienter paralleler Algorithmus benötigt polylogarithmische Zeit bei polynomialer Anzahl Prozessoren.

Das RNN-Modell von Hölldobler und Kalinke in [HK94] berechnet den Operator T_P von aussagenlogischen Programmen. Für die Klasse der akzeptablen Programme weisen die Autoren nach, daß die Berechnung einen Fixpunkt hat, der dem kleinsten Modell des Programmes entspricht. Das RNN-Modell gibt damit aber nicht nur einen Hinweis

über die deklarative Semantik eines Programmes, sondern ist gleichzeitig ein Instrument zum Schließen über einer Wissensbasis, wenn auch nur auf einer syntaktisch sehr beschränkten Logik. Das Modell selbst ist ein rekurrentes neuronales Netz mit linearem Platzaufwand bezüglich der Größe der Wissensbasis und parallel linearem Zeitaufwand bezüglich der Länge der Inferenzketten. Es realisiert massiv parallele Berechnungen und ist ein optimaler paralleler Algorithmus für die Realisierung von T_P.
Leider ließen sich bisher im rekurrenten neuronalen Netz lediglich aussagenlogische Zusammenhänge mit linearem Aufwand realisieren. Das liegt vor allem daran, daß die Darstellung von Funktionssymbolen eine unendliche Menge möglicher Terme erzeugt, die es gilt, mit endlicher Anzahl Einheiten darzustellen. In [KS95], [KS96] und [HKS98] untersuchen Hölldobler, Kalinke und Störr die Möglichkeit, prädikatenlogische Zusammenhänge mit reellwertigen Netzeinheiten darzustellen. Sie konnten nachweisen, daß für jedes rekurrente prädikatenlogische Programm ein 3-schichtiges RNN-Netz existiert, welches die Semantik des Programmes approximiert, aber es konnten keine Hinweise auf die Konstruktion eines solchen Netzes gefunden werden. Auch die Verwendung von konnektionistischen Termrepräsentationen, die in [Kal97] diskutiert wird, brachte keine zufriedenstellenden Ergebnisse. Für eine Erweiterung des RNN-Modelles auf Prädikatenlogik werden deshalb auch Konzepte anderer neuronaler Strukturen in Betracht gezogen.

Das System ROBIN wurde von Lange und Dyer in [LD89] vorgestellt. Es basiert auf der Bearbeitung von Frames und kodiert Bindungen mittels Signaturen. Die Propagierung von Aktivierungen zwischen Netzeinheiten erfolgt dabei auf 2 getrennten Ebenen.
Sun ([Sun94]) entwickelte ein hybrides System aus einem konnektionistischen und einem symbolischen Teil. Seine Architektur erlaubt damit sowohl die Anwendung formaler Regeln, als auch die Behandlung von Ähnlichkeiten.
Einen Überblick über verschiedene Architekturen, die neuronale und symbolische Berechnungen kombinieren, gibt [SB95].

Ein besonders interessantes Berechnungsmodell entwickelten Shastri und Ajjanagadde ([SA93]) mit dem SHRUTI-System. Sie versuchen damit, den Bezug zu der psychologisch begründeten Klasse des reflexiven Schließens herzustellen. Die für das RNN-Modell wichtigen Konzepte sind dabei eine eingeschränkte Prädikatenlogik, die unter linearem Zeitaufwand lösbar ist und die dynamische Bindung von Argumenten. Den Weg der Einschränkung der Logik begründen die Autoren mit dem reflexiven Schließen beim Menschen. Reflexives Schließen ist eingeschränktes Schließen. Für die Lösung komplizierterer Probleme benötigt auch der Mensch mehr Zeit und Hilfsmittel. Kann man die Annahme bestätigen, daß der Mensch wirklich nur eine eingeschränkte Klasse von Problemen so reflexartig lösen kann, und entspricht diese Klasse der von SHRUTI beschriebenen Logik, dann existiert das Paradoxon der Künstlichen Intelligenz möglicherweise gar nicht. Diese Frage stellen sich auch Hölldobler und Strohmaier ([BH93],[Str97]) und zeigen, daß SHRUTI eigentlich nur eine Anwendung linearer Reduktionstechniken ist, die teilweise parallel durchgeführt werden können. Eine Inferenz in SHRUTI ist deshalb nur proportional zur Länge der Inferenzkette und deshalb möglicherweise adäquat.

Die vorliegende Arbeit soll die Möglichkeiten diskutieren, die Konzepte von SHRUTI zur Darstellung von Prädikatenlogik auf das RNN-Modell anzupassen. Dazu wird das Vorwärtsschließen in SHRUTI ausführlich dargestellt. Die Kodierung von Wissen wird entwickelt und die Arbeit deckt dabei erstaunliche Unterschiede zum Rückwärtsschließen auf, die von den Autoren Shastri und Ajjanagadde so nicht erwähnt werden.

Nach der verbalen Betrachtung wird SHRUTI im Zusammenhang mit der Konnektionsmethode untersucht. Es wird bewiesen, daß auch das Vorwärtsschließen identisch zur Anwendung von Reduktionen ist. Dieses Ergebnis ist der Ausgangspunkt für den in der Arbeit definierten BUR-Kalkül, der auf einer Hornklausellogik basiert und mit weit weniger Annahmen auskommt als SHRUTI. Seine Probleme werden ebenfalls nur durch Anwendung von Reduktionen gelöst. Sie können deshalb mit linearer Zeit- und Platzkomplexität und deshalb adäquat bearbeitet werden. Für eine Wissensbasis mit nur einstelligen Prädikatssymbolen ist der Kalkül korrekt und vollständig. Für den allgemeinen Fall mehrstelliger Prädikate beweist die Arbeit, daß BUR vollständig, aber nicht notwendigerweise korrekt ist.
Mit diesen Eigenschaften läßt sich ein gewisser Bezug zu Methoden des approximativen Schließen ([SC95], [Mas96] und [Mas97]) ziehen. Diese Inferenzkalküle sind ebenfalls vollständig, aber nicht korrekt. Der Unterschied zum BUR-Kalkül besteht darin, nicht die Logik von vornherein einzuschränken, sondern bestimmte Teile zu vernachlässigen und ungefähre, also approximative Schlüsse mit linearem Aufwand durchzuführen. Außerdem ist der Grad der Vernachlässigung wählbar, so daß eine beliebige Annäherung an die korrekte Inferenz möglich ist.
Da dies in BUR nicht realisiert wird, stellt die Arbeit ein Theorem auf, welches für eine Klasse von Inferenzen garantieren kann, daß die im Kalkül abgeleiteten Formeln bis zu einem bestimmten Inferenzschritt tatsächlich logische Konsequenzen und damit korrekte Folgerungen im Sinne der Prädikatenlogik sind.

Das neue RNN-Modell für prädikatenlogische Programme wird als massiv paralleles Berechnungsmodell für den BUR-Kalkül entwickelt. Damit erfolgt eine strikte Trennung von theoretischem Kalkül und Implementierung. Weil das Modell eine direkte Kodierung der Inferenzrelation von BUR ist, erfüllt es auch die Anforderungen an lineare Platz- und Zeitkomplexität. Im Vergleich zu SHRUTI benötigt RNN nur zwei verschiedene Netzeinheiten und hat einen wesentlich einfacheren Aufbau. Die Arbeit zeigt außerdem am RNN-Modell ein Beispiel aus dem Alltagsschließen, für das mit nur einem Inferenzprozeß genau die Menge aller korrekten Folgerungen ermittelt werden kann.

In Kapitel 2 der Arbeit werden die Grundlagen des Schließens beleuchtet, wobei die Kenntniss von den Sprachen der Aussagenlogik und der Prädikatenlogik vorrausgesetzt wird. Im darauf folgenden Kapitel wird das RNN-Modell für aussagenlogische Programme betrachtet. Außerdem erläutert 3 auch die Versuche der Erweiterung auf prädikatenlogisches Wissen. Die Prinzipien von SHRUTI zeigt Kapitel 4 und die Anwendung linearer Reduktionen und die Definition des BUR-Kalküls erfolgt in 5. Das erweiterte dynamische RNN-Modell wird in Kapitel 6 beschrieben.

2. Grundlagen des Schließens

In diesem Kapitel werden die grundlegenden Definitionen zum Schließen betrachtet. Dabei werden zuerst logische Programme beleuchtet, die meist als Wissensbasis für Inferenzvorgänge verwendet werden. Im 2. Abschnitt diskutieren wir Formen des Schließens und die Eigenschaften entsprechender Beweismethoden. Vorraussetzung für das Verständnis der gesamten Arbeit ist die Kenntnis der Aussagenlogik, der Prädikatenlogik und der damit verbundenen Operatoren. Eine ausführliche Beschreibung dieser Sprachen ist bei Lloyd ([Llo87]) oder in [Rei97] zu finden.

2.1 Logische Programme

Die nachfolgenden Definitionen sind auf die Notation aus [Llo87] zurückzuführen.

Definition 2.1.1 *Eine* Programmklausel *ist eine Klausel der Form*

$$A \leftarrow L_1, \ldots, L_m$$

wobei A ein Atom ist und $L_1, \ldots, L_m$ Literale. Als Kopf der Klausel *bezeichnen wir A. Den* Körper der Klausel *bildet $L_1, \ldots, L_m$.*

Definition 2.1.2 *Eine* definite Programmklausel *ist eine Klausel der Form*

$$A \leftarrow B_1, \ldots, B_m$$

wobei $A, B_1, \ldots, B_m$ Atome sind. Eine Programmklausel der Form $A \leftarrow$ oder A bezeichnen wir als Fakt. *Ein Fakt ist gleichzeitig auch eine definite Programmklausel.*

Definition 2.1.3 *Ein* normales Programm *ist eine endliche Menge von Programmklauseln.*

Definition 2.1.4 *Ein* definites Programm *ist eine endliche Menge von definiten Programmklauseln.*

Bemerkung: Die Menge der definiten logischen Programme ist offensichtlich eine Teilmenge der normalen logischen Programme. Wir wollen deshalb zur Abkürzung der Schreibweise normale logische Programme als Programme und definite logische Programme als definite Programme bezeichnen.

Definition 2.1.5 *Das* Herbrand-Universum U_P *für ein Programm P ist die Menge aller Grundterme, die mit den Konstanten und Funktionssymbolen von P gebildet werden können. Falls in P keine Konstanten vorkommen, wird eine beliebige Konstante definiert, um Grundterme formen zu können.*

Definition 2.1.6 *Die* Herbrand-Basis B_P *für ein Programm P ist die Menge aller atomaren Grundformeln $p(t_1, \ldots, t_n)$, die mit den n-stelligen Prädikatszeichen p in P und den Termen t_1, …, t_n des Herbrand-Universums U_P gebildet werden können.*

Definition 2.1.7 *Eine* Herbrand-Interpretation I *für ein Programm P ist eine Abbildung $I : B_P \rightarrow \{0, 1\}$ der Elemente der Herbrand-Basis auf die Wahrheitswerte 0 für falsch und 1 für wahr.*

Bemerkung: Herbrand-Interpretationen werden häufig als Mengen repräsentiert. In einer solchen Mengenschreibweise gilt für alle Elemente einer Herbrand-Interpretation I der Wahrheitswert wahr. Allen atomaren Grundformeln von P, die nicht in I enthalten sind, wird der Wahrheitswert falsch zugeordnet.

Für die Darstellung logischer Programme soll in dieser Arbeit folgende Notation verwendet werden:

- Prädikatssymbole werden dargestellt durch p, q, r
- Variablen werden dargestellt durch X, Y, Z
- Konstanten werden dargestellt durch a, b, c
- Funktionssymbole werden dargestellt durch f, g, h
- prädikatenlogische Formeln ohne Quantoren sind allabgeschlossen

Beispiel 2.1.1 *Gegeben sei folgendes logische Programm P:*

$$\begin{aligned} p(f(a)) &\leftarrow \\ q(X, a) &\leftarrow p(X) \\ q(X, Y) &\leftarrow q(Y, X) \end{aligned}$$

Die erste Formel von P stellt einen Fakt dar. Die anderen Formeln sind ebenfalls definite Programmklauseln. P ist folglich ein definites logisches Programm. Für das Herbrand-Universum U_P und die Herbrand-Basis B_P des Programmes ergibt sich

$$\begin{aligned} U_P &= \{a, f(a), f(f(a)), f(f(f(a))), \ldots\} \\ B_P &= \{p(a), q(a, a), p(f(a)), q(a, f(a)), q(f(a), a), q(f(a), f(a)), p(f(f(a))), \ldots\} \end{aligned}$$

Man erkennt, daß beide Mengen abzählbar unendlich viele Elemente enthalten. Verallgemeinert kann man sagen, daß alle prädikatenlogischen Programme mit mindestens einem einstelligen Funktionssymbol und einem n-stelligen Prädikatszeichen ($n \geq 1$) ein abzählbar unendliches Herbrand-Universum und eine abzählbar unendliche Herbrand-Basis haben. In der Aussagenlogik existieren nur Formeln und keine Terme. Ein endliches aussagenlogisches Programm hat demzufolge als Herbrand-Universum die leere Menge $\emptyset$ und eine endliche Herbrand-Basis.

Definition 2.1.8 *Sei I eine Herbrand-Interpretation und sei F eine abgeschlossene Formel. Dann ist I* Modell *von F (Notation: $I \models F$), wenn F bezüglich I den Wahrheitswert wahr hat.*

Definition 2.1.9 *Sei S eine Menge abgeschlossener Formeln und sei I eine Herbrand-Interpretation. I ist* Modell *von S ($I \models S$), wenn I Modell für jede Formel in S ist. I wird bezeichnet als* kleinstes Modell *von S wenn es keine Herbrand-Interpretation I' gibt, für die $I' \models S$ und $I' \subset I$ gilt.*

Definition 2.1.10 *Sei S eine Menge abgeschlossener Formeln und sei F eine abgeschlossene Formel. Dann ist F* logische Konsequenz *von S (Notation: $S \models F$), wenn für jede Herbrand-Interpretation I gilt:*

$$I \models S \Rightarrow I \models F$$

Definition 2.1.11 *Sei P ein normales Programm. Die Abbildung $T_P : 2^{B_P} \rightarrow 2^{B_P}$ ist wie folgt definiert. Sei I eine Herbrand-Interpretation, dann gilt $T_P(I) = \{A \in B_P : A \leftarrow L_1, \ldots, L_n$ ist eine Grundinstanz einer Klausel in P und $I \models \{L_1, \ldots, L_n\}\}$.*

Definition 2.1.12 *[Apt97] Eine* Stufenabbildung *für ein Programm P ist eine Funktion $|\ | : B_P \rightarrow \mathbb{N}$ von Grundatomen auf natürliche Zahlen. Für $A \in B_P$ gilt, $|A|$ ist die* Stufe *von A.*

Definition 2.1.13 *[Apt97] Sei P ein Programm, $|\ |$ eine Stufenabbildung für und I eine Interpretation von P.*

- *Eine Klausel von P wird genannt* akzeptabel in Bezug auf $|\ |$ und I, *wenn I ihr Modell ist und für jede Grundinstanz $A \leftarrow \mathbf{A}, L, \mathbf{L}$ von dieser Klausel mit $I \models \mathbf{A}$ gilt*

  $$|A| > |L|.$$

  In anderen Worten, für jede Grundinstanz $A \leftarrow L_1, \ldots, L_n$ der Klausel gilt

  $$|A| > |L_i| \text{ für } i \in [1, \bar{n}],$$

  wobei

  $$\bar{n} = \min(\{n\} \cup \{i \in [1, n] \mid I \not\models L_i\}).$$

- *Ein Programm P wird genannt* akzeptabel in Bezug auf | | und I, *wenn es alle seine Klauseln sind. P wird* akzeptabel *genannt, wenn es akzeptabel in Bezug auf eine beliebige Stufenabbildung und eine Interpretation von P ist.*

2.2 Inferenz und Kalkül

Was man allgemein unter den Begriffen Schließen oder Inferenz versteht, beschreibt eine Vielzahl verschiedener Formen des Erwerbs von Wissen. Drei der wichtigsten dieser Formen sind Abduktion, Induktion und Deduktion. Für die Wissensverarbeitung spielt diese Unterscheidung eine besonders große Rolle, weil die Automatisierung solcher Prozesse, die von jeher ihr fundamentales Ziel war, offensichtlich unterschiedliche Folgerungs- und Beweismethoden nach sich zieht.
Abduktionsverfahren sind keine logisch zwingenden Schlußverfahren. Sie versuchen, von Beobachtungen auf Ursachen zu schließen, die diese Beobachtungen möglicherweise hervorrufen könnten. Induktionen sind generalisierende Inferenzmethoden. Aus konkreten Beispielen wird versucht, allgemeingültige Zusammenhänge zu finden. Auch Induktionsmethoden sind nicht logisch zwingend. Die Deduktion beschäftigt sich in erster Linie mit logischen Beweisen. Dabei wird spezielles Wissen geschlossen, das implizit in einer Wissensbasis bereits vorhanden ist. Eine Einführung in die Teilgebiete der Inferenz geben [BHS93] und [Gö95].
Im Kontext dieser Arbeit benutzen wir den Begriff des Schließens als Beschreibung von Deduktionsverfahren, die wir wiederum in Vorwärtsschließen und Rückwärtsschließen unterteilen.

Als *Vorwärtsschließen* wird eine Inferenzmethode bezeichnet, die neue Sätze bildet, welche aus einer existierenden Wissensbasis gefolgert werden können. Ausgehend von Faktenwissen wird dabei auf synthetischem Weg durch Kombinieren von Formeln neues Wissen ermittelt.
Als *Rückwärtsschließen* wird eine analytisch vorgehende Inferenzmethode bezeichnet. Ausgehend von einem gegebenen Satz wird eine Menge von Formeln der Wissensbasis gesucht, die die Gültigkeit des Satzes auf existierende Fakten zurückführt. Kann eine solche Ableitung gefunden werden, wird aus der Wissensbasis auf den Satz geschlossen.

Logikbasierte Methoden zum automatisierten Schließen über Wissensbasen werden als Kalküle bezeichnet. Sie sind die eigentlichen Werkzeuge zum Problemlösen. Dabei besteht das Ziel eines Kalküls stets darin, die Gültigkeit bestimmter Formeln aufgrund des existierenden Wissens zu zeigen. Ein Kalkül sollte damit in der Lage sein, Formeln zu beweisen. Kalküle können sowohl vorwärtschließende, als auch rückwärtsschließende Inferenzmethoden repräsentieren.

Definition 2.2.1 *Ein* logischer Kalkül K *besteht aus*

- *einem Alphabet* $\mathbf{A}_K$*,*

- *einer Sprache aus den Elementen von* $\mathbf{A}_K$*, die eine Menge* $\mathbf{F}_K$ *von Formeln beschreibt,*
- *einer Menge* $\mathbf{X}_K$ *von Axiomen mit* $\mathbf{X}_K \subseteq \mathbf{F}_K$ *und*
- *einer Inferenzrelation* $\vdash_K$ *mit* $\mathbf{F}_K \times \mathbf{F}_K$*.*

Kalküle sind meist auf mehrere Sprachen anwendbar. Das hängt auch damit zusammen, daß ein Kalkül für Prädikatenlogik natürlich auch auf Aussagenlogik anwendbar ist, weil die eine Untermenge der Prädikatenlogik darstellt. Wichtig bei der Beurteilung der Sprache bzw. Logik ist die Entscheidbarkeit derselben. Die Entscheidbarkeit gibt an, ob für eine beliebige Formel der Logik in endlicher Zeit ein Beweis oder ein Gegenbeweis für ihre Gültigkeit gefunden werden kann.
Die Aussagenlogik ist eine entscheidbare Logik. Sie repräsentiert stets eine endliche Menge von atomaren Formeln. Durch Prüfen der Gültigkeit dieser Atome kann damit stets mit endlichem Aufwand die Gültigkeit einer beliebigen Formel entschieden werden.
In der Prädikatenlogik ist die Herbrand-Basis im allgemeinen eine abzählbar unendliche Menge. Ein Kalkül kann dann zwar in endlicher Zeit feststellen, ob eine Formel gültig ist, aber bei ungültigen Formeln kann der Algorithmus ewig laufen. Die Prädikatenlogik wird deshalb als semi-entscheidbar bezeichnet. Eine endliche Herbrand-Basis impliziert Entscheidbarkeit, weil die Allgemeingültigkeit jeder Formel durch Ermitteln ihres Wahrheitswertes für alle Herbrand-Interpretationen mit endlichem Aufwand bewiesen oder widerlegt werden kann. Insbesondere ist deshalb eine Prädikatenlogik ohne ein- und mehrstellige Funktionszeichen entscheidbar.

Ob die Inferenzen eines Kalküls Beweise darstellen, wird bestimmt durch das Verhältnis zwischen der Ableitbarkeit im Kalkül und der logischen Konsequenz. Eine Formel wird als ableitbar bezeichnet, wenn sie zum gegebenen Wissen in Relation bezüglich des Inferenzoperators $\vdash_K$ steht. Verbal bedeutet dies, daß die Formel im Kalkül geschlossen werden kann. Die Ableitbarkeit bezeichnet damit eine Art Inferenz auf der Ebene des Kalküls, die aber nicht notwendigerweise auch eine gültige Inferenz im Sinne der Logik sein muß. Die Menge der logischen Konsequenzen beschreibt genau die Menge aller Formeln, die nach den Regeln der Logik geschlossen werden dürfen bzw. unter dem gegebenen Wissen gültig sind. Damit ist $F \models G$ genau dann wahr, wenn alle Modelle für F auch Modelle für G sind bzw. wenn $F \rightarrow G$ allgemeingültig ist.
Der ideale Fall einer Inferenzmethode wäre, wenn die Menge der ableitbaren Formeln genau gleich der Menge der logischen Konsequenzen ist. Über dieses Ziel definieren sich die wichtigsten Eigenschaften eines Kalküls, die Korrektheit und die Vollständigkeit.

Definition 2.2.2 *Seien* F *und* G *beliebige Formeln der Sprache eines logischen Kalküls* K*. Der logische Kalkül* K *ist* korrekt *bezüglich* $\models$*, wenn gilt:*

$$F \vdash_K G \Rightarrow F \models G$$

Der logische Kalkül K *ist* vollständig *bezüglich* $\models$*, wenn gilt:*

$$F \models G \Rightarrow F \vdash_K G$$

3. Das RNN-Modell

Das RNN-Modell ist ein rekurrentes neuronales Netz, welches die Semantik eines logischen Programmes in Form des Operators T_P darstellt. Das Kapitel stellt das Modell für die Aussagenlogik vor und untersucht verschiedene Ansätze, es auf prädikatenlogisches Wissen zu erweitern. Dabei werden die wichtigsten Probleme dieses Unterfangens umrissen.
Die Prinzipien künstlicher neuronaler Netze sollen hier nicht erläutert werden. Entsprechende Grundlagen dazu sind in [RN95] zu finden. Eine umfangreiche Darstellung der Theorie neuronaler Netze gibt der Verweis auf [HKP91].

3.1 Das RNN-Modell für Aussagenlogische Programme

In [Kal94] stellt Kalinke das RNN-Modell für aussagenlogische Programme als massiv paralleles Berechnungsmodell vor. Dabei wendet sie metrische Methoden zur Berechnung der Semantik und das Wissen über rekurrente neuronale Netze als universelle Approximatoren an. Die nachfolgenden Zusammenhänge sind dieser Arbeit und [HK94] entnommen.

Für die Darstellung der Semantik eines Programmes nutzt man den in 2.1.11 definierten Operator T_P. Dieser ist eine Abbildung auf der Menge von Herbrand-Interpretationen. Ein konnektionistisches Modell zur Berechnung des Operators benötigt deshalb eine Darstellung für Interpretationen in der Eingabe- und der Ausgabeschicht. Dieses zentrale Problem der Kodierung von Interpretationen beeinflußt in großem Maße die Implementierung des vorhandenen Wissens. Für aussagenlogische Programme ist die Herbrand-Basis stets eine endliche Menge. Eine Interpretation als Abbildung der Elemente von B_P auf 0 und 1 läßt sich deshalb einfacherweise dadurch repräsentieren, daß man für jedes Element von B_P eine binäre Netzeinheit erstellt. Deren Ausgabe gibt den Wahrheitswert der aussagenlogischen Variable und damit ihre Zugehörigkeit zur Interpretation an. Ein Atom A ist genau dann in der Interpretation I enthalten, wenn dessen korrespondierende Netzeinheit eine Ausgabe gleich 1 hat. Nach diesem Prinzip arbeiten die äußeren Schichten des vorwärtsgerichteten neuronalen Netzes zur Berechnung von T_P. Kalinke zeigt in ihrer Arbeit, daß ein solches Netz jedoch mindestens eine weitere Schicht benötigt, um die gewünschte Funktion berechnen zu können. Auf der Basis dieser Ergebnisse entwickelt sie einen Konstruktionsalgorithmus, der ein 3-schichtiges neuronales

Netz konstruiert, welches genau die Funktion T_P für ein gegebenes aussagenlogisches Programm P realisiert.

vereinfachter Konstruktionsalgorithmus aus [Kal94]:

- Das Netz besteht aus einer Eingabeschicht, einer verdeckten Schicht und einer Ausgabeschicht.
- Erstelle für jedes Atom $A \in B_P$ jeweils eine Einheit A in der Eingabe- und Ausgabeschicht des Netzes. Die Werte dieser Einheiten repräsentieren den Wahrheitswert des Atoms und die gesamte Schicht entsprechend eine Interpretation I aller Atome. Setze den Schwellwert der Ausgabeeinheiten auf $\theta = 0.5$.
- Erstelle für jede Klausel K_j in P eine Einheit j in der internen Schicht des Netzes.
- Sei K_j eine Klausel in P der Form $A_j \leftarrow L_{j1}, \ldots, L_{jm_j}$, so existiert eine Verbindung zwischen jeder Einheit L_{jl} $(1 \leq l \leq m_j)$ und der Einheit j in der internen Schicht. Ist L_{ji} ein positives Literal, so wichte die Kante mit 1. Für negative Literale L_{ji} wird die Kante mit dem Gewicht -1 bewertet.
- Die Einheit j in der internen Schicht erhält den Schwellwert θ_j mit $(k-1) < \theta_j < k$, wobei k die Anzahl der in der Einheit endenden Kanten mit positivem Gewicht ist. Damit gilt insbesondere für eine Klausel K_j in P der Form $A_j \leftarrow$ (K_j ist ein Fakt), daß die Einheit j in der internen Schicht einen Schwellwert $\theta_j < 0$ hat.
- Von jeder Einheit j in der verdeckten Schicht existiert eine Verbindung zur Einheit A_j in der Ausgabeschicht, die mit dem Gewicht 1 bewertet ist.

Die Einheiten der Eingabeschicht sind binäre Eingabeeinheiten. Sie realisieren keine Funktionsberechnung. Ihre Ausgabe ist gleich ihrer Eingabe. Die Einheiten der internen Schicht und der Ausgabeschicht sind binäre Schwellwerteinheiten. Damit realisiert die verdeckte Schicht die Konjunktion der Klauselliterale. Die Ausgabeschicht berechnet eine Disjunktion über den Ergebnissen aller Klauseln.

Beispiel 3.1.1 *Sei das aussagenlogische Programm P gegeben mit:*

$$\begin{array}{rcl} A & \leftarrow & \\ B & \leftarrow & A, C \\ B & \leftarrow & \neg C, D, \neg E \\ D & \leftarrow & \neg E \end{array}$$

Für das Herbrand-Universum U_P und die Herbrand-Basis B_P gilt:

$$\begin{array}{rcl} U_P & = & \emptyset \\ B_P & = & \{A, B, C, D, E\} \end{array}$$

Abbildung 3.1 zeigt das 3-schichtige Netz zur Berechnung von T_P nach dem Algorithmus von [Kal94]. Die Zahlen in den Einheiten stellen die Schwellwerte dar. Die Gewichte der Verbindungen sind an den Pfeilen verzeichnet.

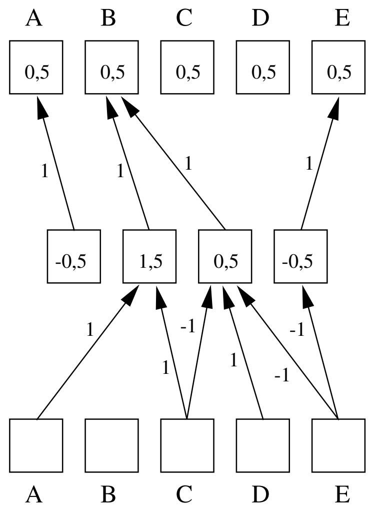


Abbildung 3.1: Das 3-schichtige Netz aus binären Schwellwerteinheiten zur Berechnung von T_P für $P = \{A; B \leftarrow A, C; B \leftarrow \neg C, D, \neg E; D \leftarrow \neg E\}$.

Nach der Konstruktion des vorwärtsgerichteten Netzes soll nun das zugehörige rekurrente Netz betrachtet werden. Dieses entsteht durch das Verbinden jeder Einheit A der Ausgabeschicht mit der korrespondierenden Einheit A der Eingabeschicht. Für die sich daraus ergebende Funktionsberechnung bedarf es weiterer Definitionen.

Definition 3.1.1 *Sei L ein vollständiger Verband mit dem Nullelement $\perp$ und sei $T : L \Rightarrow L$ eine Abbildung. Dann definieren wir mit einer ganzen Zahl d:*

$$\begin{aligned} T \uparrow 0 &= \perp \\ T \uparrow d &= T(T \uparrow (d-1)) \\ T \uparrow \infty &= \lim_{d \to \infty} T \uparrow d \end{aligned}$$

Mit Hilfe der Verbandstheorie kann man zeigen, daß die Menge aller Herbrand-Interpretationen für das Programm P einen vollständigen Verband bildet. Für die Abbildung $T_P : 2^{B_P} \rightarrow 2^{B_P}$ ist das Nullelement $\perp$ deshalb die leere Menge $\emptyset$.

Definition 3.1.2 *[Llo87] Sei L ein vollständiger Verband und $T : L \Rightarrow L$ eine Abbildung. Wir sagen T ist* monoton, *wenn $T(x) \leq T(y)$ für alle $x \leq y$.*

Lloyd hat gezeigt, daß für definite Programme der Operator T_P monoton bezüglich der Mengenoperation $\subseteq$ ist.

Definition 3.1.3 *[Llo87] Sei L ein vollständiger Verband und $T : L \Rightarrow L$ eine Abbildung. Wir sagen $a \in L$ ist der* kleinste Fixpunkt *von T, wenn a ein Fixpunkt ist (bedeutet $T(a) = a$) und für alle Fixpunkte b von T gilt $a \leq b$. Analog dazu definieren wir den* größten Fixpunkt.

Proposition 3.1.1 *[Llo87] Sei L ein vollständiger Verband und $T : L \Rightarrow L$ monoton. Dann hat T einen kleinsten Fixpunkt, $lfp(T)$, und einen größten Fixpunkt, $gfp(T)$.*

Proposition 3.1.2 *[Llo87] Sei P ein definites Programm. Dann ist $M_P = \{A \in B_P : A \text{ ist logische Konsequenz von } P\}$*

$$M_P = \{A \in B_P : P \models A\}$$

Proposition 3.1.3 *[Llo87] Sei P ein definites Programm. Dann ist $M_P = lfp(T_P) = T_P \uparrow \infty$.*

Mit den Propositionen 3.1.1 bis 3.1.3 wird gezeigt, daß die Berechnung $T_P \uparrow \infty$ für definite Programme einen kleinsten Fixpunkt hat, der unabhängig von der Anfangsinterpretation ist. Dieser Fixpunkt stellt die Semantik von P dar, weil er das kleinste Modell M_P repräsentiert. Das RNN-Modell, bestehend aus dem vorwärtsgerichteten Netz zur Berechnung von T_P und den rekurrenten Verbindungen, berechnet die Iteration $T_P \uparrow n$. Es gelangt damit für jedes definite aussagenlogische Programm nach endlicher Zeit in einen stabilen Zustand, der den Fixpunkt $T_P \uparrow \infty$ darstellt.

Kalinke hat für die Klasse der akzeptablen Programme eine ähnliche Feststellung zur Fixpunktbetrachtung gemacht.

Proposition 3.1.4 *[Kal94] Sei P ein akzeptables aussagenlogisches Programm. Für P existiert ein 3-schichtiges rekurrentes Netz, so daß jede mit einer beliebigen Anfangsinterpretation startende Berechnung konvergiert und den kleinsten Fixpunkt des Operators T_P liefert.*

Damit können wir folgendes festhalten.

Proposition 3.1.5 *Sei P ein Programm, das zur Klasse der akzeptablen aussagenlogischen oder der definiten aussagenlogischen Programme gehört. Das zu P korrespondierende RNN-Modell erreicht unabhängig vom initialen Eingabemuster nach endlicher Zeit einen stabilen Zustand. Dieser Zustand repräsentiert das kleinste Modell des Programmes P.*

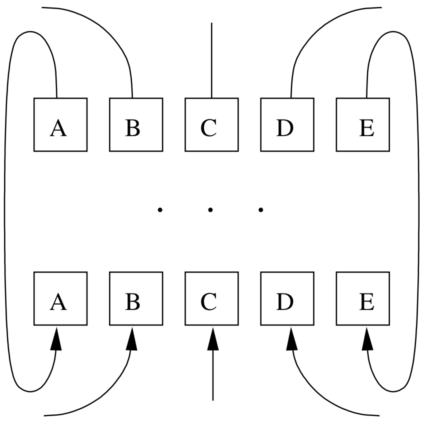


Abbildung 3.2: Die rekurrenten Verbindungen im RNN-Modell für das vorwärtsgerichtete Netz aus Abbildung 3.1.

Beispiel 3.1.2 *Betrachten wir erneut das Programm P aus Beispiel 3.1.1. Nach Definition 2.1.13 ist es ein akzeptables Programm, weil es akzeptabel in Bezug auf die Interpretation $I = B_P$ und die Stufenabbildung $|\ |$ mit:*

$$\begin{array}{rcl} A & \mapsto & 1 \\ B & \mapsto & 4 \\ C & \mapsto & 2 \\ D & \mapsto & 3 \\ E & \mapsto & 5 \end{array}$$

ist (jede Klausel ist akzeptabel in Bezug auf I und $|\ |$). Abbildung 3.2 zeigt die rekurrenten Verbindungen des RNN-Modelles für P. Die interne Struktur des Netzes entspricht dem vorwärtsgerichteten Netz aus Abbildung 3.1. Mit der willkürlichen Anfangsinterpretation $\{E\}$ realisiert das RNN-Modell damit folgende Berechnungen:

$$\begin{array}{rcl} T_P(\{E\}) \uparrow 0 & = & \emptyset \\ T_P(\{E\}) \uparrow 1 & = & \{A\} \\ T_P(\{E\}) \uparrow 2 & = & \{A, D\} \\ T_P(\{E\}) \uparrow 3 & = & \{A, B, D\} \\ T_P(\{E\}) \uparrow 4 & = & \{A, B, D\} = lfp(T_P) \end{array}$$

Abschließend soll die Komplexität des RNN-Modelles betrachtet werden. Sei n die Anzahl der Klauseln und m die Anzahl der Grundatome des Programmes P, so ist die Anzahl der Netzeinheiten durch $O(m+n)$ beschränkt. Die Anzahl der Verbindungen zwischen den Schichten ist abhängig von $O(m \times n)$. Der Platzaufwand des RNN-Modelles ist damit linear zur Größe der Wissensbasis. Die Berechnung von T_P benötigt mit $O(m+n)$ Prozessoren konstante Zeit, nämlich genau 3 Schritte. Das Erreichen des stabilen Zustandes ist linear abhängig von der Länge der längsten Inferenzkette. Im schlechtesten Fall ist dies gleich der Anzahl der Klauseln in P. Das RNN-Modell ist ein optimaler paralleler Algorithmus für die Berechnung von T_P.

3.2 Darstellung von Prädikatenlogik

3.2.1 Grundinstanziierung

Die Erweiterung des RNN-Modelles auf Prädikatenlogik stellt erneut die Frage nach der Kodierung von Interpretationen. Die einfachste Variante besteht darin, jedes Element der Herbrand-Basis wieder durch eine binäre Einheit zu kodieren, deren Ausgabe den Wahrheitswert der atomaren Grundformel angibt. Aufbauend auf dieser Kodierung können die Regeln des Programmes in derselben Weise implementiert werden, wie im aussagenlogischen Fall. Dafür ist es notwendig, alle möglichen Grundinstanzen der Programmklauseln zu erstellen. Wir wollen deshalb dieses Vorgehen als Grundinstanziierung bezeichnen.

Beispiel 3.2.1 *Betrachten wir das Beispiel aus 2.1.1 mit*

$$\begin{array}{rcl} p(f(a)) & \leftarrow & \\ q(X,a) & \leftarrow & p(X) \\ q(X,Y) & \leftarrow & q(Y,X) \end{array}$$

Die unter Nutzung des Herbrand-Universums U_P grundinstanziierten Programmklauseln aus der zweiten und dritten Regel stellen sich in der Form

$$\begin{array}{rcl} q(a,a) & \leftarrow & p(a) \\ q(f(a),a) & \leftarrow & p(f(a)) \\ & \dots & \\ q(a,a) & \leftarrow & q(a,a) \\ q(f(a),a) & \leftarrow & q(a,f(a)) \\ & \dots & \\ q(a,f(a)) & \leftarrow & q(f(a),a) \\ q(f(a),f(a)) & \leftarrow & q(f(a),f(a)) \\ & \dots & \end{array}$$

dar. Es entsteht eine abzählbar unendliche Menge an Grundregeln. Das so transformierte Programm kann nach dem Algorithmus aus [Kal94] direkt kodiert werden. In Abbildung 3.3 wird das entsprechende RNN-Modell dargestellt.

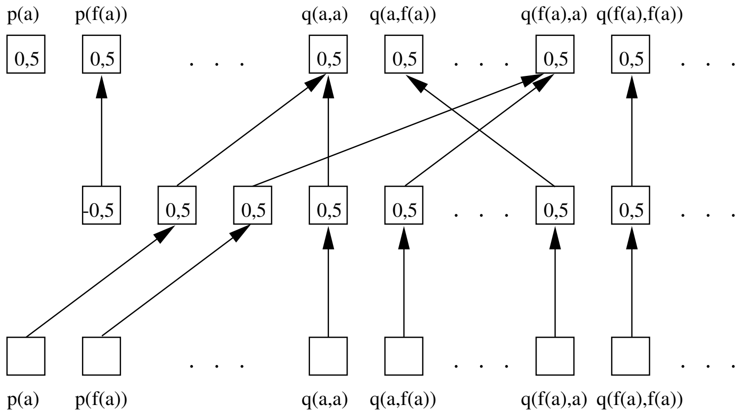


Abbildung 3.3: Ein Ausschnitt aus dem RNN-Modell für das prädikatenlogische Programm P: $\{p(f(a)); q(X,a) \leftarrow p(X); q(X,Y) \leftarrow q(Y,X)\}$ bei Grundinstanziierung. Die Gewichte der Verbindungen haben den Wert 1.

Es wird offensichtlich, daß diese Form der Kodierung eine Art Transformation in die Aussagenlogik darstellt und nur deshalb direkt mit dem gleichen Algorithmus bearbeitet werden kann. In diesem Zusammenhang muß darauf hingewiesen werden, daß bisher nur Formeln mit Allquantoren bzw. Allabschluß betrachtet wurden. In der Prädikatenlogik können jedoch auch Existenzquantoren auftreten. Für deren Darstellung ist keine Instanziierung möglich. Prädikate mit existenzquantifizierten Variablen müßten gegebenenfalls durch neu eingeführte Prädikate ohne diese Variablen repräsentiert werden.
Ein sofort ersichtlicher Nachteil ist die Unendlichkeit des Modelles. Häufig geht man in solchen Fällen den Weg, die Menge der Terme auf Konstanten zu beschränken und jede andere Form von Funktionssymbolen zu verbieten. Damit wird die Menge der Grundformeln B_P endlich. Bei einer durchschnittlichen Stelligkeit $\bar{s}$ der Prädikatssymbole und C Konstanten beträgt die Anzahl der Eingabe- und Ausgabeeinheiten trotzdem noch $O(C^{\bar{s}})$. Die Anzahl der grundinstanziierten Klauseln ist sogar noch zusätzlich abhängig von der durchschnittlichen Anzahl $\bar{m}$ der Literale im Körper der Klauseln. Alle Grundinstanzen müssen dort kombinatorisch in Relation zueinander gesetzt werden, so daß die Anzahl der verdeckten Einheiten durch $O(C^{(\bar{s}\times\bar{m})})$ beschränkt ist. Auch mit einer derartigen Einschränkung hätten wir also für die Grundinstanziierung exponentiellen Aufwand zu verzeichnen.
Es gibt eine Vielzahl weiterer Annahmen und Vereinfachungen, wie strikte Verwendung von zweistelligen Prädikatssymbolen, Kodierung von Termtupeln, etc., die die Größe des RNN-Modelles bei Grundinstanziierung verringern sollen. Am prinzipiell exponentiell hohen Aufwand eines solchen Modelles ändern sie jedoch nichts.

3.2.2 Approximation durch reellwertige Funktionen

Ein weiterer Weg zur Darstellung von Prädikatenlogik ist der Einsatz von reellwertigen Netzeinheiten. Mit solchen Einheiten ließe sich eine unendliche Zahl von Formeln kodieren, weil die Anzahl reeller Werte in einem festen Ausgabeintervall auch unendlich ist. Findet man eine Realisierung, die diese Eigenschaft reellwertiger Einheiten ausnutzen kann, so könnte man die Platzkomplexität gegenüber der Grundinstanziierung erheblich verringern. Kodiert man zum Beispiel jedes Prädikat p mit der Stelligkeit s durch eine einzige Netzeinheit, dann müssen unter Umständen mehrere Elemente der Menge $(U_P)^s$ von dieser Einheit dargestellt werden, was mehreren gültigen Prädikatsinstanzen entspräche. Die durch die Einheit zu realisierende Abbildung hätte die Form $\mathbb{R} \rightarrow 2^{(U_P)^s}$ und im Gegenzug hätte das konnektionistische Modell linearen Platzaufwand in den äußeren Netzschichten. Selbst von Hand ließe sich eine solche oder ähnliche Kodierung finden, aber offen bleibt die Frage, wie man damit die Regeln der Wissensbasis implementieren kann.

In [KS96] und [HKS98] wird das Problem von einem allgemeineren Standpunkt betrachtet. Das zu erstellende neuronale Netz soll den Operator T_P für ein gegebenes Programm P berechnen. Für die Darstellung der Eingabe- und Ausgabeinterpretationen mit reellen Zahlen benötigt man deshalb eine injektive Abbildung $R\colon 2^{B_P} \rightarrow \mathbb{R}$. Injektiv deshalb, damit zur Ermittlung der Ausgabeinterpretationen eine inverse Abbildung $R^{-1}\colon \mathbb{R} \rightarrow 2^{B_P}$ vorhanden ist. Für das Definieren dieser Abbildung R gibt es vielfältige Möglichkeiten. Entscheidend ist nun, für die Berechnungen des Netzes eine reellwertige Funktion $f_P\colon \mathbb{R} \rightarrow \mathbb{R}$ zu finden, für die gilt:

$$f_P(x) = R(T_P(R^{-1}(x))). \tag{3.1}$$

Dabei macht man sich die Feststellung von Fitting in [Fit94] zunutze, daß T_P für die Klasse der akzeptablen Programme eine Kontraktion bezüglich eines Abstandmaßes auf Interpretationen ist. Dies kann man mit den dort vorgestellten metrischen Methoden auch für die Klasse der rekurrenten Programme zeigen, für welche damit die nach 3.1 definierte Funktion stetig ist. Jede stetige reellwertige Funktion wiederum kann nach Funahashi ([Fun90]) durch ein 3-schichtiges neuronales Netz approximiert werden. Approximieren heißt dabei, daß für eine beliebig kleine Genauigkeit ϵ eine Funktion $\tilde{f}_P$ gefunden werden kann, so daß zwar $f_P \neq \tilde{f}_P$, aber für alle Argumente x die Ungleichung $|f_P(x) - \tilde{f}_P(x)| < \epsilon$ gilt. Verbal gesagt, kann man sich beliebig genau an f_P annähern.
Das Ergebnis der Untersuchungen ist ein Theorem, welches sagt, daß für jedes rekurrente logische Programm ein 3-schichtiges neuronales Netz existiert, welches die Funktion T_P approximiert. Hölldobler, Kalinke und Störr zeigen in [HKS98] außerdem, daß die Iteration der approximierenden Funktion $\tilde{f}_P$ auch die Semantik des Programmes, nämlich den Fixpunkt $T_P \uparrow \infty$ approximiert. Die Autoren geben ein konkretes Beispiel an, in dem der komplette Zusammenhang dargestellt wird.
Durch Approximation mit reellwertigen Funktionen ist die Berechnung von T_P zumindest für alle rekurrenten Programme garantiert. Dies ist das wichtigste Ergebnis der Untersuchungen. Das größte Problem dieses Ansatzes ist jedoch, daß er absolut keinen Hinweis gibt, wie man ein derartiges Netz konstruieren kann.

3.2.3 Konnektionistische Termrepräsentationen

Als Konnektionistische Termrepräsentationen bezeichnet man Modelle neuronaler Netze, die Terme einer logischen Sprache auf reelle Werte kodieren. Sie realisieren damit eine Abbildung der Form $U_P \rightarrow \mathbb{R}^n$ $(n \geq 1)$. Damit bietet sich die Möglichkeit, die entsprechenden Kodierungen in einem rekurrenten neuronalen Netz nach RNN-Vorbild zur Realisierung von Inferenzprozessen zu nutzen. Dabei werden an die Repräsentationen vier besondere Beurteilungskriterien gelegt, die Eigenschaften eines verwertbaren Systems sein sollten.

- **Verteilung:**
  Ein Term wird repräsentiert durch mehrere reellwertige Netzeinheiten. Die realisierte Abbildung hat die Form $U_P \rightarrow \mathbb{R}^n$ $(n > 1)$. Geringe Störungen sollen dadurch nicht zu signifikanten Problemen beim Kodieren oder Dekodieren der Terme führen.

- **Graceful Degradation:**[1]
  Wenn viele Elemente in einem Netz zur Darstellung eines Termes beitragen, gilt folgendes Verhalten: Mit steigender Problemgröße (Termtiefe) sinkt die Genauigkeit der Darstellung, es kann jedoch mehr als eine endliche Zahl von Termen repräsentiert werden.

- **Struktursensitivität:**
  Termen mit ähnlicher Struktur wird eine ähnliche Repräsentation zugewiesen. Damit wird die Implementierung von symbolischen Aufgaben, wie Matching oder Unifikation vereinfacht.

- **Generalisierungsfähigkeit:**
  Nach dem Training des konnektionistischen Modells mit einer festen Anzahl Termen ist das Netz in der Lage, weitere Terme darzustellen. Dies geschieht durch die Nutzung von Teiltermen und dem Finden ähnlicher Repräsentationen für ähnliche Terme.

In [Kal97] diskutiert Kalinke den Einsatz der Modelle HRR (Holographic Reduced Representation), RAAM (Recursive Autoassociative Memory) und LRAAM (Labeling RAAM) unter dem Gesichtspunkt der beschriebenen Eigenschaften.
Das HRR-Modell ist streng genommen keine konnektionistische Repräsentation. Es nutzt eine assoziative Operation zum Verknüpfen von Rollen und Argumenten und bildet damit Terme auf Vektoren fester Länge ab. Die Berechnung kann jedoch auch in einem neuronalen Netz realisiert werden und seine Einbindung in ein RNN-Modell in Form zusätzlicher Schichten wäre denkbar.
Die RAAM und die LRAAM sind rekurrente Netze zur komprimierten Repräsentation

[1] Der Begriff läßt sich schlecht ins Deutsche übertragen und soll deshalb in dieser Form verwendet werden.

	HRR	**RAAM**	**LRAAM**
kodierbare Termtiefe	3	7	3
aufwendiges Training	nicht notwendig	notwendig	notwendig
aufwendige Dekodierung	nicht notwendig	notwendig	notwendig
Verteilung	+	mehr lokal	+
Graceful Degradation	+	+	+
Struktursensitivität	+	+	+
Generalisierungsfähigkeit	+	nicht akzeptabel	keine

Tabelle 3.1: Die Eigenschaften der konnektionistischen Termrepräsentationen HRR, RAAM und LRAAM. Das Symbol „+“ zeigt das Vorhandensein des entsprechenden Kriteriums an.

von Baumstrukturen. Sie werden mit den darzustellenden Termen angelernt und bilden in der internen Schicht die resultierende Kodierung, die dann zur Darstellung von Interpretationen der Prädikatenlogik genutzt werden könnten. Der Unterschied zwischen den Modellen besteht darin, daß bei der LRAAM auch konkrete Bezeichner von Funktionen oder Konstanten für die Kodierung genutzt werden. Eine ausführliche Darlegung aller drei Modelle ist in [Sei96] zu finden.
Tabelle 3.1 faßt die Ergebnisse von Kalinke zusammen. Während das HRR-Modell den vier bereits angesprochenen Eigenschaften genügt, ist die Generalisierungsfähigkeit der anderen Algorithmen nicht zufriedenstellend. Zusätzlich sind bei RAAM und LRAAM aufwendige Trainings- und Dekodierungsvorgänge notwendig. Der entscheidende Nachteil aller Verfahren ist die stark begrenzte Termtiefe. Terme, in denen mehr als 7 Funktionssymbole verschachtelt auftreten, konnten von keinem der Modelle kodiert werden. Die Termtiefe ist damit für die Einbindung in ein RNN-System für prädikatenlogische Programme im allgemeinen Fall nicht ausreichend. Kalinke empfiehlt aufgrund der Experimente die Entwicklung eines hybriden Systems, aus einem konnektionistischen Teil zum Produzieren der reduzierten Repräsentation und einem Speicher zur Lagerung der Kodierungen und Verwendung in komplexeren Strukturen.

4. Reflexives Schließen mit SHRUTI

Die Diskussionen zum Schließen in Prädikatenlogik mit konnektionistischen Strukturen haben gezeigt, daß die Hauptproblematik in der hohen Platz- und Zeitkomplexität solcher Modelle liegt. Shastri und Ajjanagadde ([SA93]) entwickelten deshalb ein System namens SHRUTI, das trotz tausender Fakten und Regeln in der Wissensbasis innerhalb von wenigen Millisekunden komplexe Inferenzen durchführen kann. Dabei verwenden sie Konzepte, die besonders interessant für die Darstellung von Prädikatenlogik im RNN-Modell sind. Wir wollen uns deshalb in den ersten Abschnitten dieses Kapitels ausgiebig mit ihrer Arbeit beschäftigen und das Vorwärtsschließen in einem SHRUTI-System erarbeiten. Die Erweiterungen und Kritikpunkte werden in den den Abschnitten 4.4 und 4.5 erörtert.

4.1 Grundlagen

4.1.1 Reflexives Schließen

Bei den Bestrebungen, einen schnellen Inferenzmechanismus zu bauen, orientierten sich Shastri und Ajjanagadde daran, das schnelle, scheinbar anstrengungslose Schließen beim Menschen nachzubilden, der schwierige Probleme mit endlichem Speicher in extrem kurzer Zeit lösen kann. Diese Form der menschlichen Inferenz wird als reflexives Schließen bezeichnet. Es findet unbewußt, spontan und automatisiert statt und ist deshalb so effektiv. Viele Vorgänge, wie Sprachverstehen, Alltagsschließen und unbewußte Reaktionen, basieren auf ihm. Reflexives Schließen bedarf keiner zusätzlichen Hilfsmittel, sondern benutzt lediglich die natürlichen Resourcen des Menschen. Das Gegenteil zum reflexiven Schließen bildet das reflektive Schließen. Der Mensch muß dafür zielgerichtet und bewußt Schlußfolgerungen ziehen. Er bearbeitet dabei so schwierige Probleme, daß er für deren Lösung meist noch Werkzeuge, wie Papier und Stift, benötigt, um die komplexen Zusammenhänge besser verstehen zu können.
Als Grundlage für reflexives Schließen dient eine sehr große Wissensbasis, deren Inhalte dauerhaft gespeichert und schnell verfügbar sind. Sie wird als Langzeitspeicher bezeichnet. Ein Inferenzvorgang beginnt, wenn dafür eine Eingabe vorhanden ist. Sie könnte zum Beispiel in einem gesprochenen Satz bestehen, dessen Bedeutung verstanden werden muß. Es wird davon ausgegangen, daß Eingaben für reflexives Schließen einen sehr

kleinen Teil des Langzeitwissens ansprechen. Auch der für die Verarbeitung des Wissens notwendige Arbeitsspeicher ist in der Regel wesentlich kleiner als die Wissensbasis. Ein in sich geschlossener Inferenzvorgang, bestehend aus einer Eingabe und den mittels Langzeit- und Arbeitsspeicher ermittelten Folgerungen, wird als Episode bezeichnet. Eine Episode im reflexiven Schließen wäre zum Beispiel mit dem Verstehen des Satzes und einer eventuellen Reaktion bzw. Antwort abgeschlossen.
Mit der Kenntnis dieser psychologischen Hintergründe beschreiten die Autoren einen Weg, der die Klasse effektiv lösbarer Probleme einschränkt. Reflexives Schließen ist eingeschränktes Schließen und darum auch nur für eine bestimmte Inferenzklasse modellierbar. In [Sha93] definiert Shastri die Anforderungen an eine solche Klasse von Inferenzen, die reflexives Schließen nachbilden kann. Er verknüpft damit folgende Bedingungen:

- Die Zeit zum Schließen ist unabhängig von der Größe des Langzeitspeichers und ist nur abhängig von der Größe der Eingabe und der Tiefe des Ableitungsbaumes, der mit der Inferenz assoziiert ist.
- Der Platz, der zur Kodierung des Langzeitspeichers benötigt wird, und der Platz für den Arbeitsspeicher während des Schließens darf nicht schlechter als linear zur Größe des Langzeitspeichers sein.

Mit der Anforderung an die Zeitkomplexität für reflexives Schließen wird klar, daß eine Menge von Inferenzregeln in der Regel nicht komplett sequentiell abgearbeitet werden kann. Im Rahmen einer Episode müssen unterschiedliche Pfade von Schlußfolgerungen deshalb parallel durchsucht werden, damit diese den Effektivitätsbedingungen genügt. Für ein Modell zur Realisierung von effektivem Schließen bedeutet dies, es muß parallel auf der Ebene der Inferenz[1] sein und somit massiv parallele Berechnungen erlauben.

Für die Modellierung ihres Systems stellen Shastri und Ajjanagadde auch Anforderungen an die Netzknoten, die verwendet werden können. Diese Bedingungen sind biologisch motiviert und orientieren sich am Menschen, der mit einer endlichen Menge einfacher organischer Einheiten reflexives Schließen vollzieht. Die Autoren gehen von folgenden Vorraussetzungen aus:

- Netzknoten berechnen einfache Funktionen.
- Netzknoten können nur begrenzte Zustandsinformationen speichern.
- Netzknoten können keine symbolischen Namen oder Zeiger ausgeben.
- Es gibt keinen zentralen Kontrollmechanismus, der bei den Knoten spezielle Operationen während jedes Verarbeitungsschrittes durchführt.

Die Autoren legen in ihrer gesamten Arbeit großen Wert auf die biologische und psychologische Plausibilität ihres Modelles. Alle Mechanismen und Kodierungen in SHRUTI werden unter diesem Gesichtspunkt entwickelt und diskutiert. Im folgenden werden wir uns nun im wesentlichen auf die Aspekte der Wissensverarbeitung beschränken.

[1] engl: knowledge-level parallelism

4.1.2 Dynamische Bindung von Argumenten

Von grundlegender Bedeutung für die Repräsentation von prädikatenlogischen Formeln ist die Bindung von Termen an Variablen. Derartige Bindungen müssen im Rahmen des Inferenzprozesses ständig neu erstellt werden, wenn durch vorhandenes Wissen auf neue Formeln geschlossen werden kann oder wenn neue Teilformeln bewiesen werden müssen. Ein Inferenzmechanismus mit den Anforderungen an reflexives Schließen muß deshalb in der Lage sein, Bindungen innerhalb kürzester Zeit auswerten zu können, ohne durch die Realisierung die geforderte Platzkomplexität zu verletzen. Gleiches gilt für das Weiterleiten von Bindungen, welches in konnektionistischen Strukturen häufig als Propagieren bezeichnet wird.
Bei der statischen Bindung werden Konzepte durch feste physische Verbindungen zwischen den sie repräsentierenden Einheiten in Relation zueinander gesetzt. Diese Form der Bindung ist sehr starr und läßt sich in einem konnektionistischen System schwer verändern oder erzeugen. Eine Möglichkeit wäre, die Gewichte solcher Verbindungen zur Repräsentation der Bindung selbst zu benutzen und ihren Wert im Rahmen des Schließens zu verändern. Das bedeutet jedoch, daß eine sehr große Zahl von Relationen, wenn nicht sogar alle möglichen Relationen, vor der eigentlichen Inferenz erstellt werden müßten. Der Platzaufwand dafür ist exponentiell von der Zahl der Konzepte abhängig und die Anforderungen an die Platzkomplexität sind damit nicht zu erfüllen.
Aus diesen Gründen scheint der Ansatz statischer Bindung eher für Wissen geeignet zu sein, das über einen langen Zeitraum gespeichert und kaum verändert werden soll. Das Für und Wider statischer Bindung diskutiert Cooper in [Coo93].

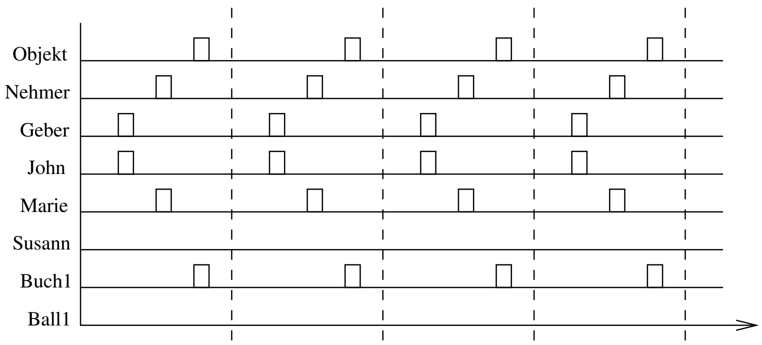


Abbildung 4.1: Die Dynamische Bindung von Argumenten mittels zeitlicher Synchronität. Die gegebenen Aktivierungen repräsentieren die Formel *geben*(*John*, *Marie*, *Buch1*).

SHRUTI verwendet einen Ansatz dynamischer Bindung, der auf zeitlicher Synchronität basiert. Dabei wird in der Aktivierung von Knoten eine Zeitachse genutzt und Wissen zyklisch aktiviert. Netzknoten, die gleichzeitig aktiviert sind, stehen in Relation zueinander. Das Prinzip der zeitlich synchronen Aktivierung wird in Abbildung 4.1 ver-

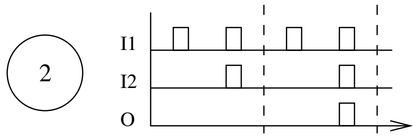
deutlicht. Das Prädikat *geben* besteht aus den Argumenten *Geber*, *Nehmer* und *Objekt*. Die Konzepte *John*, *Marie*, *Susann*, *Buch1* und *Ball1* sind Konstanten im prädikatenlogischen Sinn. Seien nun die Netzknoten, die die Konzepte und Argumente repräsentieren, wie in der Abbildung aktiviert, so ist *John* in der Rolle *Geber*, *Marie* steht für *Nehmer* und *Buch1* ist an *Objekt* gebunden. Dies zusammen repräsentiert die Formel *geben*(*John*, *Marie*, *Buch1*). Die Konstanten *Susann* und *Ball1* sind in keiner Phase aktiviert und deshalb auch nicht Füller von Argumenten einer Prädikatsinstanz. Die Dauerhaftigkeit des abgebildeten Wissens wird durch die zyklische Aktivierung erreicht, wobei ein Zyklus in der Abbildung dargestellt wird durch die vertikal gestrichelten Linien.
Mit dieser Repräsentation wird klar, daß Bindungen nur dann eindeutig sind, wenn zu einem Zeitpunkt immer maximal eine termrepräsentierende Netzeinheit aktiviert ist. Wären *Susann* und *Marie* gleichzeitig aktiviert, so bestände eine Bindung zwischen zwei Konstanten, was aus prädikatenlogischer Sicht keinen Sinn macht. Alle Terme werden deshalb immer in paarweise unterschiedlichen Zeitpunkten aktiviert. Diese Zeitpunkte bezeichnen wir als Phasen. Die Menge aller dargestellten Phasen bildet den bereits angesprochenen Zyklus.
In einem konnektionistischen Modell können solche Bindungen auf einfache Weise ausgewertet und propagiert werden. Dabei ist lediglich die Erweiterung der Aktivierungen um den Faktor Zeit zu beachten. Deutlich wird auch, daß die eigentliche Entscheidung über das repräsentierte Wissen auf einer Metaebene unter Kenntnis der Terme und der Prädikatssymbole stattfindet. Feuern zwei Einheiten in der gleichen Phase, so ist die Einheit, die einen Term repräsentiert, der Füller für die aktivierte Einheit, die zu einem Prädikatssymbol korrespondiert.

Das Propagieren der dynamischen Bindungen hat direkt zur Folge, daß eine Anzahl besonderer Netzeinheiten benötigt wird, die entsprechend zeitsensitiv arbeiten. Abbildung 4.2 zeigt die verschiedenen Einheiten, die in SHRUTI genutzt werden. Sie sind jeweils durch einen internen Schwellwert θ gekennzeichnet. Außerdem realisiert SHRUTI noch sogenannte „verbietende Verbindungen“, die von Shastri und Ajjanagadde nicht als Netzknoten bezeichnet werden. Sie realisieren aber ebenfalls eine deterministische Funktion und sind damit als gleichwertige Einheit betrachtbar. Die Abbildung verdeutlicht außerdem die Funktion der Einheiten, wie sie in [SA93] beschrieben wird:

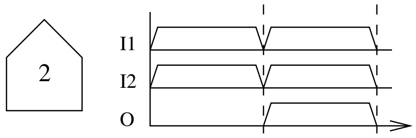
- ρ-btu
  Die ρ-btu-Einheit ist eine binäre Schwellwerteinheit[2]. Ist die Summe der Eingaben in einer Phase größer oder gleich dem internen Schwellwert θ der Einheit, dann wird die Einheit in derselben Phase des nächsten Zyklus aktiviert.

- τ-und
  Ein τ-und-Knoten wird aktiviert, wenn er innerhalb eines Zyklus einen ununterbrochenen Pulsstrom als Eingabe erhält, der den Schwellwert erreicht. Die Ausgabe dieses Typs ist eine ununterbrochene Aktivierung im nächsten Zyklus.

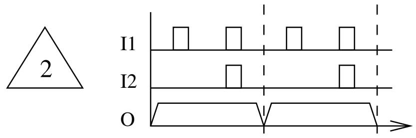
[2] btu = binary treshold unit



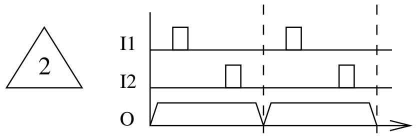
(a) ρ-btu



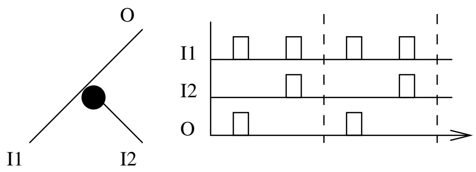
(b) τ-und



(c) τ-oder



(d) τ-oder-ähnlich



(e) verbietende Verbindung

Abbildung 4.2: Die Netzeinheiten in SHRUTI und verbietende Verbindungen. Die Einheiten werden durch die Symbole dargestellt und haben einen Schwellwert $\theta = 2$. Die Diagramme zeigen die Ausgabe O in Abhängigkeit von den Eingaben $I1$ und $I2$.

- τ-oder, τ-oder-ähnlich
  Eine τ-oder-Einheit wird aktiviert, wenn die Summe der eingehenden Aktivierungen in einer Phase den Schwellwert θ übersteigt. Die τ-oder-ähnlichen-Knoten unterscheiden sich davon nur dadurch, daß sie aktiviert werden, wenn die Eingaben in mindestens θ verschiedenen Phasen eines Zyklus aktiviert sind. Die Ausgabe der beiden Knotentypen ist ein ununterbrochener Pulsstrom im aktuellen Zyklus[3]. Beide Typen werden durch dasselbe Symbol dargestellt.

- verbietende Verbindungen
  Eine verbietende Verbindung unterbricht den Aktivierungsfluß auf der angeschloßenen Leitung genau in den Phasen, in denen sie selbst aktiv ist.

Die Nutzung der Netzknoten im Rahmen der Kodierung wird in den folgenden Abschnitten dargelegt.

[3] Dieses Verhalten ist eine Idealisierung. Tatsächlich kann eine Einheit erst ab dem Auftreten der Aktivierung die Ausgabe mit Länge des Zyklus realisieren.

4.1.3 Konstanten und Prädikate, Regeln und Fakten

Bei der Kombination symbolischer Verarbeitung und neuronaler Architektur, wie in SHRUTI, müssen sowohl die Syntax, als auch die Semantik einer Wissensbasis im System kodiert werden. Entsprechend ist für deren Darstellung eine Anordnung von Netzeinheiten zu definieren.

SHRUTI beschränkt sich in seiner Syntax auf Prädikatssymbole und Konstanten. Das Auftreten von Funktionssymbolen (Ausnahme Konstanten) im Alphabet der Logik wird verboten. Damit umgeht man das Problem, gegebenenfalls Terme mit Verschachtelungen von Funktionen darstellen und an Argumente binden zu müssen.
Eine Konstante in SHRUTI wird dargestellt durch eine einzelne ρ-btu-Einheit. Diese Einheit zeigt nach dem Prinzip der synchronen Aktivierung an, welche Phase eine Bindung eines Argumentknotens an diese Konstante repräsentiert.
Prädikatssymbole werden kodiert mit einem Collector, einem Enabler[4], und jeweils einer Einheit für jedes Argument des Prädikates. Sei s die Stelligkeit eines Prädikates, so wird es durch $s + 2$ Netzknoten repräsentiert. Diese $s + 2$ Einheiten stellen dann genau eine Instanz dieses Prädikates dar. Für das Prädikat *geben* ist dies auch in Abbildung 4.3 zu sehen. Der Collector und der Enabler sind τ-und-Einheiten und werden entsprechend ihres Zustandes über einem ganzen Zyklus aktiviert. Ein aktivierter Collector gibt an, daß die Instanz des Prädikates gültig ist. Er zeigt das vorhandene Wissen. Der Enabler spielt vor allem beim Rückwärtsschließen eine große Rolle. Er ist aktiviert, wenn das Prädikat eine Anfrage (dynamisch oder initial erstellt) darstellt. Eine Anfrage ist dabei eine Prädikatsinstanz, die im Laufe der Inferenz bewiesen werden soll. Der Beweis erfolgt, wenn nach endlicher Zeit der zugehörige Collector aktiviert wird. In einem vorwärtsschließendem SHRUTI-System werden die Enabler nur für die Aktivierung des Langzeitwissens benötigt. Die Argumenteinheiten sind ρ-btu-Knoten. Sie zeigen die Bindung der Stellen des Prädikates an. Betrachtet man die Aktivierungen aus Abbildung 4.1, dann sind die Verläufe von *Geber*, *Nehmer* und *Objekt* genau die Aktivierungen der 3 ρ-btu-Knoten des Prädikates *geben*. Ist eine Argumenteinheit nicht gebunden, aber die Prädikatsinstanz durch den Collector aktiviert, dann kodiert dies eine existenzquantifizierte Variable.

Auf der Seite der Semantik einer Wissensbasis unterscheidet SHRUTI nach Regeln und Fakten. Die Zuordnung einer Formel zu einer dieser Gruppen erfolgt dabei lediglich nach ihrem Aufbau. Regelwissen in SHRUTI hat die Form

$$\forall X_1, \ldots, X_m[p_1(\ldots) \wedge p_2(\ldots) \ldots \wedge p_n(\ldots) \rightarrow \exists Z_1, \ldots Z_l q(\ldots)]$$

während Fakten atomare Formeln der Form

$$\exists Z_1, \ldots, Z_l q(\ldots)$$

sind. Die Anfragen, die im Falle des Rückwärtsschließens an das System gestellt werden können, haben ebenfalls die Form von Faktenwissen.

[4] Die englischen Bezeichnungen werden häufig in der Literatur verwendet und sollen deshalb hier nicht übersetzt werden.

Die exakte Syntax und Implementierung der Regeln einer Wissensbasis ist abhängig davon, ob ein vorwärts- oder rückwärtsschließender SHRUTI realisiert werden soll. Für diese Problematik wird deshalb auf die nächsten Abschnitte verwiesen.

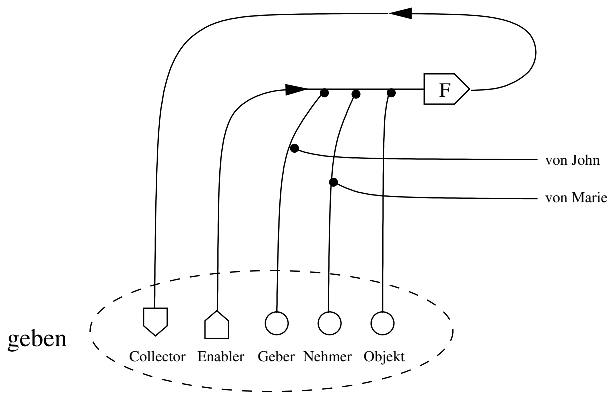


Abbildung 4.3: Die Implementierung des Langzeitfaktes $geben(John, Marie, X)$ in einem SHRUTI-System.

Die Argumente von Fakten sind Konstanten oder paarweise unterschiedliche existenzquantifizierte Variablen. Die Implementierung derartiger Langzeitfakten erfolgt nach einem einfachen Schema. Abbildung 4.3 zeigt das Prädikat *geben* zusammen mit dem Fakt $geben(John, Marie, X)$ („John gibt Marie etwas"). Das Prädikat *geben* besteht aus dem Collector, dem Enabler und den Argumenteinheiten *Geber*, *Nehmer* und *Objekt*. Der Enabler ist verbunden mit einem Faktknoten F und dieser wiederum mit dem Collector von *geben*. Von allen Argumentknoten bestehen verbietende Verbindungen zu der Kante zwischen Enabler und F und jede dieser Kanten ist über eine verbietende Verbindung mit der Konstanteneinheit verbunden, die der Fakt an dieser Stelle beinhaltet. Der Faktknoten kann folglich nur aktiviert werden, wenn auf keiner der 3 Kanten eine Aktivierung vorliegt. Die Verbindung von *Geber* verletzt diese Bedingung, wenn *Geber* an eine andere Konstante, als an *John* gebunden ist. Gleiches gilt analog für *Nehmer* und *Marie*. Die 3. Kante stellt eine existenzquantifizierte Variable dar und blockiert den Fakt, wenn irgendeine Bindung bei *Objekt* vorhanden ist. Wird nun eine Anfrage bezüglich des Prädikates *geben* gestellt, so kann diese Anfrage nur durch den Fakt F bewiesen werden, wenn die Bindungen im Prädikat genau mit den Bindungen des Langzeitfaktes übereinstimmen. In diesem Fall würde der Collector nach zwei Inferenzzyklen aktiviert werden.

4.2 Rückwärtsschließen mit SHRUTI

Das Rückwärtsschließen im SHRUTI-Modell wird von Shastri und Ajjanagadde ausführlich in ihrer Arbeit ([SA93]) vorgestellt. Die Autoren geben darin an, daß sie die Auswahl bezüglich des vorgestellten Inferenzmechanismus willkürlich getroffen haben. Ihren Ausführungen zufolge arbeitet SHRUTI gleichermaßen als Rückwärtsschließer und als Vorwärtsschließer. Die verwendeten Konzepte sind dabei vollkommen gleich. Lediglich die Richtung der Inferenzen und die Implementierung der Regeln ändert sich.
Der rückwärtsschließende SHRUTI soll hier an einem einfachen Beispiel aus der Originalarbeit vorgestellt werden.

Beispiel 4.2.1 *Es sei folgendes Langzeitwissen vorhanden:*

$$\begin{array}{rcl} & \rightarrow & geben(John, Marie, Buch1) \\ & \rightarrow & \exists X\, kaufen(John, X) \\ & \rightarrow & besitzen(Marie, Ball1) \\ geben(X,Y,Z) & \rightarrow & besitzen(Y,Z) \\ kaufen(X,Y) & \rightarrow & besitzen(X,Y) \\ besitzen(X,Y) & \rightarrow & kann\text{-}verkaufen(X,Y) \end{array}$$

Es soll herausgefunden werden, ob Marie in der Lage ist, das Buch1 zu verkaufen. Die entsprechende Anfrage an das System lautet: ‚kann-verkaufen(Marie, Buch1) ?‘

Abbildung 4.4 zeigt den Aufbau des rückwärtsschließenden SHRUTI für die Fakten und Regeln aus Beispiel 4.2.1.

Das Modell enthält die Prädikate *geben*, *kaufen*, *besitzen* und *kann-verkaufen* und die Konstanteneinheiten *John*, *Marie*, *Ball1* und *Buch1*. An den drei ersten der Prädikate ist jeweils ein Langzeitfakt der Wissensbasis implementiert. Die Regeln werden zwischen den sie betreffenden Prädikatszeichen realisiert. Dabei werden die Argumenteinheiten des Prädikates im Klauselkopf und seine Enabler-Einheit rückwärts mit den korrespondierenden Einheiten des Prädikates im Klauselkörper verbunden und nur die Collectoren in der Gegenrichtung.
Der Inferenzprozeß beginnt mit dem Stellen einer Anfrage. Dadurch werden die Argumente und der Enabler rückwärts propagiert und es entstehen neue, dynamische Anfragen. Durch mehrere Regeln mit demselben Prädikatszeichen im Klauselkopf bildet die Menge der möglichen Suchpfade eine Baumstruktur. Wegen der parallelen Propagierung der Aktivierungen in diese Inferenzzweige realisiert SHRUTI eine parallele Breitensuche. Kann in einem Inferenzschritt eine Anfrage durch das existierende Faktenwissen bewiesen werden, so wird der Collector des entsprechenden Prädikates aktiviert. Er liefert nun die Aktivierung in Vorwärtsrichtung an die Collectoren der gesamten Inferenzkette und die entsprechenden Anfragen werden als gültig markiert.

In Abbildung 4.5 wird der Inferenzvorgang für Beispiel 4.2.1 gezeigt. Für die Realisierung der Anfrage werden die Einheiten *Verkäufer* und *Marie*, sowie *kv-Objekt* und *Buch1* in

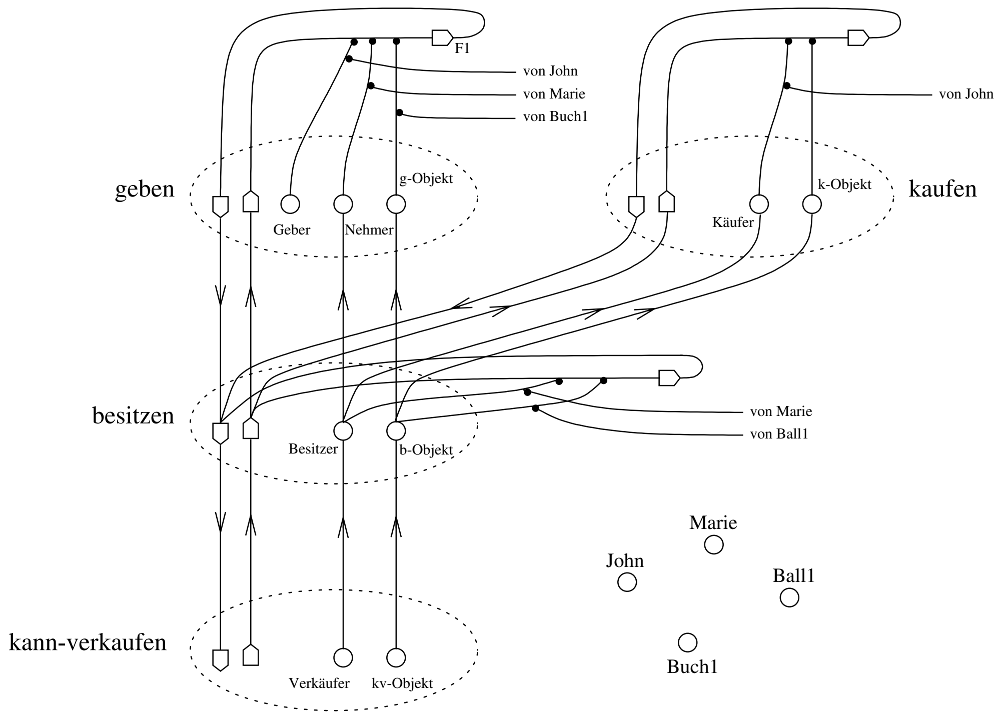


Abbildung 4.4: Das SHRUTI-Modell zum Rückwärtsschließen in der Wissensbank aus Beispiel 4.2.1

der jeweils gleichen Phase aktiviert. Der Enabler des Prädikates *kann-verkaufen* wird für einen ganzen Zyklus gesetzt. Alle Eingaben erfolgen in jedem Zyklus der Bearbeitung. Mit jedem Zyklus werden die gestellten Anfragen propagiert. So entsteht im zweiten Zyklus die dynamische Anfrage ‚*besitzen*(*Marie*, *Buch1*) ?' und im dritten die Anfragen ‚$\exists X$ *geben*(X, *Marie*, *Buch1*) ?' und ‚*kaufen*(*Marie*, *Buch1*) ?'. Mit dem Langzeitwissen darüber, daß *John Marie* das *Buch1* gegeben hat, feuert der Faktknoten $F1$. In den darauffolgenden Schritten der Inferenz werden nun die Collectoren von *geben*, *besitzen* und *kann-verkaufen* gesetzt und die zugehörigen Anfragen bewiesen. Das Ergebnis der Schlußfolgerungen ist, daß *Marie* sehr wohl in der Lage ist, *Buch1* zu verkaufen.

Im angegebenen Beispiel konnte ein Beweis für die gestellte Anfrage gefunden werden. Ist kein Faktenwissen vorhanden, um die Anfrage zu bestätigen, so kann die Inferenz nach einem endlichen Zeitraum, der der längsten möglichen Inferenzkette entspricht, abgebrochen werden. Das Resultat der Episode ist dann unter Annahme der Weltabgeschlossenheit (AWA, siehe [BHS93]), daß der Wahrheitswert der Anfrage falsch ist.

Für komplexere Klauseln, die der allgemeinen Form von Regelwissen in SHRUTI entsprechen, und den damit verbundenen Inferenzen werden folgende Festlegungen getroffen.

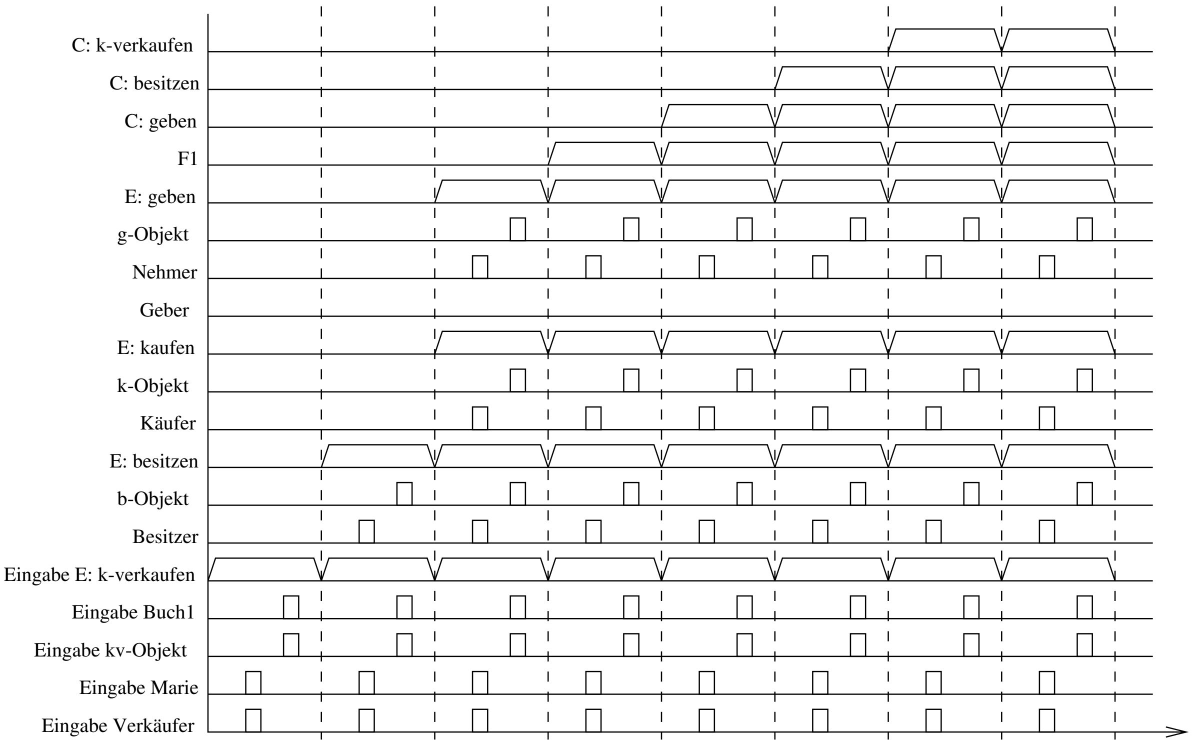


Abbildung 4.5: Der Inferenzprozeß in SHRUTI bei der Anfrage: *'kann-verkaufen(Marie, Buch1) ?'*

Bedingung 4.2.1 *Beim Rückwärtsschließen mit SHRUTI gelten folgende Annahmen:*

1. *Es gibt keine Funktionszeichen außer Konstanten.*
2. *Nur allquantifizierte Variablen dürfen als Argumente im Rumpf einer Regel vorkommen.*
3. *Alle in einem Fakt oder einer Anfrage vorkommenden Variablen treten nur einmal auf und sind existenzquantifiziert.*
4. *Tritt eine existenzquantifizierte Variable im Kopf einer Regel oder in einem Fakt auf, so darf sie nur mit Variablen unifiziert werden.*
5. *Variablen, die im Rumpf einer Regel mehrfach auftreten, müssen auch im Kopf der Regel vorkommen. Außerdem müssen sie gebunden werden, wenn der Kopf der Regel mit einer Anfrage unifiziert wird.*
6. *In einer Episode des Schließens tritt maximal eine Instanz jedes Prädikates auf.*

4.3 Vorwärtsschließen mit SHRUTI

Im Gegensatz zum Rückwärtsschließen existieren für das Vorwärtsschließen mit SHRUTI keine detaillierten Ausführungen. Einige grundlegende Unterschiede werden von Shastri und Ajjanagadde dargelegt, die genaue Logik eines solchen Modells bleibt jedoch undefiniert. Die Autoren begründen das damit, daß beide Systeme nach den gleichen Prinzipien arbeiten. Die Ergebnisse dieses Abschnittes beruhen daher auf eigenen Herleitungen mittels der Grundkonzepte von SHRUTI. Sie beziehen sich auf die Bedingungen, die durch die Art der Kodierung vorgegeben sind, und die für das Modell gestellten Annahmen. Außerdem wurden unklare Punkte durch persönliche Kontakte zur „Queensland University of Technology“ beseitigt. Dort existiert eine Implementation von SHRUTI für Rückwärts- und für Vorwärtsschließen.

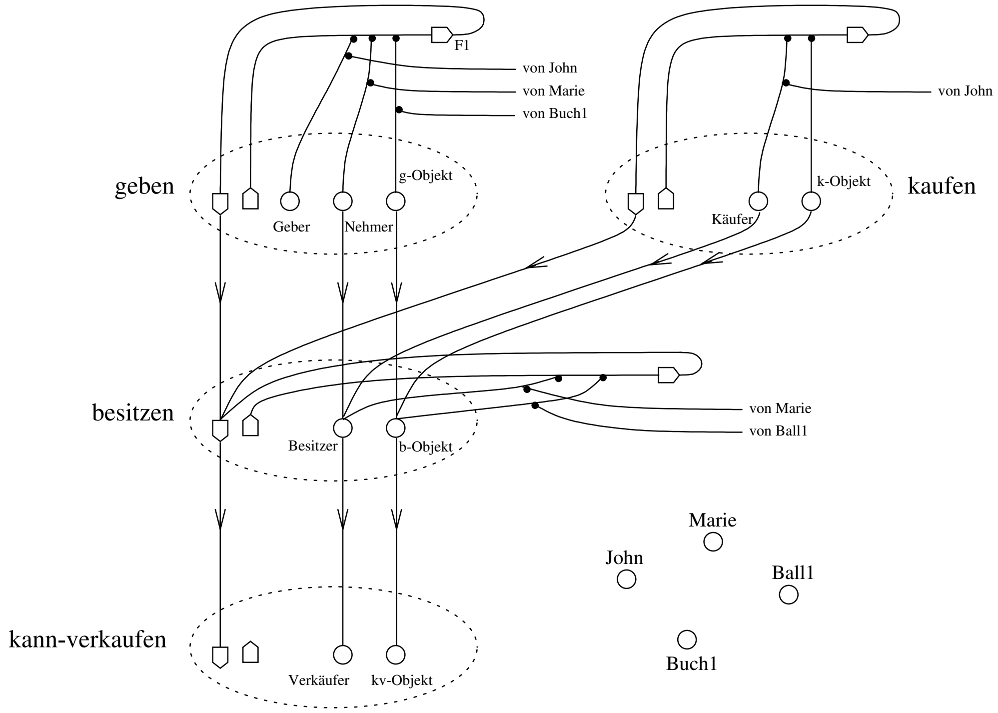


Abbildung 4.6: Ein vorwärtsschließender SHRUTI für das Beispiel 4.2.1.

Der offensichtlichste Unterschied bei einem vorwärtsschließenden SHRUTI liegt darin, daß keine Anfragen benötigt werden. Im allgemeinen wird eine Teilmenge des Faktenwissens aktiviert und mit den gegebenen Regeln darüber Schlußfolgerungen gezogen. Die Enabler-Einheiten werden deshalb auch nicht mehr für die Kodierung von Regeln benötigt. Sie verbleiben lediglich im Modell für die Aktivierung der im Inferenzprozeß zu nutzenden Fakten. Diese Auswahl aus der Gesamtmenge des kodierten Faktenwissens

funktioniert allerdings wie das Stellen einer Anfrage und ist deshalb trotzdem eine Art Rückwärtsschluß.
Weiterhin ändert sich die Richtung der Verbindungen zwischen den Argumenteinheiten von Prädikaten einer Regel. Argumente im Klauselkörper müssen nun zu den Argumentknoten des Klauselkopfprädikates propagiert werden. Die beschriebenen Unterschiede sollen an dem bereits bekannten Beispiel verdeutlicht werden.

Beispiel 4.3.1 *Betrachten wir erneut die Wissensbank aus Beispiel 4.2.1. Ziel des Inferenzvorganges ist zu erfahren, was aus der Gesamtmenge der Regeln und dem einzelnen Fakt geben(Marie, John, Buch1) geschlossen werden kann.*

Abbildung 4.6 zeigt das Modell eines vorwärtsschliessenden SHRUTI für das Beispiel und Abbildung 4.7 den zugehörigen Inferenzvorgang.

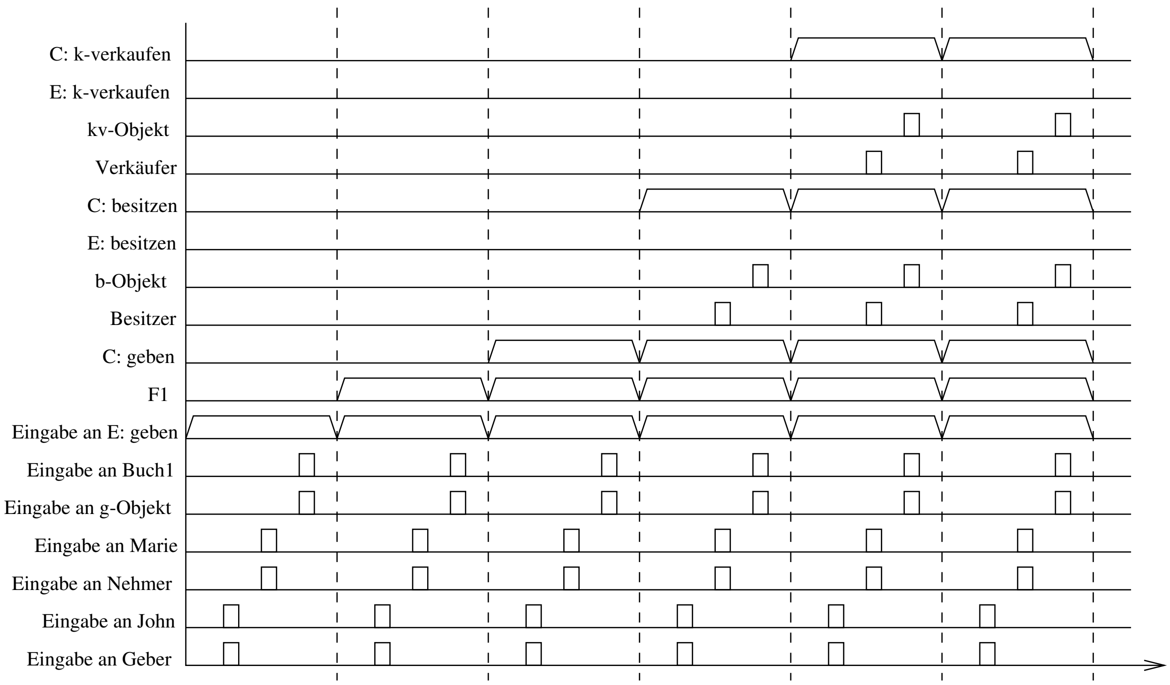


Abbildung 4.7: Der Aktivierungsverlauf beim Schließen im SHRUTI-Modell aus Abbildung 4.6 und Nutzung des Langzeitfaktes *geben*(*John*, *Marie*, *Buch1*).

Der Langzeitfakt *geben*(*John*, *Marie*, *Buch1*) wird aktiviert durch die zyklische Eingabe am Enabler von *geben* und die zyklische Bindung der Einheiten *Geber*, *Nehmer* und *g-Objekt* mit den Knoten *John*, *Marie* und *Buch1*. Nach zwei Inferenzschritten ist der Collector von *geben* aktiviert und die eigentliche Schlußkette beginnt. Mit dem Regelwissen wird jetzt die Instanz *besitzen*(*Marie*, *Buch1*) geschlossen und im letzten Schritt

entsteht die Formel *kann-verkaufen*(*Marie*, *Buch1*).
Für den Inferenzvorgang im allgemeinen gilt folgende Vorgehensweise. Die Enabler der in der Episode des Schließens zu nutzenden Langzeitfakten werden aktiviert und die Argumenteinheiten entsprechend gebunden. Weitere Prädikatsinstanzen, die nicht als Faktenwissen kodiert sind, können über die Aktivierung der korrespondierenden Collectoreinheiten natürlich auch erstellt werden. Der Inferenzvorgang beginnt, sobald Enabler- oder Collectoreinheiten eine Eingabe erhalten. Mit jedem Zyklus wird die Ausgabe der Netzeinheiten propagiert. Der Inferenzprozeß ist beendet, wenn das SHRUTI-Netz einen stabilen Zustand erreicht, daß heißt, wenn in zwei aufeinanderfolgenden Schritten keine neuen Prädikate geschlossen werden.
Betrachtet man unser Beispiel unter dem Gesichtspunkt, alle logischen Folgerungen aus dem gesamten Wissen erhalten zu wollen, so fällt auf, daß schon bei der Aktivierung von 2 der 3 Fakten der Wissensbasis die Forderung nach maximal einer Instanz pro Prädikat in der Episode des Schließens verletzt wird[5]. Bei Nutzung aller Fakten müßte so zum Beispiel dargestellt werden, daß *Marie* sowohl *Buch1*, als auch *Ball1* besitzt und daß *John* irgendetwas besitzt. Für die Darstellung all dieser Schlüsse steht jedoch nur einmal das Prädikat *besitzen* zur Verfügung. Die Ermittlung aller Folgerungen aus dem gesamten Wissen ist mit einer Inferenzepisode in SHRUTI hier nicht möglich und selbst das kann nur zur Laufzeit des Modelles dynamisch festgestellt werden.
Die Probleme, die durch mehrfache Instanzen von Prädikaten entstehen, behandelt eine Erweiterung von SHRUTI, die in Abschnitt 4.4.2 erläutert wird.

Nach der Betrachtung unseres einfachen Beispiels soll jetzt detailliert auf die Kodierung komplexer Regeln beim Vorwärtsschließen eingegangen werden. Dafür werden einige syntaktische Spezialfälle betrachtet. In diesem Zusammenhang werden auch die Grenzen deutlich, die durch die Kodierung an die Logik gesetzt sind.

Mehrfache Variablen:

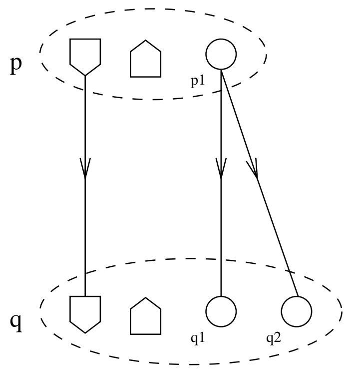


Abbildung 4.8: Die Kodierung der Regel $p(X) \rightarrow q(X, X)$

[5]vergleiche Annahme 6 aus Proposition 4.2.1.

In einem rückwärtsschließenden SHRUTI-System mußten Variablen, die mehrfach im Klauselkörper auftraten, auch im Kopf der Klausel enthalten sein und bei der Unifikation mit einer Anfrage gebunden werden. Analog dazu gilt beim Vorwärtsschließen, daß mehrfache Variablen im Klauselkopf auch im Klauselkörper enthalten sein und im Inferenzprozeß gebunden werden müssen. Abbildung 4.8 zeigt dafür ein Beispiel. Die Kodierung der Regel $p(X) \rightarrow q(X, X)$ erfolgt durch das Verbinden der Collectoren von p nach q und der Einheit des Argumentes p_1 zu den Argumenteinheiten q_1 und q_2.
Ist eine Instanz von p aktiv, so wird daraus eine Instanz von q geschlossen. Die Gleichheit der Argumente q_1 und q_2 kann jedoch nur garantiert werden, wenn das Argument p_1 tatsächlich gebunden ist. Stellt es eine existenzquantifizierte Variable dar und ist deshalb nicht aktiviert, so sind auch die Argumenteinheiten von q nicht aktiviert und repräsentieren deshalb keine gleichen Argumente. Aus einer Formel $\exists X\, p(X)$ entstünde somit eine Instanz $\exists Y, Z q(Y, Z)$, was logisch nicht korrekt wäre.
Das mehrfache Auftreten von Variablen im Klauselkörper ist ohne Einschränkung erlaubt. Die erforderliche Bindung zum Feuern der Regel kann mit den Netzeinheiten von SHRUTI evaluiert werden. In Abbildung 4.9 wird eine Regel dargestellt, die auch Variablen mehrfach im Körper enthält.
Eine Einschränkung, die in einem Rückwärtsschließer nicht existiert, betrifft alle allquantifizierten Variablen im Klauselkopf. Ist eine solche Variable nicht im Körper enthalten, so kann sie beim Feuern der Regel nicht gebunden werden und müßte eine Allquantifizierung darstellen. Das ist mit der Kodierung von SHRUTI nicht möglich. Deshalb gilt beim Vorwärtsschließen die zusätzliche Annahme, daß alle allquantifizierten Variablen, die im Klauselkopf auftreten auch im Klauselkörper auftreten müssen.

Konstanten und Existenzquantoren:

Konstanten dürfen beim Vorwärtsschließen nur im Körper der Regeln auftreten. Das liegt daran, daß die Konstanteneinheiten mit der Auswahl der Fakten aktiviert werden. Bei einer Konstante im Kopf einer Klausel könnte deshalb nicht garantiert werden, daß der zugehörige Netzknoten in dieser Episode überhaupt aktiviert ist. Im Falle des Rückwärtsschließens war dies genau umgekehrt, weil dort die Konstanteneinheiten mit einer Anfrage gesetzt wurden.
Existenzquantifizierte Variablen treten nach der Form der Regeln in SHRUTI nur im Klauselkopf auf. Das gilt für Vorwärts- und für Rückwärtsschließen. Sie dürfen jedoch nicht mehrfach auftreten, weil nicht vorhandene Aktivierungen natürlich auch nicht auf Gleichheit geprüft werden können. Mit den Mitteln der Kodierung wäre beim Vorwärtsschließen auch ein Auftreten im Klauselkörper innerhalb des Gültigkeitsbereiches der Allquantoren möglich. Wegen der Grundform der Regeln wird dies jedoch verboten.
Abbildung 4.9 zeigt die Kodierung der Regel $p(X, X, a) \rightarrow \exists Z q(X, Z)$. Bei dieser Regel treten Konstanten, existenzquantifizierte Variablen und mehrfache Variablen kombiniert auf.
Die Realisierung der existenzquantifizierten Variable Z ist am einfachsten. Ihr Argumentknoten q_2 erhält keine Verbindung und kann somit auch nicht gebunden werden.
Die Gleichheit der Argumente p_1 und p_2 wird geprüft durch das Verbinden zu der τ-oder-Einheit E_1 mit dem Schwellwert 2. Diese Einheit ist über den Zeitraum eines Zyklus nur

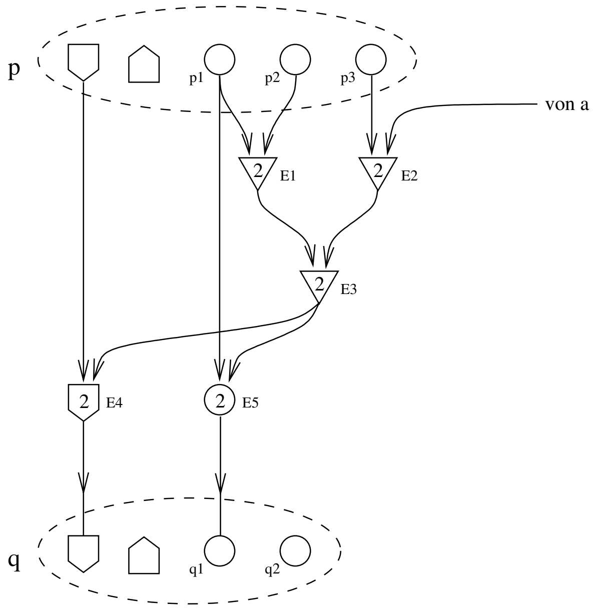


Abbildung 4.9: Die Kodierung von $p(X, X, a) \rightarrow \exists Z q(X, Z)$

dann aktiv, wenn beide Knoten gleich gebunden sind. Das Auftreten einer Konstante im Klauselkörper wird ähnlich realisiert. Alle Argumentknoten der Konstante werden zu einem τ-oder-Knoten verbunden, der auch eine Eingabe von der ρ-btu-Einheit der Konstante erhält. Im Beispiel entsteht so die Einheit E_2 mit den Verbindungen von p_3 und a. Ihr Schwellwert ist gleich der Anzahl des Auftretens von a plus 1. Sie gibt einen ununterbrochenen Pulsstrom aus, wenn alle notwendigen Argumente an die Konstante gebunden sind.
Das Feuern der Regel wird durch eine weitere τ-oder-Einheit (hier $E3$) geprüft. Sie verknüpft die Einheiten E_1 und E_2 konjunktiv und realisiert eine Art Schalter für die Propagierung der Ausgaben des Collectors und der Variable X. Dies geschieht über die Einheiten E_4 und E_5 mit dem Schwellwert 2.

Transitive Zusammenhänge:

Abbildung 4.10 zeigt die Kodierung einer Regel, bei der die verwendeten Variablen in einem transitiven Zusammenhang stehen. Es wird die Klausel $p(X, Y) \wedge q(Y, Z) \rightarrow r(X, Z)$ dargestellt. Die τ-oder-Einheit E_1 prüft das mehrfache Auftreten der Variablen Y. Eine Instanz von r wird genau dann abgeleitet, wenn jeweils eine Instanz von p und von q aktiv ist und E_1 feuert. Dies realisiert die τ-und-Einheit E_2 mit dem Schwellwert 3. Die ρ-btu-Knoten E_3 und E_4 können die Bindungen der Variablen X und Z nur an r weitergeben, wenn p_2 und q_1 gleich sind.
Transitive Zusammenhänge konnten in einem rückwärtsschließenden SHRUTI-System nicht realisiert werden. Dort bestand die Annahme, daß mehrfache Variablen im Klau-

selkörper auch im Klauselkopf auftreten müssen. Beim Vorwärtsschließen sind sie jedoch möglich, weil Variablen ohne Einschränkung mehrfach im Klauselkörper auftreten dürfen.

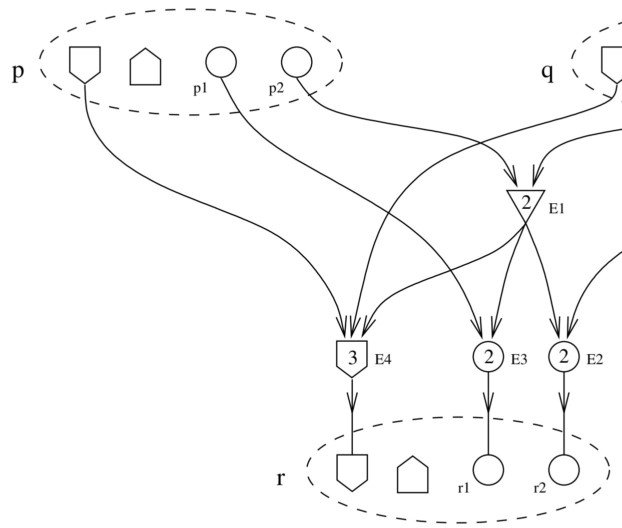


Abbildung 4.10: Die Darstellung von Transitivität bei $p(X, Y) \wedge q(Y, Z) \rightarrow r(X, Z)$

Die aufgeführten Varianten haben gezeigt, daß einige wichtige Unterschiede zum Rückwärtsschließen bestehen. Die Beschränkungen des Vorwärtsschließens sollen deshalb noch einmal zusammengefaßt werden.

Bedingung 4.3.1 *Beim Vorwärtsschließen mit SHRUTI gelten folgende Annahmen:*

1. *Es gibt keine Funktionszeichen außer Konstanten.*
2. *Nur allquantifizierte Variablen und Konstanten dürfen als Argumente im Rumpf einer Regel vorkommen.*
   Nur Variablen dürfen als Argumente im Kopf einer Regel vorkommen.
3. *Alle in einem Fakt vorkommenden Variablen treten nur einmal auf und sind existenzquantifiziert.*
4. *Allquantifizierte Variablen, die im Kopf einer Regel auftreten, müssen auch im Rumpf der Regel vorkommen.*
5. *Variablen, die im Kopf einer Regel mehrfach auftreten, müssen auch im Rumpf der Regel vorkommen. Sie müssen gebunden werden, wenn der Körper der Regel mit einem Fakt oder einer Ableitung aus Fakten unifiziert wird.*
6. *In einer Episode des Schließens tritt maximal eine Instanz jedes Prädikates auf.*

Anschaulich sieht die Syntax des Wissens damit folgendermaßen aus:

- **Regeln:**

$$\forall X_1, \ldots, X_m [p_1(\ldots) \wedge p_2(\ldots) \ldots \wedge p_n(\ldots) \rightarrow \exists Z_1, \ldots Z_l q(\ldots)]$$

  Die Argumente der Prädikate p_i sind Elemente der Menge $\{X_1, \ldots, X_m\}$ und der Konstanten. Die Argumente von q sind Elemente der Menge $\{Z_1, \ldots, Z_l\} \cup \{X_1, \ldots, X_m\}$

- **Fakten:**

$$\exists Z_1, \ldots, Z_l q(\ldots)$$

  Die Argumente von q sind Elemente der Menge $\{Z_1, \ldots, Z_l\}$ und der Menge der Konstanten.

4.4 Erweiterungen

Neben den Grundprinzipien gibt es in SHRUTI noch zwei wesentliche Erweiterungen. Zum einen betrifft das die Verwendung von Typen bzw. Sorten und zum anderen Probleme, bei denen zwangsläufig die Annahme 6 (siehe Propositionen 4.2.1 und 4.3.1) verletzt wird und im Rahmen einer Inferenzepisode mehrere Prädikatsinstanzen auftreten. Die Realisierungen dafür sollen an einer umgangssprachlichen Geschichte erläutert werden, die zweifelsohne vom Menschen reflexartig verstanden wird und somit zur Klasse des Reflexiven Schließens gehören sollte.

„Klaus und sein Sohn John spielten Schach. Da kam John's Schwester Marie herein und fragte ihren Vater: ‚...'“

Betrachten wir nun das Problem, aus den Beziehungen zwischen den Personen herauszufinden, wer der *Vater* von *Marie* ist. Das zugehörige Fakten- und Regelwissen könnte formal wie folgt aussehen[6]:

Beispiel 4.4.1

$$\begin{aligned} &\rightarrow \quad Sohn(John, Klaus) \\ &\rightarrow \quad Schwester(Marie, John) \\ \forall X, \forall Y\!:\!Mann\ Sohn(X,Y) &\rightarrow \quad Vater(Y,X) \\ \forall X,Y,Z\ Vater(X,Y) \wedge Schwester(Z,Y) &\rightarrow \quad Vater(X,Z) \end{aligned}$$

4.4.1 Typhierarchie

Die Realisierung von Sorten erfolgt mit einer speziellen Architektur, der sogenannten Typhierarchie. Diese Architektur ist eine Baumstruktur von ρ-btu-Einheiten, in der alle

[6]Wegen der Benutzung von Sorten werden hier die Allquantoren mit aufgeführt.

Konstanten und Typen repräsentiert sind. In den obersten Schichten der Typhierarchie befinden sich die Netzknoten, die Sorten darstellen. Alle Netzeinheiten aus einer unteren Schicht, die mit den Sorten verbunden sind, stehen zu ihnen in einer *ist-ein*-Relation. Die Konstanten der Wissensbasis finden sich deshalb als Blätter der Baumstruktur wieder. Alle *ist-ein*-Verbindungen sind bidirektionale Verbindungen. Sie werden über einen speziellen Kontrollmechanismus gesteuert, der in [SA93] nachzulesen ist.
Mit der Kodierung einer separaten Typhierarchie realisiert SHRUTI eine strikte Trennung von Typen und Prädikaten, obwohl derartige *ist-ein*-Beziehungen auch mit Prädikatssymbolen darstellbar wären. Die Typhierarchie ist für vorwärts- und rückwärtsschließende Systeme gleich.

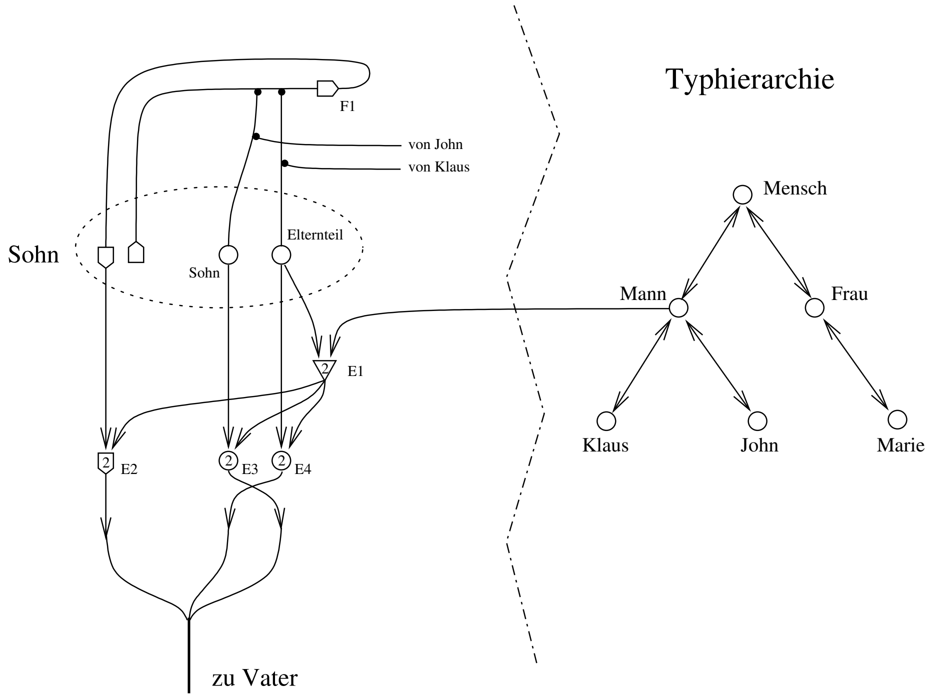


Abbildung 4.11: Die Regel $\forall X, \forall Y{:}Mann\ Sohn(X,Y) \rightarrow Vater(Y,X)$ aus Beispiel 4.4.1 in einem vorwärtsschließenden SHRUTI mit einer Typhierarchie.

In der Wissensbasis zu unserer „Vater“-Geschichte enthält die erste Regel die Sorte *Mann*. Aus einer Beziehung *Sohn-Elternteil* kann nur dann geschlossen werden, daß *Elternteil* der *Vater* von *Sohn* ist, wenn *Elternteil* ein *Mann* ist. Die Implementierung der Regel beim Vorwärtsschließen zeigt Abbildung 4.11.
Die Hierarchie repräsentiert die Bindungen *ist-ein*$(Klaus, Mann)$, *ist-ein*$(John, Mann)$, *ist-ein*$(Marie, Frau)$, *ist-ein*$(Mann, Mensch)$ und *ist-ein*$(Frau, Mensch)$. Die τ-oder-Einheit E_1 der Regel prüft, ob der gebundene *Elternteil* auch tatsächlich ein Mann ist. Sie fungiert wiederum als Schalter für das Feuern der Regel. Im Inferenzprozess bedeutet

dies, daß bei einer Aktivierung von Klaus in der Typhierarchie eine gleichzeitige synchrone Aktivierung der Knoten *Mann* und *Mensch* auftritt. Ist im Inferenzprozess also tatsächlich der Fakt F_1 aktiviert, so feuern *Elternteil* und *Mann* synchron und die Formel $Vater(Klaus, John)$ kann abgeleitet werden.

4.4.2 Multiple Instanzen

Die zweite Regel in der Wissensbank unseres Beispiels enthält das Prädikat *Vater* sowohl im Klauselkörper, als auch im Klauselkopf. Dabei treten als Terme unterschiedliche Bindungen auf, so daß zu vermuten ist, daß mehrere Instanzen des Prädikates im Laufe des Inferenzprozesses entstehen. Bisher sind wir beim Rückwärtsschließen und beim Vorwärtsschließen immer davon ausgegangen, daß in einer Episode der Inferenz jedes Prädikat nur maximal einmal auftritt. Hier wird diese Annahme vermutlich verletzt werden, und es können mehrfache (multiple) Instanzen von *Vater* abgeleitet werden. Realisieren wir die Regel trotzdem wie bisher, dann müssen mehrere Instanzen auf einem Speicherplatz für *Vater* dargestellt werden. Dabei kann der sogenannte CROSS-TALK auftreten.

Definition 4.4.1 *Als* CROSS-TALK *bezeichnet man die Tupel, welche durch die Überlagerung der Elemente einer Relation repräsentiert werden, obwohl sie selbst nicht in der Relation enthalten sind.*

Anders gesagt heißt das, daß Instanzen dargestellt werden, die eigentlich gar nicht dargestellt werden sollen. Abbildung 4.12 zeigt die Problematik an den Prädikatssymbolen p und r und jeweils einer Belegung der korrespondierenden SHRUTI-Einheiten.
Bei dem einstelligen Prädikat p kann aus den aktivierten Phasen eindeutig die Menge der Instanzen erkannt werden. Bei der Überlagerung von Kopien im zweistelligen Prädikat r bildet die Menge der Aktivierungen der Knoten ein Kreuzprodukt, wodurch die Menge $\{r(a,a), r(a,b), r(b,a), r(b,b)\}$ repräsentiert wird. Der Phasenverlauf kann jedoch auch durch die Überlagerung von $r(a,a)$ und $r(b,b)$ entstanden sein. In diesem Fall wären die Formeln $r(a,b)$ und $r(b,a)$ der CROSS-TALK und inkorrekt, was jedoch aus dem Verlauf selbst nicht erkannt werden kann. Die Abbildung verdeutlicht, daß nur bei einstelligen Prädikaten dieses Problem keine Rolle spielt.

Shastri und Ajjanagadde verweisen auf einen Mechanismus ([MS92a], [MS92b]), der CROSS-TALK während des Schließens verhindern soll. Der Grundgedanke ist, die Überlagerung mehrerer Prädikatsinstanzen zu umgehen. Dazu wird für jedes Prädikat eine endliche Anzahl k von Speicherplätzen bereitgestellt, und die im Inferenzprozeß auftretenden Prädikatsinstanzen werden von einem Schalter darauf verteilt. Ein solcher MI-Schalter wird für jedes Prädikat benötigt, für das mehrere Kopien vorhanden sind. Alle Regeln, die eine Instanz eines Prädikates dynamisch erzeugen können haben eine Verbindung zu dem Schalter dieses Prädikates. Dort werden die eingehenden Instanzen auf Gleichheit geprüft und jeder Instanz genau ein Speicherplatz zugewiesen.
Der Schaltermechanismus wird ebenfalls nur durch die in 4.1.2 beschriebenen Netzein-

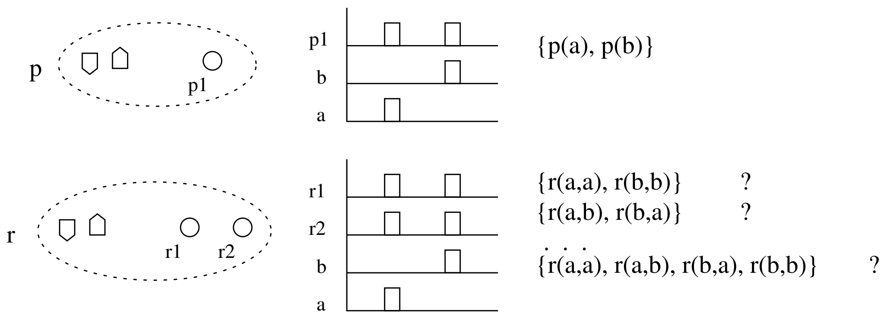


Abbildung 4.12: Das CROSS-TALK-Problem in SHRUTI.

heiten realisiert.

Kommen wir auf unser Beispiel zurück. Lediglich das Prädikat *Vater* kann während einer Episode der Inferenz mehrfach auftreten. Wir stellen deshalb $k = 3$ Kopien für das Prädikat bereit. Für die anderen Prädikatssymbole ist eine Instanz ausreichend. Abbildung 4.13 zeigt die Kodierung der zweiten Regel aus Beispiel 4.4.1 in einem vorwärtsschliessenden SHRUTI-System.

Da jeder Speicherplatz genau eine Instanz darstellt, muß jede der 3 möglichen Instanzen von *Vater* nun in Kombination mit jeder Instanz des Prädikats *Schwester* auf das Feuern der Regel getestet werden. Demzufolge werden 3 Kopien der Regel angelegt. Die Regeln werden in der Abbildung als gestrichelte Boxen dargestellt. In ihrem Innern ist eine Kodierung entsprechend den Mechanismen nach Abschnitt 4.3. Sie wird aus Gründen der Übersichtlichkeit nicht dargestellt, ist aber in den Kopien jeweils gleich.

Kann in einer der Kombinationen von *Vater* und *Schwester* eine weitere Instanz von *Vater* abgeleitet werden, dann propagiert die aktivierte Regel die geschlossene Formel an den MI-Schalter des Prädikates *Vater* und die neue Instanz wird dem nächsten freien Speicherplatz zugewiesen, sofern sie nicht schon vorhanden ist.

Die Abbildung zeigt, daß jede Kopie einer Regel immer nur mit genau einer Instanz für jedes Prädikat verbunden ist. Dies ist auch beim Rückwärtsschließen so und führt uns zu einer Veränderung der bisherigen Annahmen in SHRUTI.

Bedingung 4.4.1 *Durch die Nutzung mehrfacher Prädikatsinstanzen in SHRUTI wird die Annahme 6 im Rückwärts- und im Vorwärtsschließen[7] ungültig. Es gelten nun folgenden ergänzende Bedingungen*

6. *Eine Regel darf nur beschränkt oft kopiert werden.*

7. *Jede Kopie einer Regel ist mit maximal einer Instanz jedes Prädikates im Klauselkopf beim Rückwärtsschließen und im Klauselkörper beim Vorwärtsschließen verbunden.*

[7] siehe Bedingungen 4.2.1 und 4.3.1

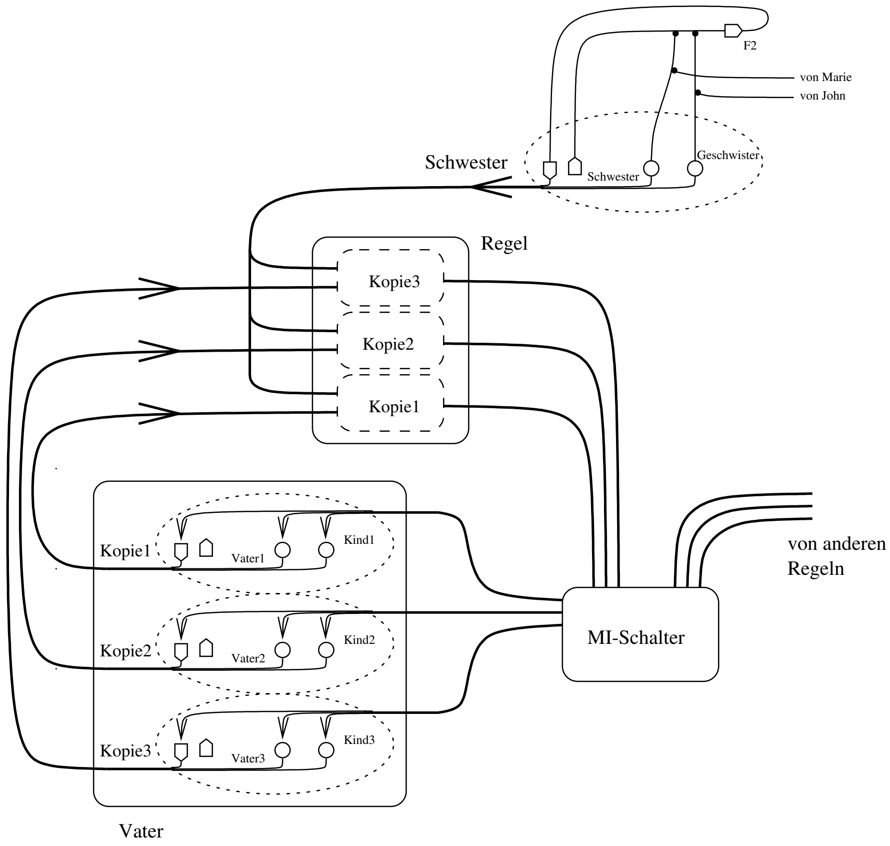


Abbildung 4.13: Die Darstellung von $Vater(X, Y) \wedge Schwester(Z, Y) \rightarrow Vater(X, Z)$ in einem vorwärtsschließenden SHRUTI-System.

Damit besteht überhaupt erst die Möglichkeit, mit mehreren Instanzen eines Prädikates gleichzeitig zu schließen, und Regeln mit mehrfachem Auftreten eines Prädikatssymboles zu implementieren. Dies ist in der Grundversion nicht realisierbar.
Die maximale Anzahl an Instanzen eines Prädikates legen Shastri und Ajjanagadde auf $k = 7 \pm 2$ fest. Dieser Wert ist psychologisch begründet und stellt die maximale Anzahl von Objekten dar, die beim reflexiven Schließen gespeichert werden können. Aus Sicht der Logikkalküle findet man deshalb eine Vielzahl Probleme, die trotz der mehrfachen Instanzen nicht in SHRUTI lösbar sind, weil die Anzahl k der Kopien nicht ausreichend ist. Dann werden im Laufe der Inferenzkette Instanzen zwangsläufig überlagert, und es kann CROSS-TALK entstehen. Dies widerspricht jedoch nicht den Anforderungen des Modelles, weil derartige Probleme nach Auffassung der Autoren nicht mehr zur Klasse der reflexartig lösbaren gehören.

4.5 Aufwandsabschätzung und Diskussion

Die Platzkomplexität von SHRUTI entspricht in der Grundversion genau den Ansprüchen, die die Autoren an die Klasse des reflexiven Schließens[8] stellen. Für die Kodierung der Konstanten und Fakten wird jeweils nur eine Netzeinheit benötigt. Der Aufwand zur Kodierung aller Prädikate mit der durchschnittlichen Stelligkeit $\bar{s}$ ist begrenzt durch $O(\bar{s} \times D)$, wenn D die Anzahl der Prädikate repräsentiert. Für jede Regel wird eine Kodierung vorgenommen, die linear abhängig ist von den in ihr enthaltenen Variablen und Konstanten. Beim Vorwärtsschließen ist sie sogar noch abhängig von der mittleren Stelligkeit der Prädikate. Im schlechtesten Fall werden so $O((\bar{v} + \bar{s}) \times K)$ Netzeinheiten benötigt, wobei $\bar{v}$ die mittlere Anzahl Variablen und Konstanten in den K Klauseln der Wissensbasis ist. Bezeichnet man weiterhin die Anzahl der Konstanten mit C, so hat SHRUTI eine worst-case-Komplexität bezüglich der Netzeinheiten von $O(C + \bar{s} \times D + (\bar{v} + \bar{s}) \times K)$. Die Platzkomplexität ist damit sowohl beim Rückwärts-, als auch beim Vorwärtsschließen linear abhängig zur Größe der Wissensbasis.
Bei den Erweiterungen in SHRUTI hat die Implementierung mehrfacher Prädikatsinstanzen entscheidenden Einfluß auf die Platzkomplexität. Shastri und Ajjanagadde empfehlen, im allgemeinen Fall die Instanziierungskonstante k für alle Prädikate gleich zu wählen. Damit erhöht sich die Anzahl der Einheiten zur Repräsentation der Prädikate genau um diesen Faktor k. Die Kodierung der Regeln wird unterschiedlich beeinflußt. Beim Rückwärtsschließen wird eine um k^2 höhere Anzahl Netzeinheiten benötigt. Der Platzaufwand ist damit quadratisch abhängig von der Anzahl verwendeter Instanzen. Wesentlich einschneidender ist die Veränderung in einem vorwärtsschließendem SHRUTI-System. Dort müssen zur Implementierung von Regeln mit mehreren Prädikaten im Klauselkörper alle k Instanzen dieser Prädikate jeweils kombinatorisch verknüpft werden. Sei m die Anzahl der Prädikatszeichen im Körper einer Klausel, dann werden effektiv $O(k^m)$ Kopien der Regel kodiert. Die Platzkomplexität beim Vorwärtsschließen mit multiplen Instanzen ist damit exponentiell.
Mani und Shastri diskutieren in [MS92a] und [MS92b] Einschränkungen und Transformationen, die speziell dieses Problem auf einen vergleichbaren Aufwand wie beim Rückwärtsschließen reduzieren sollen. Sie gehen davon aus, daß der Wert m typischerweise bei 2 liegt und der durchschnittliche Wert von k gleich 3 ist, oder andernfalls Umformungen der Wissensbasis durchgeführt werden, die dazu führen. Gelänge dies, hätte man einen Mehraufwand von ($\approx 3^2$), was dem Mehraufwand eines Rückwärtsschließers mit $k = 3$ entspräche. Die von den Autoren aufgeführten Bedingungen schränken aber die verwendbare Logik weiter ein, oder führen zu einer Vervielfachung der Anzahl an Regeln, so daß letztendlich die Platzkomplexität im allgemeinen Fall exponentiell zur Größe der Wissensbasis ist.

Der Zeitaufwand in einem SHRUTI-Modell der Grundform ist parallel linear zur Länge der Inferenzkette. Dies gilt gleichermaßen für Rückwärtsschließen und Vorwärtsschließen. Allerdings sollte man vorsichtig mit der Behauptung umgehen, daß der Aufwand unabhängig von der Größe der Wissensbasis ist. Man kann zeigen, daß im worst-case

[8] siehe Abschnitt 4.1.1

die Länge der Inferenzkette gleich der Anzahl der Klauseln ist. Ein Beispiel dafür ist ein Programm der Form $\{p_1(a); p_1(X) \rightarrow p_2(X); \ldots; p_{5000}(X) \rightarrow p_{5001}(X)\}$. Damit wäre eine Abhängigkeit des Zeitaufwandes zur Größe der Wissensbasis gegeben. Mit der Betrachtung des average-case und der Annahme, daß der Langzeitspeicher enorm groß ist, tritt ein solcher Fall jedoch praktisch nie ein.
Der zusätzliche Zeitaufwand bei Nutzung der SHRUTI-Erweiterungen ist bei den Inferenzmethoden Vorwärts und Rückwärts ebenfalls gleich. Die Typhierarchie benötigt bei einer Anfrage l_2 Inferenzschritte mehr, wobei l_2 die Länge der kürzesten Ableitung in der Typhierarchie ist. Der Mehraufwand bei multiplen Instanzen wird durch den Aufbau der MI-Schalter hervorgerufen und besteht im Faktor k. Mit den Erweiterungen hat SHRUTI deshalb ebenfalls parallel lineare Zeitkomplexität.

Die dargelegten Komplexitäten werden in Tabelle 4.1 zusammengefaßt.

		Platzkomplexität	*Zeitkomplexität*
Rückwärts	Grundversion	linear	parallel linear
	Erweiterungen	quadratisch	parallel linear mit Faktor k
Vorwärts	Grundversion	linear	parallel linear
	Erweiterungen	exponentiell	parallel linear mit Faktor k

Tabelle 4.1: Übersicht über die Komplexität von SHRUTI.

Beim Vergleich der Inferenzmethoden Rückwärtsschließen und Vorwärtsschließen fällt insbesondere der Platzbedarf bei Verwendung multipler Instanzen auf. Am Beispiel 4.3.1 haben wir jedoch schon feststellen müssen, daß diese Erweiterung gerade beim Vorwärtsschließen nahezu unumgänglich ist. Die Aktivierung aller Fakten führt nämlich in den meisten Fällen zur Ableitung multipler Instanzen. Ein Beweis ohne die entsprechende Erweiterung müßte beschränkt werden auf die Nutzung eines Bruchteils des vorhandenen Faktenwissens. Vollständige Schlüsse sind damit nicht möglich. Benutzt man aus diesen Gründen den Mechanismus für mehrfache Instanzen, dann widerspricht der exponentielle Platzaufwand eindeutig den Anforderungen an reflexives Schließen. Mani und Shastri argumentieren zwar, daß die Komplexität linear zur Größe der Wissensbasis ist und lediglich ein Proportionalitätsfaktor existiert, der mit speziellen Transformationen der Wissensbasis auf $\approx 3^2$ reduziert werden kann, aber das ist nicht plausibel für eine Vielzahl allgemeiner Probleme, die durchaus reflexartig vom Menschen gelöst werden

können. Für den allgemeinen Fall bleibt der Aufwand exponentiell.
Anders beim Rückwärtsschließen. Dort ist der Platzaufwand zur Kodierung der Regeln selbst bei mehrfachen Prädikatsinstanzen nur quadratisch. In vielen Beispielen ist außerdem nicht einmal das notwendig, weil in einer Episode stets alle Fakten zum Beweis genutzt werden können, selbst wenn sie über dem gleichen Prädikatssymbol gelten. Dies ist dazu völlig unabhängig von der zu beweisenden Anfrage.
Die Diskussion führt zu der Annahme, daß die Entscheidung von Shastri und Ajjanagadde zur Darstellung des Rückwärtsschließens wahrscheinlich doch nicht ganz willkürlich war. Dafür sprechen auch noch andere Punkte.
Rückwärtsschließen in SHRUTI bedeutet, eine atomare Anfrage auf ihre Bindung zu prüfen und entsprechend auf mehrere Symbole zu propagieren. Beim Vorwärtsschließen muß dagegen aus einer Menge von Prädikatsinstanzen eine einzelne atomare Formel geschlossen werden. Dazu ist eine Unifikation über dieser Menge notwendig. Interessant ist in diesem Zusammenhang, daß trotzdem für die Implementierung der Regeln keine verbietenden Verbindungen und auch keine τ-oder-ähnlich-Einheiten notwendig sind. Alle Zusammenhänge können mit den 3 anderen Netztypen dargestellt werden.

Die Anzahl der Netzeinheiten scheint generell diskussionswürdig zu sein. Eine solche Vielzahl von verschiedenen Funktionen ist äußerst ungünstig aus Sicht eines einfachen Modelles. Die Übersichtlichkeit, speziell beim Auftreten mehrerer Instanzen von Prädikaten, geht verloren. Außerdem bleibt die Fähigkeit des Lernens in einem SHRUTI-System relativ unklar. Ein detaillierter Algorithmus für derartige Vorgänge wird auch von den Autoren nicht beschrieben. Allerdings spricht die Schnelligkeit des Systems für sich.
Die Darstellung von Sorten in einer gesonderten Architektur ist unverständlich, wenn man betrachtet, daß die Sortensymbole genausogut als Prädikate repräsentiert werden könnten.

Ähnliche Kritiken übt auch Ajjanagadde ([Ajj94]) selbst an SHRUTI. Er legt die Problematik an den 3 Unterscheidungen dar, die grundlegend für das Modell sind: die Unterscheidung zwischen Argumenten und Füllern, die zwischen Typen und Prädikaten und die Trennung von Regeln und Fakten. Er gibt in dem Zusammenhang ein Modell an, welches lediglich mit einem Einheitentyp und den verbietenden Verbindungen auskommt, außerdem die genannten Unterscheidungen mißachtet und dafür 3 verschiedene Aktivierungsstufen benutzt. Allerdings ist dieses Modell in seiner Arbeit so ungenau beschrieben, daß einige Zweifel aufgetreten sind, ob es überhaupt in der beschriebenen Weise funktioniert.

In der Gesamtheit betrachtet ist SHRUTI eine äußerst interessante Methode zum Rückwärtsschließen in einer speziellen Hornklausellogik. Seine Stärken liegen in der Schnelligkeit des Modells und in der Plausibilität der Konzepte. Außerdem ist die Erklärung logisch aufgebaut, daß nur eine Klasse von Problemen reflexartig gelöst werden kann. Wenn sich das tatsächlich für eine formale Logik bestätigen lassen könnte, hätte man eine Erklärung, warum automatisierte Inferenzmethoden für viele Fälle so viel Zeit benötigen. Fest steht aber auch, daß mit SHRUTI einige Probleme ungelöst bleiben müssen.

5. Adäquates Problemlösen durch Reduktionen

Eine Vielzahl von Kalkülen sind zwar in der Lage korrekte und vollständige Beweise für Zusammenhänge zwischen formalem Wissen zu finden, benötigen dafür aber unverhältnismäßig viel Zeit. Das gilt auch für Probleme, die vom Menschen enorm schnell und scheinbar ohne Anstrengungen gelöst werden können. Dieser Widerspruch, der als das Paradoxon der Künstlichen Intelligenz bezeichnet wird, ist durch die Betrachtung von reflexivem Schließen im Sinne von SHRUTI etwas kleiner geworden. Aus der theorethischen Sicht der Kalküle hat Bibel mit seiner Annahme zu automatisierten Deduktionsverfahren in [Bib91] das Ziel für Theorembeweismethoden abgesteckt, nämlich adäquate Beweismethoden zu finden.

„Es gibt eine adäquate allgemeine Beweismethode, welche jeden von einem Menschen durchführbaren Beweis automatisiert finden kann, vorrausgesetzt das Problem wird in angemessenen formalisierten Termen gestellt.“

Unter Adäquatheit versteht Bibel dabei die Verhältnismäßigkeit des Aufwandes zum Lösen eines Problems.

„Adäquatheit ist die Eigenschaft einer Beweismethode, einfachere Probleme schneller zu lösen, als schwierigere Probleme.“

Unter dem Gesichtspunkt der Adäquatheit eines Beweises sollen in diesem Kapitel Transformationen mit linearem Zeitaufwand betrachtet werden, die Reduktionen der Konnektionsmethode. Im zweiten Abschnitt wird die Beziehung von SHRUTI zu diesen Vereinfachungen untersucht. In 5.3 wird abschließend ein vorwärtsschließender Kalkül definiert, der nur mit Reduktionen arbeitet.

5.1 Die Konnektionsmethode

Eine Inferenzmethode zum Beweisen der Gültigkeit von Formeln ist die Konnektionsmethode von Bibel ([Bib87b]). Dabei werden Formeln in Matrizenform so dargestellt, daß die einzelnen Klauseln in Spalten nebeneinander angeordnet werden. Für die Beweisführung werden neben der Matrix einer Formel noch sogenannte Konnektionen verwendet. *Konnektionen* sind Kanten zwischen Literalen mit gleichem Prädikatssymbol, aber unterschiedlichem Vorzeichen. Solche Literale werden als zueinander *komplementär* bezeichnet. Ein *Pfad* durch eine Matrix ist eine beliebige Folge von jeweils einem Literal jeder Klausel bzw. Spalte. Eine Menge M von Konnektionen in einer Matrix wird *aufspannend* genannt, wenn jeder mögliche Pfad durch die Matrix mindestens eine Konnektion aus der Menge M enthält. Eine Menge M von Konnektionen stellt einen *Konnektionsbeweis* dar, wenn sie eine aufspannende Menge ist, und alle Terme, der durch die Konnektionen aus M verbundenen Literale, mit einer Substitution θ simultan unifiziert werden können. Wegen der Forderung zur simultanen Unifizierung ist besonders die Menge der minimalen aufspannenden Konnektionen interessant. Eine aufspannende Menge M von Konnektionen ist *minimal*, wenn es keine aufspannende Menge M' von Konnektionen gibt mit $M' \subset M$.

In den folgenden Ausführungen wird die Konnektionsmethode an Formeln in Normalform demonstriert. Sie ist jedoch auch für Nichtnormalform-Formeln definiert. Die Transformation prädikatenlogischer Formeln in Matrixnormalform wird in [Bib87b] beschrieben.

Beispiel 5.1.1 *Betrachten wir eine Matrix in Normalform, die eine Menge M aufspannender Konnektionen besitzt.*

$$\begin{array}{cccc} & & p(X) & r(a,Z) \\ & & q(X,Y) & \\ \neg p(a) & \neg q(X,Y) & \neg r(Y,b) & \end{array}$$

M ist aufspannend, weil jeder der 3 möglichen Pfade $(\neg p(a), \neg q(X,Y), p(X), r(a,Z))$, $(\neg p(a), \neg q(X,Y), q(X,Y), r(a,Z))$ und $(\neg p(a), \neg q(X,Y), \neg r(Y,b), r(a,Z))$ mindestens eines der Paare $(\neg p(a), p(X))$, $(\neg q(X,Y), q(X,Y))$ oder $(\neg r(Y,b), r(a,Z))$ enthält. Die Menge ist gleichzeitig minimal, weil es keine echte Teilmenge an Konnektionen von M gibt, die ebenfalls aufspannend ist.
Mit der Substitution $\theta = \{X \mapsto a, Y \mapsto a, Z \mapsto b\}$ sind die konnektierten Terme simultan unifizierbar. Damit ist ein Beweis für die Matrix gefunden worden. Das bedeutet, daß die Formel in disjunktiver Normalform

$$\exists X,Y,Z[\neg p(a) \vee \neg q(X,Y) \vee (p(X) \wedge q(X,Y) \wedge \neg r(Y,b)) \vee r(a,Z)]$$

allgemeingültig ist. Äquivalent dazu bedeutet es für die Formel

$$\forall X,Y,Z[p(a) \wedge q(X,Y) \wedge (\neg p(X) \vee \neg q(X,Y) \vee r(Y,b)) \wedge \neg r(a,Z)]$$

in konjunktiver Normalform, daß sie unerfüllbar ist.

Das Finden eines Beweises in der Konnektionsmethode erfolgt durch eine systhematische Suche in der Menge aller möglichen Pfade durch die Matrix. Sie beginnt mit der Auswahl einer Startklausel. In der ausgewählten Spalte wird dann für ein Literal eine Konnektion gesucht und entsprechend dieser Konnektion die Menge der Pfade in disjunkte Mengen zerlegt. Die Mengen von Pfaden, die noch keine Konnektion enthalten, werden anschließend in derselben Weise weiterbearbeitet. Während dieser Abarbeitung wird dynamisch überprüft, ob die ausgewählten Konnektionen auch durch eine Substitution unifizierbar sind. Die Methode ist erfolgreich, wenn unter dieser Bedingung die Restmenge der konnektionslosen Pfade leer und damit die Menge der Konnektionen aufspannend ist. Ein Beweis für die Gültigkeit einer Formel wird aber auch dadurch erbracht, daß Klauseln der Wissensbasis beliebig oft kopiert werden, und die so erweiterte Matrix eine aufspannende Menge von Konnektionen hat.

Proposition 5.1.1 *Auf der Konnektionsmethode basierende Kalküle sind korrekt und vollständig.*

Der Aufwand für die Suche bei der Konnektionsmethode ist stark abhängig von der Auswahl der Startklausel und der Auswahl der Konnektionen. So kann es durchaus nötig sein, Backtracking-Schritte auszuführen, weil die gewählten Konnektionen keine simultane Unifizierung zulassen. Dementsprechend können keine genauen Angaben zum Zeitaufwand gemacht werden, im schlechtesten Fall ist er jedoch exponentiell zur Anzahl der Literale in der Formel.
Höchste Priorität vor dem Ermitteln einer aufspannenden Menge hat deshalb die Vereinfachung der Matrix mit Methoden, die polynomiellen Zeitaufwand, möglichst sogar quadratischen oder linearen Bedarf haben. Solche Transformationen sollen das Problem „reduzieren“ und werden deshalb als Reduktionen bezeichnet. Reduktionen werden in [Bib87b] und [Bib92] betrachtet und sollen nach Bibel folgenden Anforderungen genügen:

- Reduktionen überführen eine Formel in ein echt kleineres Problem.
- Reduktionen sind gültigkeitserhaltend.
- Reduktionen sind schnell auf ihre Anwendbarkeit testbar.
- Reduktionen sind schnell durchführbar.

Die erste Eigenschaft drückt aus, daß nach Anwendung einer Reduktion der verbleibende Aufwand zum Lösen des Problems geringer ist. Bei vielen Reduktionen macht sich dies durch die Abnahme der Literale in der Matrix bemerkbar. Eine weitere Möglichkeit ist, daß eine Substitution durchgeführt wird, die zwingend für den Beweis der Formel ist.
Ist die so erhaltene Matrix gültig, dann muß auch die nichtreduzierte Formel gültig sein[1]. Dies sagt die zweite Eigenschaft über Reduktionen aus. Die letzten beiden Anforderungen beschreiben den Zeitaufwand. Schnell bedeutet in diesem Zusammenhang, daß er für Erkennen und Ausführen nicht schlechter als polynomial zur Größe der Matrix sein darf.

Für die weiteren Betrachtungen wollen wir eine Auswahl von Reduktionen vorstellen.

[1] Die Umkehrung ist nicht zwingend erforderlich.

- **MULT**: Ein Literal, das in einer Klausel mehrfach auftritt, bezeichnet man als *multiples* Literal. Existiert ein solches Literal, dann können alle Auftreten bis auf eines aus der Klausel entfernt werden.
- **PURE**: Enthält eine Klausel ein Literal für das in der gesamten Matrix kein komplementäres Literal auftritt, so kann die Klausel aus der Matrix entfernt werden.
- **TAUT**: Tritt in einer Klausel ein Literal und sein Komplement auf, dann ist die Klausel eine *Tautologie*. Sie kann aus der Matrix entfernt werden.
- **ISOL**: Wenn die Literale einer Konnektion an keiner anderen Konnektion beteiligt sind, die Literale einer Konnektion Grundliterale sind, oder eines der Literale Grundliteral ist und das andere an keiner weiteren Konnektion beteiligt ist, dann bezeichnen wir die Konnektion als *isoliert*. Sind die Literale einer isolierten Konnektion unifizierbar, dann können die beteiligten Klauseln durch ihre Resolvente ersetzt oder andernfalls die Konnektion entfernt werden.
- **PICS**: Sei eine Konnektion zweier Literale $p(s_1, \ldots, s_n)$ und $\neg p(t_1, \ldots, t_n)$ gegeben. Die Konnektion wird *isoliert in ihrem i-ten Punkt* genannt, wenn die Konnektion isoliert ist, oder wenn für die Argumente s_i und t_i gilt, daß sie Grundterme sind, oder daß eines der beiden Argumente ein Grundterm ist und das Literal des anderen Argumentes nicht an einer weiteren Konnektion beteiligt ist. Wenn die Konnektion in diesem Punkt unifizierbar ist, kann auf den Klauseln einer solchen *punktweise isolierten Konnektion*[2] die erfolgreiche Substitution ausgeführt werden. Andernfalls kann die Konnektion entfernt werden.

5.2 SHRUTI als Reduktionsanwendung

Die Autoren von SHRUTI betrachten ihr Modell nur aus dem Gesichtspunkt einer parallelen Implementation für symbolisches Schließen. Sie zeigen eine konnektionistische Struktur, die mit parallel linearem Zeitaufwand arbeitet, gehen aber nicht auf die theoretischen Hintergründe ihres Modelles ein. Die Berechnungen des Netzes sollen nun im Zusammenhang mit dem Konnektionskalkül betrachtet werden. Damit untersuchen wir, inwieweit die Komplexitäten von SHRUTI in Beziehung zur Forderung Bibels nach adäquaten Beweismethoden stehen.

Beginnen wollen wir mit einer Feststellung zur Natur der existenzquantifizierten Variablen in SHRUTI.

Proposition 5.2.1 *Die Kodierung von existenzquantifizierten Variablen in SHRUTI ist äquivalent zur Skolemisierung dieser Variablen.*

Beweis:

[2]engl.: pointwise isolated connection

1. Die Syntax der Fakten und Anfragen sichert, daß zu Beginn einer Inferenz keine existenzquantifizierte Variable mehrfach auftritt (Annahme 3 der Bedingung 4.2.1 und 4.3.1). Mit Annahme 5 gilt dies auch für den Inferenzprozeß. Damit sind alle existenzquantifizierten Variablen in SHRUTI unterschiedlich.

2. Existenzquantifizierte Variablen werden durch Einheiten ohne Aktivierung dargestellt. Es wird definiert, daß alle Einheiten ohne Aktivierung prinzipiell als unterschiedliche Bindungen zu betrachten sind. In Verbindung mit Punkt 1 des Beweises ist dies äquivalent zum Ersetzen jeder existenzquantifizierten Variable durch eine Skolemfunktion bzw. -konstante.

3. Mit Annahme 4 aus Bedingung 4.2.1 wird für das Rückwärtsschließen die Unifikation mit allquantifizierten Variablen gesichert. Im Vorwärtsschließen ist dafür keine Annahme notwendig, weil dies mit der Syntax des Klauselkörpers schon garantiert wird.

Die Punkte 1, 2 und 3 des Beweises repräsentieren existenzquantifizierte Variablen als unterschiedliche Bindungen in SHRUTI und garantieren die korrekte Unifikation entsprechend einer skolemisierten Variablen. □

5.2.1 Rückwärtsschließen

In Abschnitt 4.2 haben wir bereits festgestellt, daß Rückwärtsschließen in SHRUTI einer parallelen Breitensuche gleicht. Die Wurzel des Suchbaumes ist die Anfrage, die an das Modell gestellt wird und die Verzweigungen entstehen durch die Regeln, in denen das Prädikat der Anfrage im Klauselkopf auftritt. Die Prädikate im Klauselkörper wiederum bilden die dynamisch erstellten Anfragen und damit die Teilziele in den unteren Ebenen des Suchbaumes.
Die Bearbeitung des Suchbaumes kann als Durchführung dreier Schritte betrachtet werden. Im ersten Schritt werden die Bindungen der Anfrage in die Äste des Suchbaumes propagiert. Dort erfolgt im 2. Schritt eine Auswertung der Teilziele und entsprechend dem vorhandenen Faktenwissen eine Bewertung des Suchpfades als Erfolg oder Mißerfolg. Die Bewertung wird in Form der Collectoraktivierung in die Wurzel des Baumes zurückpropagiert und erst im dritten Schritt werden für die Variablen in der Anfrage die Bindungen berechnet, unter denen die Anfrage wahr ist.

Bezeichnen wir das gesamte Wissen eines SHRUTI-Modelles, also die Menge aller Regeln und Fakten, mit F und sei G die Anfrage an das System, so wollen wir erfahren, ob $F \models G$ gilt. Das ist das Urteil, das wir einer positiven Antwort in SHRUTI zuordnen wollen. Wollen wir diesen Zusammenhang in einem Kalkül, zum Beispiel dem Konnektionskalkül, beweisen, so ist das äquivalent zum Beweis der Allgemeingültigkeit der Formel $\neg F \vee G$ oder dem Beweis der Unerfüllbarkeit der Formel $F \wedge \neg G$.
Betrachten wir nun noch einmal die Syntax in einem SHRUTI-Rückwärtsschließer. Jede Regel hat die Form

$$\forall X_1, \ldots, X_m [p_1(\ldots) \wedge p_2(\ldots) \ldots \wedge p_n(\ldots) \rightarrow \exists Z_1, \ldots Z_l q(\ldots)]$$

oder anders geschrieben

$$\forall X_1, \ldots, X_m[\neg p_1(\ldots) \vee \neg p_2(\ldots) \ldots \vee \neg p_n(\ldots) \vee \exists Z_1, \ldots Z_l q(\ldots)]$$

und alle Anfragen und Fakten haben die Form

$$\exists Z_1, \ldots, Z_l q(\ldots)$$

Die existenzquantifizierten Variablen in den Regeln werden mittels Skolemisierung durch Funktionssymbole ersetzt. Die Variablen des Faktenwissens werden durch diesen Schritt auf Konstantensymbole abgebildet. Das entspricht nach Proposition 5.2.1 genau dem Vorgehen in SHRUTI und ist zulässig, weil Skolemisierung gültigkeitserhaltend ist[3].
Mit der Kodierung in SHRUTI wird deshalb die Gesamtheit aller Regeln und Fakten als allabgeschlossene Formel F in konjunktiver Normalform betrachtet. Die Anfrage G an das System ist ein Literal, dessen Variablen existenzquantifiziert bzw. skolemisiert sind. Damit hat die Gleichung $\neg F \vee G$ einen Existenzabschluß und liegt in disjunktiver Normalform vor. Das ist genau die Normalform, die mit der Konnektionsmethode als allgemeingültig bewiesen werden kann. Die Negation $F \wedge \neg G$ liegt als allabgeschlossene Formel in konjunktiver Normalform vor. Das wiederum ist genau die Form, für die die Konnektionsmethode die Unerfüllbarkeit beweist. Damit ist gezeigt, daß die Logik von SHRUTI genau einer zu beweisenden Normalformgleichung entspricht, für die lediglich eine aufspannende Menge von Konnektionen gefunden werden muß.

Die hier erläuterte Problematik wird in [BH93] am Beweis der Unerfüllbarkeit und in [Str97] am Beweis der Allgemeingültigkeit eines SHRUTI-Beispiels beschrieben. In diesen Quellen wird auch gezeigt, daß der Suchbaum von SHRUTI genau den Konnektionen im Konnektionskalkül entspricht und die erfolgreichen Äste des Suchbaumes genau der Formel nach maximaler Anwendung der Reduktionen PICS, ISOL und PURE. Eine große Bedeutung haben dabei die Annahmen aus Bedingung 4.4.1 für multiple Instanzen. Sie beschreiben das Kopieren von Regeln und garantieren, daß jedes Literal im Kopf einer Klausel nur an maximal einer Konnektion verbunden und diese damit punktweise isoliert ist.

Diese Ergebnisse für das Rückwärtsschließen in SHRUTI werden von Strohmaier in den folgenden Propositionen festgehalten. Sie gibt auch die entsprechenden Beweise an.

Proposition 5.2.2 ([Str97]) *Sei F ein Beweisproblem aus SHRUTI und $\{A\} \in F$ die Anfrage. Dann ist $F \setminus \{\{A\}\}$ nicht allgemeingültig.*

Proposition 5.2.3 ([Str97]) *Sei S der Suchbaum für ein Beweisproblem F nach den ersten beiden Schritten von SHRUTI und sei F' die Formel, die aus F durch maximale Anwendung der Reduktionen PICS, PURE und ISOL hervorgeht. Dann entsprechen die von der Anfrage $\{G\}$ aus erreichbaren Konnektionen von F' genau den mit „Erfolg“ markierten Ästen des Suchbaumes S.*

[3] siehe [Bib92] Seite 88

Proposition 5.2.4 ([Str97]) *Die Terme jeder minimalen aufspannenden Paarung von F' sind simultan unifizierbar.*

Proposition 5.2.5 ([Str97]) *Reflexives Schließen, wie es von Shastri und Ajjanagadde in [SA93] definiert wurde, ist durch alleinige Anwendung von Reduktionen ausführbar.*

Weil Reduktionen gültigkeitserhaltend sind, haben wir somit einen Beweis, daß Rückwärtsschließen in SHRUTI korrekt ist. Es ist aber nicht vollständig, weil die Annahme 5 aus Bedingung 4.2.1 nur dynamisch während der Laufzeit geprüft werden kann. Ein weiterer Grund dafür ist auch die Annahme 6 aus Bedingung 4.4.1. Reicht die beschränkte Anzahl an Kopien nicht für alle aktiven Prädikatsinstanzen aus, dann können nicht alle Suchpfade des Problems evaluiert werden und eine gültige Anfrage bleibt möglicherweise unbewiesen.

5.2.2 Vorwärtsschließen

Die Unterschiede zwischen den beiden Richtungen der Inferenz werden besonders an der Betrachtung der Suche deutlich. Beim Rückwärtsschließen haben wir bereits gezeigt, daß ein Suchbaum existiert, der immer genau eine Wurzel hat, nämlich die Anfrage, die an das SHRUTI-System gestellt wird. Außerdem besteht eine echte Verzweigung, weil jede Anfrage auf die Gültigkeit mehrerer Teilziele abgebildet wird. Zwar handelt es sich auch beim Vorwärtsschließen um eine parallele Breitensuche, weil die möglichen Inferenzpfade gleichzeitig durchsucht werden, die Inferenz bildet jedoch im allgemeinen keine echte Baumstruktur. Das liegt daran, daß hier nicht eine einzelne Anfrage, sondern mehrere Fakten zum Beginn der Episode aktiviert werden. Daß diese Fakten möglicherweise als Langzeitwissen im Modell kodiert sind, ändert nichts daran, daß sie erst zum Beweisen genutzt werden können, wenn sie aktiviert sind. Bis zu diesem Zeitpunkt sind sie nur passiv vorhanden. Außerdem müssen beim eigentlichen Inferenzprozeß mehrere atomare Formeln unifiziert werden können, um eine neue Formel abzuleiten. Beim Vorwärtsschließen geht es offensichtlich nicht darum, die Allgemeingültigkeit des bestehenden Wissens zu beweisen, sondern Modelle zu generieren.
Sei in einem SHRUTI-Modell eine Wissensbasis aus einer Menge von kodierten Regeln und eine Menge aktivierter Fakten gegeben. Durch die Aktivierung werden von den Fakten Prädikatsinstanzen gebildet. Die Menge aller aktiven Instanzen ist damit Modell für jede Klausel des Faktenwissens. Ausgehend von diesem Modell wird bei der Berechnung für eine Regel die Existenz und Unifizierbarkeit der Atome im Klauselkörper geprüft und eine entsprechende Bewertung „Erfolg“ oder „Mißerfolg“ berechnet. Diese Bewertung entspricht häufig der Ausgabe einer zusätzlichen τ-und-Einheit, die mit dem Collector des Klauselkopfprädikates verbunden ist (vergleiche Abbildungen 4.9, 4.10) und gibt damit gleichzeitig an, ob eine Grundinstanz der Regel existiert, so daß die Menge aller Instanzen Modell für den Klauselkörper ist. Ist dies der Fall, wird die erfolgreiche Substitution für die Berechnung einer neuen Instanz des Klauselkopfprädikates verwendet und die neue Menge von gültigen Instanzen ist offensichtlich Modell der Regel. Anhand dieser

Erklärung wird sofort deutlich, daß das Vorgehen eine enge Beziehung zur Berechnung des Operators T_P (siehe Definition 2.1.11) hat. Auch dort wird eine Formel abgeleitet, wenn die gegebene Interpretation Modell für eine Grundinstanz des Klauselkörpers ist. Die Berechnungen in SHRUTI erfolgen parallel für alle Regeln bis zum Erreichen eines Fixpunktes, dem Zeitpunkt, ab dem keine neuen Prädikatsinstanzen mehr abgeleitet werden können. Das Bestreben von SHRUTI ist damit, ein Modell für die Menge aller Regeln und Fakten zu generieren, welches durch die aktiven Prädikatsinstanzen nach Ende einer Episode repräsentiert wird.

Für die beschriebene Form der Modellgenerierung ist in der Konnektionsmethode eine bestimmte Reihenfolge der Anwendung von Reduktionen notwendig.

Definition 5.2.1 (Prioritätsprinzip 1) *Eine PURE-Reduktion sollte auf einer Klausel erst dann angewendet werden, wenn alle Konnektionen dieser Klausel evaluiert wurden. Die Evaluierung der Konnektionen geschieht durch die Reduktion ISOL.*

Beim Vorwärtsschließen ist keine Anfrage gegeben, mit deren Negation die Unerfüllbarkeit zu beweisen ist, wie beim Rückwärtsschließen. In der Normalformmatrix der Wissensbasis tritt deshalb keine rein negative Klausel auf und mindestens ein positives Literal ist ohne Konnektion. Ohne Beachtung von Prioritätsprinzip 1 könnte so Schritt für Schritt mittels PURE jede Klausel aus der Matrix entfernt werden, ohne überhaupt ein Modell zu generieren. Die Modelle werden letztendlich nur durch Anwendung der Reduktion ISOL erstellt, weshalb diese den Vorrang vor PURE erhält.

Beispiel 5.2.1 *Sei eine Wissensbasis P gegeben, die aus den Fakten P_F*

$$\begin{aligned} &\rightarrow\ p(a,b)\\ &\rightarrow\ p(c,a)\\ &\rightarrow\ r(b) \end{aligned}$$

und dem Regelwissen P_R

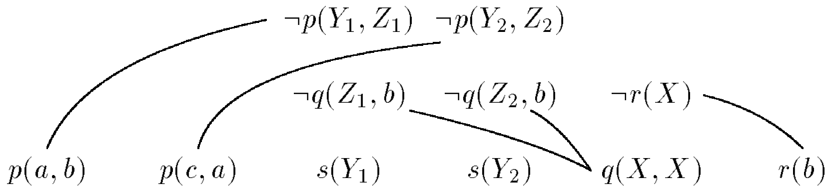
$$\begin{aligned} r(X) &\rightarrow q(X,X)\\ p(Y,Z)\wedge q(Z,b) &\rightarrow s(Y) \end{aligned}$$

besteht. Vom Prädikatssymbol p existieren die 2 Instanzen $p(a,b)$ und $p(c,a)$ als Fakten. Es wird die Erweiterung für multiple Instanzen (siehe Abschnitt 4.4.2) genutzt und es gelten somit die Annahmen aus Bedingung 4.4.1. Alle Regeln, in denen das Prädikat p im Klauselkörper auftritt, werden kopiert. Die zweite Klausel von P_R wird so ersetzt durch die beiden Regeln

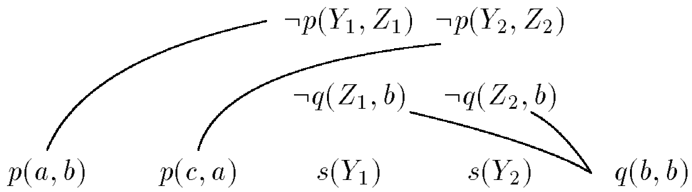
$$\begin{aligned} p(Y_1,Z_1)\wedge q(Z_1,b) &\rightarrow s(Y_1)\\ p(Y_2,Z_2)\wedge q(Z_2,b) &\rightarrow s(Y_2), \end{aligned}$$

wobei jeweils eine Instanz von p mit einer Regel verbunden ist.

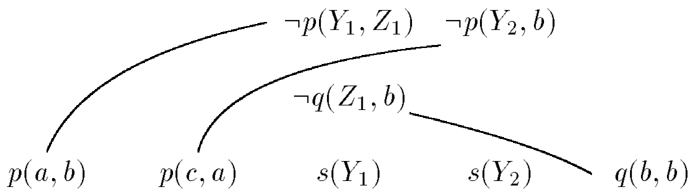
Betrachten wir nun das Beispiel 5.2.1 als Formel in Konjunktiver Normalform und seine Abarbeitung mittels Reduktionen. Die Darstellung der Reduktionen erfolgt schrittweise, tatsächlich ist sie für isolierte Verbindungen auch parallel möglich.



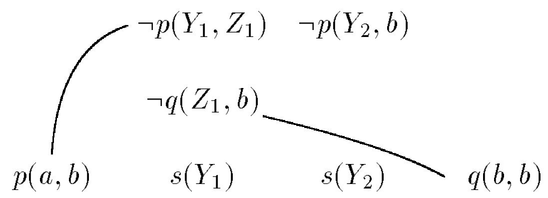
Die abgebildete Konnektionsstruktur entspricht genau den Verbindungen im SHRUTI-Modell. Mit der Aktivierung aller Fakten werden konkrete Instanzen gebildet und indirekt auch die Konnektionen aktiviert. Die Menge der Instanzen I_1 ist gleich P_F ($I_1 = P_F$) und damit auch Modell für alle Fakten ($I_1 \models P_F$).
Die Konnektionen $(r(b), \neg r(X)), (p(a,b), \neg p(Y_1, Z_1))$ und $(p(c,a), \neg p(Y_2, Z_2)$ sind isoliert und die mittleren beiden Klauseln enthalten ein pures Literal. Nach dem Prioritätsprinzip wenden wir nun nicht PURE an, sondern beginnen mit der isolierten Verbindung $(r(b), \neg r(X))$. Mit der Substitution $\{X \mapsto b\}$ kann $r(b, b)$ abgeleitet werden, und man erhält folgende reduzierte Matrix.



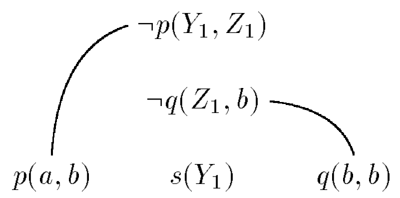
Die Menge $I_2 = I_1 \cup \{q(b, b)\}$ ist nun Modell von P_F und auch Modell für die Regel $r(X) \rightarrow q(X, X)$. Außerdem sind durch den Reduktionsschritt weitere isolierte Verbindungen entstanden. Bei Anwendung von ISOL auf die Konnektion $(q(b,b), \neg q(Z_2, b))$ wird die Substitution $\{Z_2 \mapsto b\}$ durchgeführt.



Die Reduktion ISOL für $(p(c, a), \neg p(Y_2, b))$ führt zu dem Ergebnis, daß die Literale der Konnektionen nicht unifizierbar sind. ISOL schlägt fehl und es verbleibt die Resolvente der Klausel.



Jetzt sind alle Konnektionen der zweiten Kopie von $p(Y, Z), q(Z, b) \rightarrow s(Y)$ entfernt, und ohne Verletzung des Prioritätsprinzips kann PURE angewandt werden.

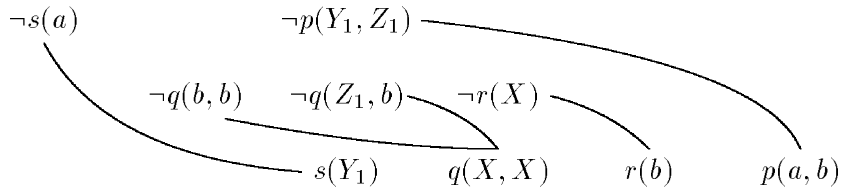


Die beiden Konnektionen der verbleibenden Matrix sind ebenfalls isoliert. Ihre Evaluierung führt zu der Substitution $\{Y_1 \mapsto a, Z_1 \mapsto b\}$ und somit zu der Formel

$$s(a)$$

Die Folgerung $s(a)$ ist damit die letzte Inferenz, die aus der Formel gezogen werden kann. Die Menge $I_3 = I_2 \cup \{s(a)\} = \{p(a,b), p(c,a), r(b), q(b,b), s(a)\}$ ist damit Modell für alle Fakten und alle Regeln und es gilt $I_3 \models (P_R \cup P_F)$. Gleichzeitig ist I_3 sogar kleinstes Modell von $P_R \cup P_F$.
Die verbleibende Matrix wird nun durch eine weitere PURE-Reduktion in eine leere Matrix verwandelt. Das Problem wurde allein durch maximale Anwendung der Reduktionen ISOL und PURE gelöst und als erfüllbar bewiesen, weil keine aufspannende Menge von Konnektionen existiert.

Werden die Resolventen $q(b, b)$ und $s(a)$ der erfolgreichen ISOL-Reduktionen negiert zu der initialen Matrix hinzugefügt, dann bildet sich eine minimale aufspannende Menge von Konnektionen, die mit $\{X \mapsto b, Y_1 \mapsto a, Z_1 \mapsto b\}$ simultan unifizierbar ist.



Die Matrix ist unerfüllbar und die atomaren Resolventen sind damit logische Konsequenzen aus $(P_R \cup P_F)$. Zur Menge aller Konsequenzen gehören weiterhin die Fakten $p(a,b)$, $p(c,a)$ und $r(b)$. Die Menge der logischen Konsequenzen ist für dieses Beispiel gleich dem mit den Reduktionen ermittelten Modell I_3.

Theorem 5.2.6 *Vorwärtsschließen in SHRUTI ist identisch mit dem Generieren von Modellen durch Anwendung von Reduktionen nach dem Prioritätsprinzip 1.*

Beweis:
Der Beweis wird induktiv über die Länge der Inferenzkette d geführt. Er betrachtet eine Menge von Formeln P in KNF, bestehend aus dem Faktenwissen P_F und den Regeln P_R mit der Syntax von Klauseln in SHRUTI.

- $(\mathbf{d = 0})$ Die Inferenz wird initialisiert mit der Aktivierung von Fakten P_F. Die Menge der Prädikatsinstanzen ist $I_0 = \emptyset$.
- $(\mathbf{d = 1})$ Sei $K_1 \in P_F$ ein Fakt. Die Menge der Instanzen I_0 ist Modell für den Klauselkörper von K_1. Eine Instanz von K_1 wird gebildet und die neue Menge von Instanzen I_1 ist Modell für K_1. Die anderen Fakten werden parallel bearbeitet und I_1 ist Modell für jeden Fakt und damit Modell für P_F.
  Ist die Menge der Instanzen die leere Menge ($I_1 = \emptyset$), dann gilt sogar $I_1 \models P$ und der Inferenzvorgang ist beendet.
- $(\mathbf{d = 2})$ Sei $K_2 \in P_R$ eine Klausel (oder die Kopie einer Klausel), deren negative Literale alle jeweils mit einem Fakt verbunden sind. Nach der Syntax von Fakten in SHRUTI und nach Proposition 5.2.1 ist jeder Fakt ein Grundatom. Annahme 7 aus Bedingung 4.4.1 garantiert, daß jedes negative Literal von K_2 nur eine Konnektion hat. Damit sind alle Konnektionen im Klauselkörper isoliert. Nach Prioritätsprinzip 1 wird ISOL angewendet. Dabei sind zwei Fälle zu unterscheiden. Alle Konnektionen sind
  - unifizierbar:
    Es existiert eine Grundinstanz von K_2, so daß I_1 Modell ist für den Klauselkörper von K_2. Das Atom im Kopf der Grundinstanz wird abgeleitet und die neue Menge der Instanzen I_2 ist Modell für P_F und für K_2. Wegen Annahme 4 aus Bedingung 4.3.1 und Proposition 5.2.1 ist die abgeleitete Instanz erneut ein Grundatom.
  - nicht unifizierbar:
    Ist K_2 eine Originalklausel, dann existiert keine Grundinstanz von K_2, so daß I_1 Modell ist für den Klauselkörper. Deshalb ist I_1 bereits Modell für P_F und K_2. Es wird keine Instanz gebildet und die verbleibende Klausel wird durch PURE entfernt.
    Ist K_2 Kopie einer Klausel, dann sind nur die konnektierten Instanzen Modell für K_2. Alle anderen Kopien der Originalklausel werden parallel in Schritt 2 bearbeitet, so daß mit deren abgeleiteten Instanzen gilt $I_2 \models K_2$.

Ist ein negatives Literal von K_2 nicht konnektiert, dann gilt dasselbe, wie bei Nichtunifizierbarkeit.

- $(\mathbf{d = n \rightarrow m > n})$ Sei K_n $(n \geq 1)$ eine Klausel (oder die Kopie einer Klausel), für die mit der Menge der Instanzen I_n gilt $I_n \models P_F \cup K_n$. Es gelte weiterhin, daß K_n durch Prioritätsprinzip 1 zu Grundatom oder leerer Klausel reduziert wird.
  Sei K_m $(m > n)$ eine Klausel (oder die Kopie einer Klausel). Jedes negative Literal von K_m ist mit dem Klauselkopf einer Klausel K_n verbunden, wenn es von einem Fakt aus erreichbar[4] ist. Ist dies nicht der Fall, gilt $I_n \models K_m$ und die Klausel wird mit PURE entfernt. Andernfalls sind nach Vorraussetzung alle Konnektionen isoliert. Die Evaluierung mit ISOL erfolgt wie in Punkt $(\mathbf{d = 2})$, so daß gilt $I_m \models P_F \cup K_m$.

Es wurde für $d = 1$ und $d = 2$ und jede beliebige größere Tiefe gezeigt, daß die Menge aller durch Prioritätsprinzip 1 entstehenden Grundatome Modell für jede Regel bis zu dieser Tiefe ist. □

Bemerkung: Der Beweis verdeutlicht, daß die Abarbeitung nach dem Prioritätsprinzip eine positive Hyperresolution[5] ist. Dieses Verfahren ist korrekt und vollständig. Das gilt jedoch nicht gleichzeitig für SHRUTI.

Proposition 5.2.7 *Sei $P = P_F \cup P_R$ eine Menge von Formeln in SHRUTI und sei I_∞ die Menge aller Instanzen am Ende des Inferenzprozesses über der Wissensbasis P bzw. das durch SHRUTI generierte Modell. Dann gilt:*

$$I_\infty \subseteq T_P(\emptyset) \uparrow \infty$$

Beweis: Der Beweis für Theorem 5.2.6 hat induktiv gezeigt, daß für beliebige Inferenzlänge $d > 0$ genau dann eine neue Instanz abgeleitet wird, wenn I_d Modell für den Klauselkörper einer Klausel ist. Das ist gleichbedeutend damit, daß eine Grundinstanz des Klauselkörpers existiert, für die I_d Modell ist. Die Anwendung des Prioritätsprinzips in einem parallelen Inferenzschritt ist nach Definition 2.1.11 offensichtlich die Berechnung von $T_P(I_d)$. Mit der Speicherung von Prädikatsinstanzen $I_d = I_{(d-1)} \cup T_P(I_{(d-1)})$ und der Gültigkeit von $I_1 = P_F$ wird folglich $T_P(\emptyset) \uparrow d$ berechnet. Das gilt jedoch nur, wenn alle Annahmen von SHRUTI eingehalten werden. Annahme 5 aus Bedingung 4.3.1 ist nur dynamisch prüfbar. Wird sie im Inferenzprozeß verletzt oder ist die Anzahl der Klauselkopien nach Annahme 6 der Bedingung 4.4.1 nicht ausreichend, dann erfolgt keine Ableitung und I_d ist nicht Modell für die entsprechende Klausel. Damit gilt allgemein $I_d \subseteq T_P(\emptyset) \uparrow d$ und für $d \rightarrow \infty$ ist die Proposition bewiesen. □

Die Menge I_∞ enthält nur Elemente, die auch in $T_P(\emptyset) \uparrow \infty$ sind. Nach den Propositionen 3.1.2 und 3.1.3 ist das genau die Menge der Grundformeln, die logische Konsequenz aus P sind. Deshalb gilt auch für alle Grundatome G,

$$G \in I_\infty \Rightarrow P \models G,$$

[4] Erreichbarkeitsbegriff siehe [Str97] Seite 53

[5] siehe [Bib92] Seite 69

was zusammen mit Proposition 5.2.7 sofort zu der Feststellung führt

Theorem 5.2.8 *Vorwärtsschließen in SHRUTI ist korrekt, aber nicht vollständig.*

Die Proposition 5.2.5 von Strohmaier kann mit den Ergebnissen von Vorwärtsschließen in SHRUTI bestätigt werden. Die parallele Ausführung der Reduktionen im Modell erreicht die parallel lineare Zeitkomplexität bezüglich der Inferenzkette. So gesehen ist SHRUTI eine Implementation der Konnektionsmethode und wenn die Logik tatsächlich der reflexiv ausführbaren entspricht, adäquat im Sinne von Bibel.
Außerdem wurde deutlich, daß jede Anwendung der Reduktionen ISOL vor PURE in einer Hornklausellogik in enger Beziehung zur Berechnung von T_P steht, weil zuerst die Literale im Klauselkörper resolviert werden. Damit ist ein erster Zusammenhang zwischen dem RNN-Modell und dem Vorwärtsschließen in SHRUTI gefunden.

5.3 BUR - Ein Kalkül mit Reduktionen

5.3.1 Die Reduktionen DB, ISOL und PURE

Für eine Klasse logischer Programme, die nur mittels Reduktionen lösbar ist, aber gegenüber SHRUTI den Vorteil hat, keine Kopien von Klauseln erstellen zu müssen, soll zunächst eine Form der Datenverdichtung betrachtet werden. Diese Reduktionsmethode ist in einer Hornklausellogik geeignet, isolierte Verbindungen zu erzeugen. Sie wird als DB-Reduktion bezeichnet und in Verbindung mit ISOL in [Bib87a] als Vorwärtsreduktion beschrieben.

- **DB:** Der Reduktion DB liegt folgende Originalformel zugrunde:

  $$p(r) \leftarrow r \in \{c_1, \ldots, c_k\} \leftrightarrow p(c_1) \wedge \ldots \wedge p(c_k) \tag{5.1}$$

  Die konjunktive Verknüpfung einer Menge von k Instanzen des einstelligen Prädikates p wird ersetzt durch eine Instanz $p(r)$, wobei r Element der Menge von Argumenten $\{c_1, \ldots, c_k\}$ ist. Die Verallgemeinerung der Formel für mehrstellige Prädikate p mit der Stelligkeit s ergibt

  $$p\vec{r} \leftarrow \vec{r} \in \{(c_{11}, \ldots, c_{1s}), \ldots, (c_{k1}, \ldots, c_{ks})\} \leftrightarrow p(c_{11}, \ldots, c_{1s}) \wedge \ldots \wedge p(c_{k1}, \ldots, c_{ks}) \tag{5.2}$$

  Die Menge von k Instanzen wird äquivalent ersetzt durch $p\vec{r}$, wobei $\vec{r}$ ein Tupel aus der Menge der Bindungen repräsentiert. Weil $\vec{r}$ ein Vektor bzw. Tupel ist, wird er in der Formel nicht geklammert.

Die Betrachtung einer Menge von Tupeln ist vergleichbar mit einer Menge von Datensätzen mit der Anzahl s an Attributen. In Kombination von DB und ISOL bedeutet dies, daß die Evaluierung der isolierten Verbindungen gegebenenfalls über Mengen von

Vektoren unifizieren muß. Es verwundert deshalb nicht, daß dafür die Datenbankoperationen Vereinigung $\cup$, Durchschnitt $\cap$, Projektion π und Join $\bowtie$ benutzt werden.

Die nachfolgenden Definitionen sind auf die Notation aus [MW98] zurückzuführen.

Definition 5.3.1 *Seien zwei Relationen $R_1(R_{11}, \ldots, R_{1s_1})$ und $R_2(R_{21}, \ldots, R_{2s_2})$ gegeben, für die $s_1 = s_2$ gilt und die Wertebereiche aller Attribute R_{1i} und R_{2i} $(1 \leq i \leq s_1)$ gleich sind. Die* Vereinigung $\cup$ *ist definiert durch:*

$$R_1 \cup R_2 = \{r \mid r \in R_1 \vee r \in R_2\}$$

Der Durchschnitt $\cap$ *ist definiert durch:*

$$R_1 \cap R_2 = \{r \mid r \in R_1 \wedge r \in R_2\}$$

Definition 5.3.2 *Sei eine Relation $R(R_1, \ldots, R_s)$ und eine Folge von Attributen $B_i = (R_{i_1}, \ldots, R_{i_k})$ aus der Menge der Attribute $\{R_1, \ldots, R_s\}$ gegeben. Für ein Element $r = (r_1, \ldots, r_s)$ der Relation R gilt:*

$$r[B] = (r_{i_1}, \ldots, r_{i_k}).$$

Die Projektion π *ist definiert durch:*

$$\pi_B(R) = \{r[B] \mid r \in R\}$$

Definition 5.3.3 *Seien $R_1(R_{11}, \ldots, R_{1s_1})$ und $R_2(R_{21}, \ldots, R_{2s_2})$ Relationen. Das* kartesische Produkt $\otimes$ *ist definiert durch:*

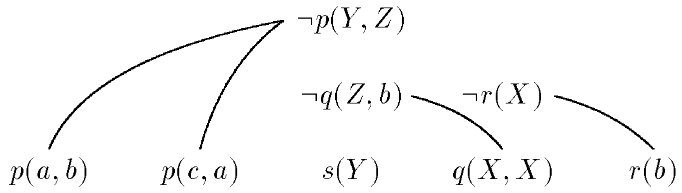
$$R_1 \otimes R_2 = \{(r_{11}, \ldots, r_{1s_1}, r_{21}, \ldots, r_{2s_2}) \mid (r_{11}, \ldots, r_{1s_1}) \in R_1 \wedge (r_{21}, \ldots, r_{2s_2}) \in R_2\}$$

Definition 5.3.4 *Seien $R_1(R_{11}, \ldots, R_{1s_1})$ und $R_2(R_{21}, \ldots, R_{2s_2})$ Relationen. Seien weiterhin eine Menge von Attributen $\{R_{1i_1}, \ldots, R_{1i_k}\}$ und $\{R_{2j_1}, \ldots, R_{2j_k}\}$ gegeben, wobei für alle l $(1 \leq l \leq k)$ gilt, daß der Wertebereich von R_{1i_l} gleich dem Wertebereich von R_{2j_l} ist. Der* natürliche Verbund *wird definiert durch:*

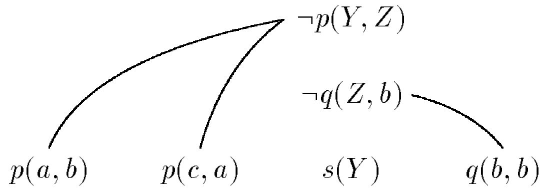
$$\begin{aligned} R_1 \bowtie_{(R_{1i_1}=R_{2j_1},\ldots,R_{1i_k}=R_{2j_k})} R_2 \quad = \quad & \{r = (r_{11}, \ldots, r_{1s_1}, r_{21}, \ldots, r_{2s_2}) \mid r \in R_1 \otimes R_2 \wedge \\ & r_{1i_1} = r_{2j_1} \wedge \ldots \wedge r_{1i_k} = r_{2j_k}\} \end{aligned}$$

Der natürliche Verbund wird auch als natural Join *bezeichnet und ist ein Spezialfall der allgemeinen Join-Operation $\bowtie$.*

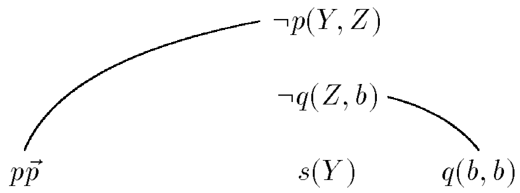
Die Reduktion DB hat die Aufgabe, isolierte Verbindungen zu erzeugen. Auf das Prioritätsprinzip hat dies insofern Einfluß, daß DB angewendet wird, wenn eine Konnektion nicht isoliert ist, weil mehrere Prädikatsinstanzen auftreten. Die Kombination von DB, ISOL und PURE, sowie die damit verbundenen Datenbankoperationen sollen noch einmal am Beispiel 5.2.1 demonstriert werden. So wird auch der Unterschied zum Vorwärtsschließen mit Reduktionen in SHRUTI deutlich. Im Gegensatz dazu werden hier keine Klauselkopien erstellt, so daß auch die Regel $p(Y, Z) \wedge q(Z, b) \rightarrow s(Y)$ nur einmal in der Normalformmatrix auftaucht.



Die Menge der atomaren Klauseln ist die Menge der Fakten $I_1 = \{p(a,b), p(c,a), r(b)\}$ und es gilt $I_1 \models P_F$. Die Konnektion $(r(b), \neg r(X))$ ist isoliert. Bei Durchführung von ISOL wird die Substitution $\{X \mapsto b\}$ berechnet und die Formel $q(b,b)$ gefolgert.



Die Menge $I_2 = I_1 \cup \{q(b,b)\}$ ist nun Modell für das Faktenwissen P_F und die Regel $r(X) \rightarrow q(X,X)$. In der verbleibenden Matrix ist nur die Konnektion $(q(b,b), \neg q(Z,b))$ isoliert, die anderen Konnektionen nicht. Nun wird die DB-Reduktion auf die Prädikatsinstanzen $p(a,b)$ und $p(c,a)$ angewendet und nach Gleichung 5.2 erhält man mit Hilfe der Vereinigungsoperation $\{(a,b)\} \cup \{(c,a)\}$ die Bedingung $\vec{p} \in \{(a,b),(c,a)\}$ und die Matrix



Jetzt können alle verbleibenden Konnektionen mit ISOL bearbeitet werden. Betrachtet man p als Relation der Form $p(p_1,p_2) = \{(a,b),(c,a)\}$ und q als Relation der Form $q(q_1,q_2) = \{(b,b)\}$, dann ist die Durchführung der ISOL-Reduktionen eine simultane Unifikation durch die Datenbankoperation

$$s(Y) \leftarrow Y \in \pi_{p_1}(p \bowtie_{(p_2=q_1, q_2=b)} q)$$

und man erhält wie beim Vorwärtsschließen in SHRUTI

$$\begin{array}{lcl} s(Y) & \leftarrow & Y \in \pi_{p_1}\{(a,b,b,b)\} \\ s(Y) & \leftarrow & Y \in \{a\} \\ s(a) & & \end{array}$$

Das Beispiel macht deutlich, wann welche Datenbankoperation für die Ausführung der Reduktionen notwendig ist.

- Bei der DB-Reduktion erfolgt die Vereinigung $\cup$ aller Tupel, die den Bindungen der abgeleiteten Formeln jeweils eines Prädikatszeichens entsprechen.
- Die parallele Anwendung von ISOL auf eine Klausel ist im allgemeinen Fall eine Kombination der Projektion π und der Join-Operation $\bowtie$. Mit dem natürlichen Verbund werden die Tupel ermittelt, mit denen die Literale einer Klausel simultan unifizierbar sind. Die Projektion ist dann dafür verantwortlich, aus dem Verbund eine Folge von Attributen zu erstellen, die genau der Bindung im Klauselkopf entspricht.
- Der Durchschnitt $\cap$ kann lediglich in Spezialfällen zu unifizierender Literale eingesetzt werden. Haben die Klauselkörperliterale bei paralleler Anwendung von ISOL exakt dieselbe Bindung (Bsp.: $p(X,Y) \wedge q(X,Y) \rightarrow r(X,Y)$), dann reduziert sich die Nacheinanderausführung von Verbund und Projektion auf die Berechnung des Durchschnittes zwischen den Mengen, die die Bindungen repräsentieren.
- Die PURE-Reduktion bedarf nicht zwingend der Ausführung einer Datenbankoperation. Sie wird aber implizit durch die Join-Operation realisiert. Wenn ein Literal im Klauselkörper keine Konnektion hat, dann ist die Menge der abgeleiteten Formeln dieses Prädikates leer und der natürliche Verbund über ihnen ebenfalls.

Fassen wir den Sachverhalt noch einmal zusammen. Um zu verhindern, daß das Kopieren von Klauseln nötig ist, wird eine Datenverdichtung angewendet, die DB-Reduktion. Sie schafft isolierte Verbindungen, allerdings bildet sie auch Mengen von Vektoren. Dies wiederum hat zur Folge, daß die parallele Evaluierung der Konnektionen einer Klausel mit ISOL der Berechnung von Datenbankoperationen entspricht. Das Ergebnis ist wieder eine Menge von Vektoren, die die Bindungen der abgeleiteten Formeln darstellt.
Betrachtet man die DB-Reduktion genau, fällt auf, daß der eingesparte Platzaufwand durch die Repräsentation der Relationen wieder verbraucht wird. In einer Relation werden immer die Bindungen von einem Prädikat zusammengefaßt. Für ein Atom mit der Stelligkeit s gibt es C^s verschiedene solcher Bindungen ($C = |U_P|$). Im schlechtesten Fall müßten alle Tupel repräsentiert werden, so daß für jedes Prädikatssymbol eine worst-case-Komplexität von $(O(C^s))$ folgt. Unter Betrachtung des average-case ergibt sich vermutlich für die Anzahl darzustellender Instanzen $\frac{C^s}{2}$. Das ändert aber noch nichts an der exponentiellen Komplexität zur Darstellung der Mengen.
Hinzu kommt bei dieser Problematik der Aufwand der Join-Operation. Sie besteht nach Definition aus der Nacheinanderausführung von kartesischem Produkt und Restriktion. Werden bei einer Anwendung von Join zwei etwa gleich große Relationen verknüpft und genügen die resultierenden Tupel den Anforderungen der Restriktion, dann ist daß Ergebnis eine quadratische Anzahl von Tupeln, die zu repräsentieren ist. Bei einer Klausel mit n Literalen werden möglicherweise $(n-1)$ Join-Operationen notwendig und im worst-case ergibt sich eine Platzkomplexität von $O(C^{s\times(n-1)})$. Selbst wenn die Anzahl der Instanzen sehr klein ist, ist mit exponentiellem Platzaufwand und darauf folgend mit exponentiellem Zeitbedarf bezüglich der Menge der abgeleiteten Formeln zu rechen.

Die kombinierte Anwendung von DB, ISOL und PURE ist aus diesen Gründen für eine Verwendung in einem Kalkül, der linearen Platz- und parallel linearen Zeitaufwand haben soll, nicht geeignet.

5.3.2 Der BUR-Kalkül

Die Suche nach einem Ausweg aus der exponentiellen Platzkomplexität bei Anwendung von DB führt zu der Idee, daß $s = 1$ gelten muß. Dann ergibt sich nach den bisherigen Betrachtungen im schlechtesten Fall ein Aufwand für die Repräsentation der Relationen, der durch $O(C^1 \times 1)$ beschränkt und damit linear zur Größe des Herbrand-Universums ist.
Die Festlegung $s = 1$ soll aber nicht eine generelle Einschränkung der Sprache auf einstellige Prädikatssymbole darstellen. Deshalb wird eine neue Form der Repräsentation von Atomen genutzt. Statt die Bindungen der Instanzen eines Prädikates durch eine Menge von Vektoren der Länge s zu repräsentieren, werden sie durch s Mengen von Vektoren der Länge 1 dargestellt. Das bedeutet praktisch, daß die Formeln durch die Menge der Argumente in den einzelnen Stellen gekennzeichnet wird. Man erreicht damit für die Darstellung der Formeln im worst-case eine Platzkomplexität von $O(C \times s)$. Werden die Terme, wie in SHRUTI, auf Konstantensymbole beschränkt, so ist C endlich und der Speicheraufwand tatsächlich linear.
Die beschriebene Transformation bildet Mengen in jeder Stelle eines Prädikates. Sie soll deshalb im folgenden als punktweise DB-Anwendung bezeichnet werden. Sie ist nicht gültigkeitserhaltend, weil die Formeln nicht durch Vektoren, sondern durch das Kreuzprodukt von Argumenten kodiert werden, und dabei CROSS-TALK auftreten kann. Aus diesem Grund soll die punktweise Anwendung von DB auch nicht als Reduktion bezeichnet werden. Sie ist nicht gültigkeitserhaltend und entspricht damit nicht den allgemeinen Anforderungen an Reduktionen nach Bibel.

- **punktweise DB-Anwendung:** Die Anwendung von DB in jedem Argument (Punkt) eines Prädikates wird beschrieben durch

$$\begin{aligned} [p(r_1, \ldots, r_s) &\leftarrow r_1 \in \{c_{11}, \ldots, c_{k1}\}, \ldots, r_s \in \{c_{1s}, \ldots, c_{ks}\}] \\ &\leftarrow p(c_{11}, \ldots, c_{1s}) \wedge \ldots \wedge p(c_{k1}, \ldots, c_{ks}) \end{aligned} \tag{5.3}$$

  Eine Menge von k Instanzen wird damit ersetzt durch s Mengen, wobei jede Menge die gebundenen Argumente in einer Stelle des Prädikates enthält. Die Äquivalenz $\leftrightarrow$, die noch in Gleichung 5.2 galt, wird ersetzt durch $\leftarrow$, weil die Transformation nicht gültigkeitserhaltend ist.

Die veränderte Repräsentation der Bindungen hat natürlich auch Einfluß auf die Datenbankoperationen bei der parallelen Anwendung von ISOL. Jetzt sind lediglich einstellige Relationen zu verknüpfen und die Ausführung von natürlichem Verbund und Projektion reduziert sich deshalb auf die Berechnung des Durchschnittes zwischen den Mengen, die

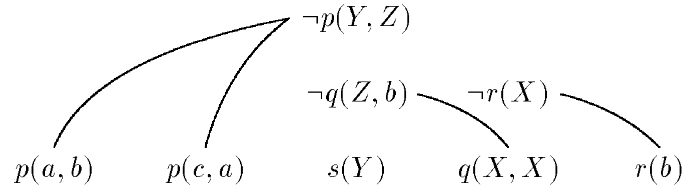
an dieselbe Variable oder Konstante gebunden sind. So werden für das Generieren von Modellen nur die Operationen Vereinigung $\cup$, Durchschnitt $\cap$, Projektion π und kartesisches Produkt $\otimes$ benötigt.
Die Vereinigung und die Projektion haben linearen Zeitaufwand und linearen Platzbedarf bezüglich der Größe der Relationen. Die Durchschnittsoperation ist im worst-case quadratisch von der Größe der Relationen abhängig, bei entsprechender Repräsentation der Menge kann sie aber auch mit linearem Zeitbedarf realisiert werden. Das kartesische Produkt bildet im Normalfall wieder Tupeln. Durch die Gültigkeit der Gleichung 5.3 kann das jedoch vermieden werden, so daß tatsächlich der Platzaufwand linear ist.

Deshalb können mit der punktweisen Anwendung von DB alle Reduktionen mit linearem Platz- und Zeitaufwand realisiert werden und die Grundlage für einen Kalkül mit solchen Anforderungen ist damit geschaffen.

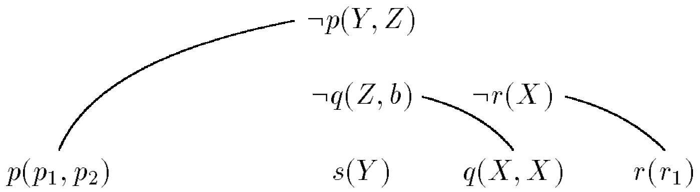
Als Grundlage für das Generieren von Modellen im BUR-Kalkül wird nun eine neue Abarbeitungsreihenfolge festgelegt. Sie unterstützt erneut ISOL vor PURE und behält damit den Zusammenhang zum Operator T_P.

Definition 5.3.5 (Prioritätsprinzip 2) *Die Reduktion DB wird punktweise (in jedem Argument) auf alle Fakten der Wissensbasis angewendet. Eine PURE-Reduktion sollte auf einer Klausel erst dann angewendet werden, wenn alle Konnektionen dieser Klausel evaluiert wurden. Die Evaluierung der Konnektionen geschieht durch die Reduktion ISOL.*

Auch mit Prioritätsprinzip 2 soll noch einmal das Beispiel 5.2.1 bearbeitet werden. Es müssen keine Klauselkopien erstellt werden und jede Klausel aus P taucht genau einmal in der Normalformmatrix auf.



Mit der punktweisen Anwendung von DB auf die Fakten $p(a,b)$, $p(c,a)$ und $r(b)$ ergibt sich $p_1 \in \{a,c\}$, $p_2 \in \{b,a\}$ und $r_1 \in \{b\}$ und die Matrix



Die Repräsentation der Bindungen hat zur Folge, daß für die Menge der abgeleiteten Formeln gilt $I_1 = \{p(a,a), p(a,b), p(c,a), p(c,b), r(b)\}$. Es fällt auf, daß durch CROSSTALK zwei neue Formeln entstanden sind, nämlich $p(a,a)$ und $p(c,b)$. I_1 ist damit

trotzdem Modell für P_F, aber nicht mehr kleinstes Modell.
Alle Konnektionen der Matrix sind nun isoliert. Bei der Evaluierung der Verbindung $(r(r_1), \neg r(X))$ ergibt sich $q(q_1, q_2) \leftarrow q_1 \in \{b\}, q_2 \in \{b\}$ und die Matrix

$$\begin{array}{ccc} & \neg p(Y,Z) & \\ & \neg q(Z,b) & \\ p(p_1,p_2) & s(Y) & q(q_1,q_2) \end{array}$$

Die Menge $I_2 = I_1 \cup q(b,b)$ ist Modell für P_F und die erste Regel aus P_R. Alle verbleibenden Konnektionen können mit ISOL bearbeitet werden. Für die Darstellung dessen als Datenbankoperation betrachten wir die Menge der Bindungen von p und q in jedem ihrer Punkte, und erhalten $\mathbf{p}_1 = \{a, c\}$, $\mathbf{p}_2 = \{b, a\}$, $\mathbf{q}_1 = \{b\}$ und $\mathbf{q}_2 = \{b\}$. Die parallele Durchführung der ISOL-Reduktionen entspricht einer simultanen Unifikation durch die Berechnung

$$s(s_1) \leftarrow s_1 \in \pi_{p_1}(\mathbf{p}_1 \otimes (\mathbf{p}_2 \cap \mathbf{q}_1) \otimes (\mathbf{q}_2 \cap \{b\}))$$

Dies resultiert in

$$\begin{array}{lcl} s(s_1) & \leftarrow & s_1 \in \pi_{p_1}\{(a,b,b),(c,b,b)\} \\ s(s_1) & \leftarrow & s_1 \in \{a, c\} \end{array}$$

Damit ergibt sich die Menge $I_3 = I_2 \cup \{s(a), s(c)\}$, die Modell von P, aber nicht kleinstes Modell von P ist. Die abgeleiteten Formeln $p(a,a)$, $p(c,b)$ und $s(c)$ sind keine logischen Konsequenzen aus P.

Für die dargelegten Zusammenhänge wird nun der Kalkül definiert. Die Abkürzung BUR steht dabei für die englische Bezeichnung „bottom-up reductions", die vorwärtsschließende Reduktionen kennzeichnet.

Definition 5.3.6 *Das Alphabet* $\mathbf{A}_{BUR}$ *des Kalküls besteht aus folgenden Symbolen: den Variablen* $(X, Y, Z, \ldots)$*, den Konstanten* $(a, b, c, \ldots)$*, den Prädikatssymbolen* $(p, q, r, \ldots)$*, den Junktoren* $(\wedge, \vee, \rightarrow, \neg)$*, dem Quantor* $\forall$ *und Begrenzungssymbolen.*

Definition 5.3.7 *Eine Formel der Menge* $\mathbf{F}_{BUR}$ *des Kalküls ist entweder eine Regel, ein Fakt oder eine Anfrage.*

Definition 5.3.8 *Eine Regel aus* $\mathbf{F}_{BUR}$ *ist eine Formel der Form*

$$\forall X_1, \ldots, X_m[B_1(\ldots) \wedge B_2(\ldots) \wedge \ldots \wedge B_n(\ldots) \rightarrow A(\ldots)]$$

wobei $B_1, B_2, \ldots, B_n$ *und* A *Atome sind. Ihre Argumente sind Elemente der Mengen* $\{X_1, \ldots, X_m\}$ *und* U_P *(Menge der Konstanten).*

Definition 5.3.9 *Ein Fakt aus $\mathbf{F}_{BUR}$ ist eine Formel der Form*

$$\rightarrow A(c_1, \ldots, c_s) \text{ bzw. } A(c_1, \ldots, c_s) \text{ mit } c_1, \ldots, c_s \in U_P$$

wobei A ein Atom ist. Eine Anfrage hat die gleiche Syntax wie ein Fakt.

Ein Fakt aus der Prädikatenlogik erster Stufe, der die Form $\exists X\, A(X)$ hat, kann durch Skolemisierung auf einen Fakt der BUR-Logik abgebildet werden. Dabei wird die existenzquantifizierte Variable durch eine neue Konstante ersetzt. Die Skolemisierung ist gültigkeitserhaltend und deshalb in diesem Zusammenhang zulässig.

Wie man leicht sieht, handelt es sich bei der hier definierten Sprache um eine Teilmenge der Formeln der Prädikatenlogik erster Stufe. Das heißt insbesondere, daß alle Definitionen für prädikatenlogische Programme auch für diese Sprache gültig sind. Damit gilt nach Definition 2.1.2, daß alle Wissensbasen des BUR-Kalküls auch definite Programme im prädikatenlogischen Sinn sind.

Definition 5.3.10 *Für die Menge der Axiome gilt $\mathbf{X}_{BUR} = \emptyset$.*

Definition 5.3.11 *Sei $P = P_F \cup P_R$ eine Menge von Fakten P_F und Regeln P_R aus $\mathbf{F}_{BUR}$ und sei $G = p(a_1, \ldots, a_s)$ eine Anfrage. Die Inferenzrelation im BUR-Kalkül ist rekursiv definiert über der Tiefe der Inferenz $d > 0$, wie folgt:*

1. *Fakten sind abgeleitete Formeln im Inferenzschritt 1:*

$$P \vdash^1_{BUR} G \qquad \text{, wenn } G \in P_F$$

2. *die punktweise Anwendung von DB ergibt abgeleitete Formeln:*

$$P \vdash^d_{BUR} G \qquad \text{, wenn für alle } i \text{ mit } 1 \le i \le s \text{ gilt: } P \vdash^d_{BUR} p(c_1, \ldots, c_s) \wedge c_i = a_i$$

3. *eine abgeleitete Formel wird auch abgeleitet in jedem weiteren Inferenzschritt:*

$$P \vdash^{(d+1)}_{BUR} G \qquad \text{, wenn } P \vdash^d_{BUR} G$$

4. *die Anwendung von ISOL ergibt abgeleitete Formeln:*

$$P \vdash^{(d+1)}_{BUR} G \qquad \text{, wenn } [B_1 \wedge \ldots \wedge B_n \rightarrow A] \in P_R \text{ und } P \vdash^d_{BUR} \sigma B_1 \wedge \ldots \wedge P \vdash^d_{BUR} \sigma B_n \wedge \sigma A = G$$

5. *alle in einem Inferenzschritt d abgeleiteten Formeln sind abgeleitete Formeln im BUR-Kalkül:*

$$\forall d : P \vdash^d_{BUR} G \Rightarrow P \vdash_{BUR} G$$

oder verbal gesagt gilt:

$\vdash_{BUR}= \{(P,G) \mid G$ *ist Grundatom durch Anwendung von Prioritätsprinzip 2 auf* $P\}$

Die Äquivalenz der formalen und der verbalen Darstellung in der Definition der Inferenzrelation $\vdash_{BUR}$ ist offensichtlich für die Punkte 1 bis 3 und 5. Für die Äquivalenz der Formel in Punkt 4 zur Evaluierung der isolierten Konnektionen einer Klausel ist eine zusätzliche Annahme erforderlich. Hat man eine Klausel, die eine Variable nur im Klauselkopf enthält, so enthält die Resolvente bei erfolgreicher ISOL-Unifikation auch diese Variable, weil sie nicht substituiert werden kann. Deshalb ist es wichtig, daß eine allquantifizierte Variable genau die Menge der Konstanten repräsentiert.
Die Beschränkungen im BUR-Kalkül sollen, vergleichbar mit SHRUTI (Bedingungen 4.2.1 und 4.3.1), noch einmal zusammengefaßt werden.

Bedingung 5.3.1 *Im BUR-Kalkül gelten folgende Annahmen*

1. *Es gibt keine Funktionszeichen außer Konstanten.*
2. *Alle Variablen in den Formeln des Kalküls sind allquantifiziert.*
3. *Fakten und Anfragen sind atomare Grundformeln.*
4. *Eine allquantifizierte Variable repräsentiert genau die Menge der Konstanten, also die Elemente von* U_P.

Definition 5.3.12 *Wir definieren die Menge* I_d *der abgeleiteten Formeln nach* d *Inferenzschritten im BUR-Kalkül mit*

$$I_d = \{G \mid P \vdash^d_{BUR} G\}$$

5.3.2.1 Einstellige Prädikate

Proposition 5.3.1 *Sei eine Wissensbasis* $P = P_R \cup P_F$ *im BUR-Kalkül gegeben, die nur einstellige Prädikatssymbole enthält. Für die Menge* I_d *der mit* d *Inferenzschritten im Kalkül abgeleiteten Formeln gilt:*

$$I_d = T_P(\emptyset) \uparrow d$$

Beweis: Der Beweis wird induktiv über die Länge der Inferenzkette d geführt. Er arbeitet auf Grundlage der Definition 5.3.11 für die Inferenzrelation $\vdash^d_{BUR}$ und der Definition 5.3.12 für die Menge I_d im Kalkül abgeleiteter Formeln.

- $(\mathbf{d = 1})$ Nach Punkt 1 von 5.3.11 gilt $G \in I_1 \Leftarrow G \in P_F$. Der Punkt 2 sagt für einstellige Atome aus, daß $p(a_1) \in I_1 \Leftarrow p(c_1) \in I_1 \wedge c_1 = a_1$ gilt. Beide Seiten der Implikation sind äquivalent und es folgt $p(a_1) \in I_1 \Leftrightarrow p(a_1) \in I_1$. Das heißt in diesem Fall, daß keine weitere Formel zu I_1 hinzugefügt wird. Weil 1 und 2 die einzigen Bestimmungen für die Tiefe $d = 1$ sind, gilt $I_1 = P_F$ und nach Definition von T_P (2.1.11)

$$I_1 = T_{P_F}(\emptyset) \uparrow 1 = T_P(\emptyset) \uparrow 1$$

- $(\mathbf{d = n \to n+1})$ Sei eine Formelmenge I_n nach n Inferenzschritten gegeben mit $I_n = T_P(\emptyset) \uparrow n$. Es folgt der Inferenzschritt $d = n + 1$.
  Nach Punkt 4 aus Definition 5.3.11 gilt, daß die Menge I_{n+1} alle σA enthält, die Grundinstanz eines Klauselkopfes aus P_R sind, wobei I_n Modell für den Klauselkörper ist. Mit der Definition des Operators T_P ist dies die Berechnung $T_{P_R}(I_n)$, so daß gilt: $G \in I_{(n+1)} \Leftarrow G \in T_{P_R}(I_n)$
  Mit Punkt 3 gilt außerdem: $G \in I_{(n+1)} \Leftarrow G \in I_n$
  Wie bei $(d = 1)$ gezeigt, wird mit Punkt 2 für einstellige Prädikate keine weitere Formel abgeleitet, so daß $G \in I_{n+1} \Leftrightarrow G \in I_n \vee G \in T_{P_R}(I_n)$ gilt. In Mengenschreibweise bedeutet das

  $$I_{n+1} = T_{P_R}(I_n) \cup I_n$$

  Nach Vorraussetzung gilt $I_n = T_P(\emptyset) \uparrow n$ und damit für die neue Menge abgeleiteter Formeln $I_{n+1} = T_{P_R}(T_P(\emptyset) \uparrow n) \cup T_P(\emptyset) \uparrow n$.
  Weil P ein definites logisches Programm ist, ist T_{P_R} monoton und es gilt

  $$I_{n+1} = T_P(\emptyset) \uparrow (n+1)$$

□

Wie bereits erwähnt hat T_P für definite Programme einen Fixpunkt, der mit $T_P \uparrow \infty$ bezeichnet wird. Programme im BUR-Kalkül sind nach Definition definite Programme und folglich hat der BUR-Kalkül für einstellige Prädikate einen Fixpunkt $I_\infty = T_P \uparrow \infty$. Der Fixpunkt von T_P ist nach Proposition 3.1.2 und 3.1.3 gleich der Menge aller Grundatome G, für die $P \models G$ gilt, also der Menge aller logischer Konsequenzen. Daraus folgt sofort:

Theorem 5.3.2 *Der BUR-Kalkül für einstellige Prädikate ist korrekt und vollständig.*

5.3.2.2 Mehrstellige Prädikate

Proposition 5.3.3 *Sei eine Wissensbasis $P = P_R \cup P_F$ im BUR-Kalkül gegeben. Für die Menge I_d der mit d Inferenzschritten im Kalkül abgeleiteten Formeln gilt:*

$$I_d \supseteq T_P(\emptyset) \uparrow d$$

Beweis: Der Beweis wird induktiv über die Länge der Inferenzkette d geführt.

- $(\mathbf{d = 1})$ Wie im Beweis für Proposition 5.3.1 wird gezeigt, daß I_1 alle Fakten P_F enthält. Nach Punkt 2 der Definition 5.3.11 gilt zum Beispiel für ein Prädikat p der

Stelligkeit $s = 2$, daß $p(a_1, a_2)$ abgeleitet wird, wenn $p(a_1, a_3)$ und $p(a_3, a_2)$ Fakten sind. Zur Menge I_1 werden unter Umständen also noch Formeln hinzugefügt. Es kann der bereits erläuterte CROSS-TALK auftreten. Damit gilt offensichtlich:

$$I_1 \supseteq T_P(\emptyset) \uparrow 1$$

- $(\mathbf{d = n \rightarrow n + 1})$ Sei eine Menge I_n nach n Inferenzschritten gegeben mit $I_n \supseteq T_P(\emptyset) \uparrow n$. Es folgt der Inferenzschritt $d = n + 1$.
  Analog zum Beweis für einstellige Prädikate gilt $G \in I_{(n+1)} \Leftarrow G \in T_{P_R}(I_n)$ und $G \in I_{(n+1)} \Leftarrow G \in I_n$.
  Wie im Schritt $(d = 1)$ werden nach Punkt 2 durch CROSS-TALK weitere Formeln abgeleitet, so daß folgt:
  $$I_{n+1} \supseteq T_{P_R}(I_n) \cup I_n$$
  Weil P ein definites logisches Programm ist, ist T_{P_R} monoton und es gilt
  $$I_{n+1} \supseteq T_P(\emptyset) \uparrow (n + 1)$$

□

Die Berechnung von I_d ist nach Definition eine monotone Abbildung und hat einen Fixpunkt I_∞, der eine Übermenge von $T_P \uparrow \infty$ repräsentiert. Der Fixpunkt enthält damit nach Proposition 3.1.2 und 3.1.3 alle Grundatome F, für die gilt $P \models F$, aber möglicherweise noch weitere. Daraus folgt sofort:

Theorem 5.3.4 *Der BUR-Kalkül im allgemeinen ist vollständig, aber nicht notwendigerweise korrekt.*

5.3.3 Beweisführung

Mit den nachgewiesenen Eigenschaften des BUR-Kalküls ist es nun wichtig, ein Theorem zu finden, das eine Aussage über die Korrektheit abgeleiteter Formeln liefert. Wenn für eine bestimmte Anzahl von Inferenzschritten entschieden werden könnte, daß die abgeleiteten Formeln garantiert logische Konsequenzen einer Wissensbasis P im Sinne der Prädikatenlogik sind, so könnte eine große Zahl von Formeln bewiesen und die Menge der mit einem Rückwärtsschließer zu beweisenden Anfragen enorm eingeschränkt werden. Für ein solches Vorgehen machen wir uns zunutze, daß nur bei der punktweisen Anwendung von DB eine nicht gültigkeitserhaltende Transformation stattfindet. Nur wenn dabei CROSS-TALK entsteht, werden Formeln abgeleitet, die keine korrekten Schlußfolgerungen sind. Für das Auftreten von CROSS-TALK betrachten wir die Menge von Bindungen in einem Argument jedes Prädikates.

Definition 5.3.13 *Seien P eine Menge von Formeln im BUR-Kalkül und p ein Prädikatssymbol mit der Stelligkeit s. Wir definieren die Menge der Bindungen $\mathbf{p}_i^d$ von p im Argument i $(1 \leq i \leq s)$ nach dem Inferenzschritt d mit*

$$\mathbf{p}_i^d = \{a \mid P \vdash^d_{BUR} p(c_1, \ldots, c_s) \wedge c_i = a\}$$

Damit kommen wir nun zu einer Feststellung, wann die Elemente der Inferenzrelation $\vdash_{BUR}$ garantiert als korrekte Folgerungen betrachtet werden können.

Theorem 5.3.5 (Korrektheitstheorem) *Sei P eine Wissensbasis im BUR-Kalkül. Wenn nach d Inferenzschritten alle Prädikatszeichen p der Bedingung*

$$\forall i, j : |\mathbf{p}_i^d| > 1 \wedge i \neq j \Rightarrow |\mathbf{p}_j^d| = 1$$

genügen, dann gilt für jede Grundinstanz G

$$P \vdash_{BUR}^{d} G \Rightarrow P \models G.$$

Das heißt, daß alle bis zum Schritt d abgeleiteten Formeln tatsächlich logische Konsequenzen von P sind.

Beweis: Der Beweis für Proposition 5.3.3 hat gezeigt, daß allein Punkt 2 der Definition 5.3.11 aussschlaggebend dafür ist, ob $I_d = T_P(\emptyset) \uparrow d$ oder $I_d \supseteq T_P(\emptyset) \uparrow d$ gilt. Mit der punktweisen Anwendung von DB gilt in einer Tiefe d:

$$\begin{aligned} p(a_1, \ldots, a_s) \in I_d \quad \Leftarrow \quad & p(c_{11}, \ldots, c_{1s}) \in I_d \wedge c_{11} = a_1 \wedge \ldots \wedge \\ & p(c_{s1}, \ldots, c_{ss}) \in I_d \wedge c_{ss} = a_s \end{aligned}$$

Die Bedingung sagt, daß alle abgeleiteten Formeln von p nur in einer Stelle i verschiedene Bindungen haben. Deshalb gilt für alle $j \neq i$ die Gleichung $c_{ij} = c_{1j} = \ldots = c_{sj}$. Damit vereinfacht sich die Formel auf

$$\begin{aligned} p(a_1, \ldots, a_s) \in I_d \quad \Leftarrow \quad & p(c_{i1}, \ldots, c_{ii}, \ldots, c_{is}) \in I_d \wedge c_{i1} = a_1 \wedge \ldots \wedge c_{is} = a_s \\ \Leftrightarrow \quad & p(a_1, \ldots, a_s) \in I_d \end{aligned}$$

Beide Seiten sind äquivalent und es kann zu I_d keine zusätzliche Formel hinzugefügt werden. Es entsteht kein CROSS-TALK und $I_d = T_P(\emptyset) \uparrow d$ gilt. Weil T_P für definite Programme monoton ist, gilt auch $I_d \subseteq T_P \uparrow \infty$. Der Fixpunkt von T_P ist die Menge aller logischen Konsequenzen und das Theorem ist bewiesen. □

Mit dem gegebenen Theorem hat man nun ein echtes Werkzeug zum Führen von Beweisen. Man weiß, daß der BUR-Kalkül vollständig ist und hat damit eine obere Schranke für die Menge der logischen Konsequenzen. Bezeichnen wir also die Menge der abgeleiteten Formeln mit $\mathbf{G}_v = I_\infty$, so enthält $\mathbf{G}_v$ alle korrekten Folgerungen. Außerdem kann durch Überwachen jedes Inferenzschrittes eine Menge von Formeln $\mathbf{G}_k$ ermittelt werden, die garantiert logische Konsequenzen aus der Wissensbasis P sind. Die Differenz zwischen $\mathbf{G}_v$ und $\mathbf{G}_k$ bilden genau die Formeln, die erst in einem Inferenzschritt d abgeleitet wurden, in dem die Bedingung von Theorem 5.3.5 verletzt wurde. Über diese Formeln $\mathbf{G}_v \setminus \mathbf{G}_k$ kann ohne andere Beweise keine Aussage bezüglich der Korrektheit getroffen werden. Dies ist sozusagen die Restmenge der Formeln, über die man noch nicht weiß, ob sie logische Konsequenzen sind. Im Idealfall gilt $\mathbf{G}_v = \mathbf{G}_k$, das heißt alle im Kalkül abgeleiteten Formeln sind korrekt. In der worst-case-Betrachtung ist der Fixpunkt der

Inferenz die komplette Herbrand-Basis und die Menge $\mathbf{G}_k$ ist leer, was bedeuten würde, daß für alle Elemente von B_P keine Korrektheitsentscheidung möglich ist.
Zum Beweisen der Formeln der Restmenge kann man im wesentlichen zwei Strategien anwenden. Zum einen besteht die Möglichkeit, mit einer Teilmenge von P und einer Teilmenge, der als korrekt bewiesenen Folgerungen, ein Programm P' zu erstellen und für dieses erneut eine überwachte Berechnung mit dem BUR-Kalkül durchzuführen. Man ermittelt eine Menge $\mathbf{G}'_k$ und die zu beweisende Restmenge reduziert sich auf $\mathbf{G}_v \setminus \mathbf{G}_k \setminus \mathbf{G}'_k$. Durch geschicktes Kombinieren der Regeln, Fakten und korrekten Folgerungen und mehrere derartiger Berechnungen kann deshalb möglicherweise jede Formel in $\mathbf{G}_v$ auf diesem Weg bewiesen werden. Dabei sollte man sinnvollerweise darauf achten, daß nicht schon die aktivierten Fakten P'_F die Bedingung des Korrektheitstheorems 5.3.5 verletzen, da sonst $\mathbf{G}'_k = \emptyset$ gilt. Bei dem beschriebenen Vorgehen kann jedoch nicht für eine Formel nachgewiesen werden, daß sie nicht logische Konsequenz ist. Außerdem ist im schlechtesten Fall wiederum festzustellen, daß keine der gewählten Kombinationen eine Formel der Restmenge beweist. In der Betrachtung des average-case ist zu vermuten, daß einige Formeln bewiesen werden können, jedoch eine große Zahl an unklaren Ableitungen übrigbleibt.
Dann hilft als zweites nur die Möglichkeit diese Formeln mit einem rückwärtsschließenden Verfahren zu beweisen oder zu wiederlegen. Dazu kann über der Restmenge $\mathbf{G}_v \setminus \mathbf{G}_k \setminus \mathbf{G}'_k \setminus \ldots$ eine SHRUTI-Inferenz oder eine Resolution angewendet werden, oder mit der Konnektionsmethode Beweise durchgeführt werden. Aus der Syntax der Formeln und dem Verhältnis zu Reduktionen wird klar, daß der Zeitaufwand für den Beweis einer Formel nicht schlechter als parallel linear sein muß. Die Anzahl der Formeln bzw. Anfragen ist jedoch exponentiell. Im schlechtesten Fall beträgt sie B_P mit einer Anzahl von $O(D \times C^{\bar{s}})$ Elementen, wobei D die Anzahl der Prädikate, C die Anzahl der Konstanten und $\bar{s}$ die mittlere Stelligkeit der Prädikate bezeichnet. Der Aufwand zum Ermitteln aller korrekten Formeln ist deshalb auch exponentiell.
Ein average-case-Betrachtung ist schwierig. Vermutlich gilt jedoch daß $\mathbf{G}_k \neq \emptyset$, $\mathbf{G}_v \neq B_P$ und durch Anwendung der ersten Strategie kann $\mathbf{G}_v \setminus \mathbf{G}_k$ noch soweit reduziert werden, daß der weitere Zeitaufwand nur noch polynomial, quadratisch oder sogar linear ist, weil es die Anzahl verbleibender Restformeln ist. Dieser Durchschnittsfall verspricht eine Lösung mit akzeptablem Zeitaufwand für eine Vielzahl von Problemen. Es wäre interessant, ob dies gerade solche Probleme sind, die der realen Welt bzw. dem Alltagsschließen entsprechen. Wenn das tatsächlich der Fall wäre, dann bietet der BUR-Kalkül in Kombination mit Rückwärtsschließern eine sehr gute Methode zum Beweisen mit linearem Aufwand an Ressourcen und polynomialem Zeitaufwand.

6. Ein RNN-Modell mit dynamischer Bindung

Es wird nun für den in Abschnitt 5.3 definierten BUR-Kalkül ein massiv paralleles Berechnungsmodell erstellt. Die Kodierung erfolgt entsprechend der Inferenzrelation des Kalküls, so daß tatsächlich die beschriebene Form der Reduktionen realisert wird. Das Modell hat demzufolge linearen Platzbedarf und parallel linearen Zeitbedarf und realisiert vollständige, aber nicht notwendigerweise korrekte Inferenzen. Als Basis dient eine konnektionistische Struktur nach dem Vorbild des RNN-Modelles[1]. Unter Verwendung des SHRUTI-Konzeptes der dynamischen Bindung mit synchroner Aktivität (Abschnitt 4.1.2) erfolgt die Darstellung der prädikatenlogischen Formeln.
Im Abschnitt 6.1 werden die Grundlagen für die dynamischen Berechnungen im Netz dargelegt. Dazu gehören die verwendeten Netzeinheiten und eine formale Darstellung ihrer Funktion. Das neue RNN-Modell und seine Kodierung zeigt 6.2. Abschnitt 6.3 führt eine genaue Aufwandsabschätzung bezüglich Platz- und Zeitkomplexität durch und in Abschnitt 6.4 wird die Beweisführung im BUR-Kalkül mittels Inferenzen im RNN-Modell demonstriert. Das Kapitel endet in 6.5 mit der Darstellung eines Beispiels aus dem reflexiven Schließen und dem Vergleich mit dem entsprechenden SHRUTI-Modell. Das neue Modell mit dynamischer Bindung wird im folgenden auch kurz als RNN-Modell bezeichnet.

6.1 Zeitformalismus

Das Prinzip synchroner Aktivierung wurde in SHRUTI bisher mit den Begriffen Phase und Zyklus nur verbal dargelegt. Für die Untersuchung der Eigenschaften des Berechnungsmodells ist es notwendig, dies zu formalisieren. Ein Zyklus wird im folgenden durch ω und eine Phase durch π repräsentiert[2]. Beide Symbole stellen Zeitintervalle, also Mengen der Form

$$\begin{aligned} \omega &= [\omega_0, \omega_1] \\ \pi &= [\pi_0, \pi_1] \end{aligned}$$

[1]siehe Kapitel 3

[2]Dieses π hat nichts mit der Datenbankoperation Projektion zu tun.

dar. Während des Inferenzvorganges korrespondiert die Aktivierung einer bestimmten Phase zur Bindung an eine Konstante, während die Länge einer Phase keine wesentliche Rolle in den Betrachtungen spielt. Der Einfachheit halber sollen deshalb nur die Mengen betrachtet werden und die Phase, die zu einer Konstante $c \in U_P$ korrespondiert, soll kurz mit π_c bezeichnet werden. Mathematisch gesehen betrachten wir also den Zyklus ω als endliche Menge diskreter Werte, nämlich der Phasen $\{\pi_c \mid c \in U_P\}$. Damit diese vereinfachte Darstellung überhaupt möglich wird, müssen für jeden beliebigen Zeitpunkt t zwei Forderungen erfüllt sein:

- Alle existierenden Phasen liegen im Zyklus.

$$t \in \pi_c \Rightarrow t \in \omega \text{ für alle t,}\pi_c$$

- Zwei unterschiedliche Phasen haben unterschiedliche Zeitintervalle.

$$t \in \pi_c \wedge \pi_c \neq \pi_{c'} \Rightarrow t \notin \pi_{c'} \text{ für alle t,}\pi_c\text{,}\pi_{c'}$$

Für die Darstellung der zeitbezogenen Aktivierung verwenden wir zwei Typen von Schwellwerteinheiten, die im folgenden kurz mit SWEP (Schwellwerteinheit mit Phasenausgabe) und SWEZ (Schwellwerteinheit mit Zyklusausgabe) bezeichnet werden. Abbildung 6.1 zeigt die Einheiten und deren Funktionsberechnung. Beide besitzen einen internen Schwellwert θ und werden aktiviert, wenn die Summe aller eingehenden Aktivierungen in einer Phase den Schwellwert erreicht. Schwellwerteinheiten mit Phasenausgabe geben eine Aktivierung in genau den Phasen aus, in denen dies passiert. Eine aktivierte SWEZ gibt eine Aktivierung über einem ganzen Zyklus aus. Wie sich später am Modell zeigen wird, lassen sich durch Kombinationen dieser beiden Einheiten die benötigten Funktionen realisieren.

Die Gesamteingabe ζ und die Ausgabe ξ der Einheiten definieren wir über den Zeiträumen π_c und ω. Die Eingabe ζ einer Einheit berechnet sich aus der Summe aller eingehenden Aktivierungen.

$$\zeta(\pi_c) = \sum_{\text{für alle i}} \zeta_i(\pi_c)$$

Die Ausgabe einer SWEP wird berechnet nach:

$$SWEP(\pi_c) = \xi(\pi_c) = 1 \Leftrightarrow \zeta(\pi_c) \geq \theta \tag{6.1}$$

Auch ein SWEZ-Knoten wird aktiviert, wenn der Schwellwert in einer Phase überschritten wird. Er gibt jedoch in aktiviertem Zustand einen Wert von 1 über dem ganzen Zyklus aus. Wir definieren deshalb seine Ausgabe über dem Zyklus ω und über einer beliebigen Phase π_c.

$$SWEZ(\omega) = \xi(\omega) = 1 \Leftrightarrow \exists c' : \zeta(\pi_{c'}) \geq \theta \tag{6.2}$$

$$SWEZ(\omega) = \xi(\pi_c) = 1 \Leftrightarrow \exists c' : \zeta(\pi_{c'}) \geq \theta \tag{6.3}$$

Bemerkung:
Im Modell werden wir später von der Annahme ausgehen, daß für das Propagieren von Aktivierungen über eine Netzschicht genau die Zeit $\Delta\omega$ benötigt wird. Damit ist auch die Zeit zur Berechnung der Ausgabe einer einzelnen Einheit gleich $\Delta\omega$.

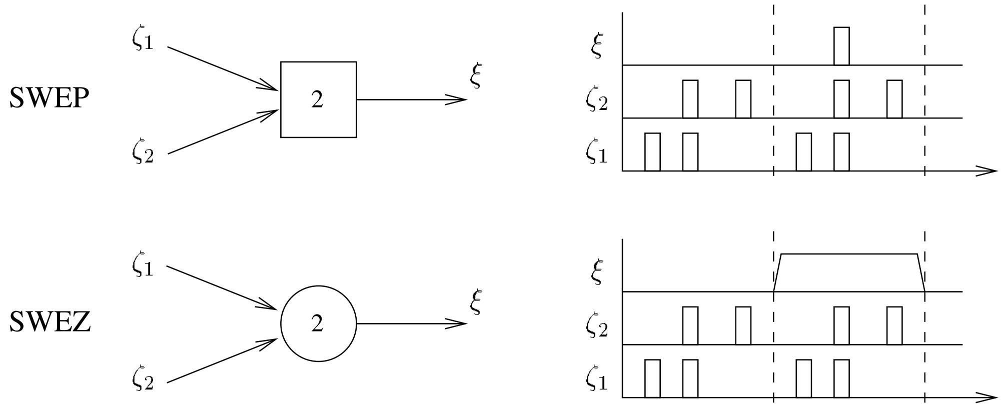


Abbildung 6.1: Die zwei Typen von Netzeinheiten und ihre Funktion. Es sind jeweils Einheiten mit einem Schwellwert $\theta = 2$ und 2 Eingaben ζ_1 und ζ_2 dargestellt. Die Ausgabe ξ der Knoten bei einer gegebenen Eingabe zeigen die Diagramme.

6.2 Aufbau des Netzes

Wiederholen wir zunächst noch einmal die Vorraussetzungen aus dem BUR-Kalkül.
Alle Fakten und abgeleiteten Formeln sind Grundatome. Mit der Inferenzrelation soll also eine Funktion über Interpretationen realisiert werden, die im allgemeinen eine Übermenge des Operators T_P ergibt. Aus diesem Grund gehen wir von einem ähnlichen Grundaufbau des Berechnungsmodells aus, wie beim RNN-Modell für Aussagenlogik. Das heißt, das neuronale Netz hat eine Eingabeschicht, eine Ausgabeschicht und eine noch nicht festgelegte Anzahl verdeckter Schichten zwischen ihnen. Die beiden äußeren Schichten repräsentieren jeweils Interpretationen. Außerdem bestehen rekurrente Verbindungen von jeder Einheit der Ausgabeschicht zu der korrespondierenden Einheit der Eingabeschicht, so daß das Modell eine iterative Berechnung durchführt. Der weitere Aufbau des konnektionistischen Modelles wird durch die Definition der Inferenzrelation $\vdash^d_{BUR}$ (siehe Punkte 1-5 aus Definition 5.3.11) beeinflußt:

1. Die Definition sagt aus, daß Fakten im ersten Inferenzschritt abgeleitet werden. Beim RNN-Modell für Aussagenlogik wird das realisiert, indem Fakten als Klauseln in den internen Schichten kodiert werden. Dann kann ein einmal konstruiertes Netz jedoch nur für eine Berechnung verwendet werden. Für das neue RNN-Modell wird deshalb die Festlegung getroffen, das Faktenwissen P_F als Eingabe an das Netz anzulegen und lediglich die Regeln P_R der Wissensbasis P in den internen Schichten zu kodieren. Durch diese Entscheidung liegt auch die Menge der abgeleiteten Formeln an der Eingabeschicht an.

2. Die punktweise Anwendung von DB wird am einfachsten dadurch realisiert, daß jede Stelle eines Prädikates durch eine Netzeinheit repräsentiert wird, die die Menge aller Bindungen in diesem Punkt darstellt. Die abgeleiteten Formeln eines Prädikates sind dann das Kreuzprodukt aus diesen Mengen.
   Weil im BUR-Kalkül keine existenzquantifizierten Variablen auftreten, ist jede Einheit garantiert in einer Phase gebunden, wenn eine Instanz des Prädikates aktiv ist. Die Darstellung einer Instanzeinheit, ähnlich der des Collectors in SHRUTI, ist deshalb nicht notwendig. Deren Information ist implizit auch im Argumentknoten enthalten. Jedes Prädikat mit der Stelligkeit s wird aus diesen Gründen im neuen RNN-Modell mit genau s Einheiten kodiert.

3. Die Definition beschreibt, daß einmal abgeleitete Formeln auch in allen folgenden Inferenzschritten abgeleitet werden. Um dies für die Fakten P_F zu realisieren, muß die Festlegung getroffen werden, daß Fakten in jedem Berechnungsschritt des Modelles an der Eingabeschicht angelegt werden. Dieses zyklische Aktivieren wird auch als „clampen“ bezeichnet. Damit wird gleichzeitig erreicht, daß in jedem Berechnungsschritt die Ableitung aus den Fakten erfolgt, praktisch die Inferenzkette neu beginnt. Damit ist der Punkt 3 für alle abgeleiteten Formeln erfüllt.

4. Dieser Punkt verlangt die Ableitung von Formeln aus der gegebenen Interpretation auf der Basis des Regelwissens. Dies wird wie im RNN für Aussagenlogik in den inneren Schichten des Modelles realisiert. Mit der Festlegung in Punkt 1 ist die Interpretation an der Ausgabeschicht eine Übermenge der Berechnung von T_{P_R}. Die Menge, der im BUR-Kalkül abgeleiteten Formeln, wird durch die Eingabeschicht dargestellt. An dieser Belegung erfolgt auch die Entscheidung über die Erfüllung des Korrektheitstheorems.

5. Der letzte Definitionspunkt hat keine direkten Auswirkungen auf das Modell. Im Zusammenhang mit Punkt 3 kann lediglich festgestellt werden, daß alle im BUR-Kalkül abgeleiteten Formeln im Fixpunkt der Berechnung an der Eingabeschicht repräsentiert werden.

Unter den gegebenen Festlegungen wird nun die Kodierung von Interpretationen nach dem Vorbild SHRUTI definiert. Dabei wird eine bestimmte Notation für die Bezeichnung der Einheiten verwendet. Ein Netzknoten wird entweder durch seinen Typ (SWEP, SWEZ) oder durch die Ausgabe ξ repräsentiert. Der hochgestellte Index I sagt, daß es sich um eine Eingabeeinheit handelt. Die tiefgestellte Zeichenkette bezeichnet die Konstante oder die Stelle eines Prädikates, zu der die Einheit gehört.

- Jede Einheit in der Eingabeschicht, die zu einem Konstantensymbol a korrespondiert, ist in einer Phase genau dann aktiviert, wenn die Phase die Konstante repräsentiert. Sie stellt damit die Menge $\{a\}$ dar.

$$a = c \Leftrightarrow \xi_a^I(\pi_c) = 1 \tag{6.4}$$

- Eine Prädikatsinstanz ist Element der Interpretation I, wenn die korrespondierenden Einheiten des Prädikatssymbols jeweils eine Bindung repräsentieren. Jede Einheit $\xi^I_{p_i}$ $(1 \leq i \leq s)$ steht für die Menge $\mathbf{p}_i$.

$$p(c_1, \ldots, c_s) \in I \Leftarrow \xi^I_{p_1}(\pi_{c_1}) = 1 \wedge \ldots \wedge \xi^I_{p_s}(\pi_{c_s}) = 1 \tag{6.5}$$

- Mit den Gleichungen 6.4 und 6.5 lautet das Prinzip der synchronen Aktivierung:

$$\begin{aligned} p(a_1, \ldots, a_s) \in I \quad \Leftarrow \exists c_1, \ldots, c_s : \quad & \xi^I_{p_1}(\pi_{c_1}) = 1 \wedge \xi^I a_1(\pi_{c_1}) = 1 \wedge \ldots \wedge \\ & \xi^I_{p_s}(\pi_{c_s}) = 1 \wedge \xi^I a_s(\pi_{c_s}) = 1 \end{aligned} \tag{6.6}$$

Bemerkung: Der Unterschied zwischen c und a liegt darin, daß a ein konkretes Element von U_P, also eine Konstante auf der Symbolebene darstellt. Der Bezeichner c steht lediglich für eine beliebige Konstante, die durch eine der Phasen kodiert wird.

Für die genaue Realisierung des BUR-Kalküls muß außerdem die Annahme 4 aus Bedingung 5.3.1 gewährleistet werden. Dafür müssen während des Inferenzvorganges alle Konstanten zyklisch in einer separaten Phase aktiviert werden.

- Jede Phase π_c im Zyklus ω repräsentiert eine Konstante $c \in U_P$ bzw. die Menge aller Phasen repräsentiert genau U_P.

$$\pi_c \in \omega \Leftrightarrow c \in U_P \tag{6.7}$$

Zum Aufbau des RNN-Modelles fehlt nun noch die Implementierung des Regelwissens in den inneren Schichten. Sie soll im folgenden an einem Beispiel dargelegt werden.

Beispiel 6.2.1 *Sei eine Klausel K der Form*

$$p(X, Y) \wedge q(Y, X) \rightarrow r(X)$$

gegeben. Die Menge der Bindungen in einer Stelle eines Prädikates sei nach Definition 5.3.13 beschrieben, wobei der Inferenzschritt d hier außer Acht gelassen werden kann. Die parallele Anwendung von ISOL auf diese Mengen ist eine Kombination von Join, kartesischem Produkt und Projektion. Der natürliche Verbund über einstelligen Relationen entspricht dabei der Durchschnittsbildung. Für die Bindung von r ergibt sich folgende Gleichung[3]*:*

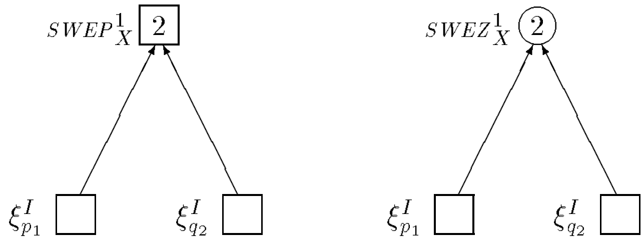
$$\mathbf{r} = \pi_{\mathbf{X}}((\mathbf{p_1} \cap \mathbf{q_2}) \otimes (\mathbf{p_2} \cap \mathbf{q_1}))$$

Das Ergebnis des kartesischen Produkt ist nach Definition die leere Menge, wenn eine der beteiligten Relationen leer ist. Das bedeutet:

$$\begin{aligned} \mathbf{r} \quad &= \begin{cases} \emptyset & \textit{wenn } (\mathbf{p_1} \cap \mathbf{q_2}) = \emptyset \vee (\mathbf{p_2} \cap \mathbf{q_1}) = \emptyset \\ \mathbf{p_1} \cap \mathbf{q_2} & \textit{sonst} \end{cases} \\ &= \begin{cases} \mathbf{p_1} \cap \mathbf{q_2} & \textit{wenn } (\mathbf{p_1} \cap \mathbf{q_2}) \neq \emptyset \wedge (\mathbf{p_2} \cap \mathbf{q_1}) \neq \emptyset \\ \emptyset & \textit{sonst} \end{cases} \end{aligned} \tag{6.8}$$

[3] π_X steht hier für die Projektion der an X gebundenen Elemente

Es gilt nun, die Berechnung der Gleichung 6.8 mit den bereits vorgestellten Netzeinheiten und der definierten Kodierung zu realisieren. Nach Gleichung 6.5 wird jede der dargestellten Mengen von einer Einheit in der Eingabeschicht repräsentiert. Diese Einheiten sind vom Typ SWEP, weil sie eine phasenorientierte Ausgabe haben müssen.
Es wird in einer neuen Schicht für die Variable X eine Einheit $SWEP^1_X$ und eine Einheit $SWEZ^1_X$ erstellt. Beide Einheiten erhalten Verbindungen von den Eingabeeinheiten, die zu den Stellen gehören, an die X gebunden ist. Der Schwellwert der Einheiten ist gleich der Anzahl ihrer Verbindungen. Folgende Verbindungen kommen zustande[4].

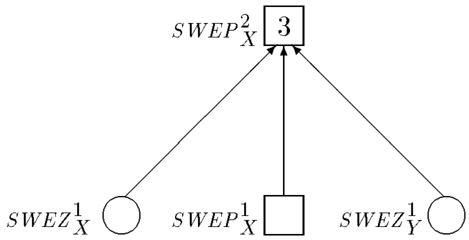


Mit den Funktionen der Netzeinheiten nach den Gleichungen 6.1 und 6.3 ergibt sich:

$$\begin{aligned} SWEP^1_X(\pi_c) = 1 \quad &\Leftrightarrow \quad \xi^I_{p_1}(\pi_c) = 1 \wedge \xi^I_{q_2}(\pi_c) = 1 \\ &\Leftrightarrow \quad c \in (\mathbf{p_1} \cap \mathbf{q_2}) \end{aligned}$$

$$\begin{aligned} SWEZ^1_X(\pi_c) = 1 \quad &\Leftrightarrow \quad \exists c' : \xi^I_{p_1}(\pi_{c'}) = 1 \wedge \xi^I_{q_2}(\pi_{c'}) = 1 \\ &\Leftrightarrow \quad (\mathbf{p_1} \cap \mathbf{q_2}) \neq \emptyset \end{aligned}$$

Eine für die Variable Y erstellte $SWEZ^1_Y$, die nach dem gleichen Schema mit den Einheiten von p_2 und q_1 verbunden ist und den Schwellwert 2 hat, berechnet analog dazu $(\mathbf{p_2} \cap \mathbf{q_1}) \neq \emptyset$. Betrachten wir nun noch einmal die zu realisierende Gleichung 6.8, dann fällt auf, daß mit den auf diesem Weg erstellten Einheiten genau alle benötigten Teilfunktionen realisiert werden. Jetzt müssen deren Ausgaben nur noch so verknüpft werden, das eine Einheit die Bindung für r repräsentiert. Dazu ist eine weitere SWEP-Einheit in einer neuen Netzschicht notwendig. Sie wird mit $SWEP^2_X$ bezeichnet und darf nur dann aktiviert sein, wenn die drei Einheiten der ersten Schicht aktiviert sind. Das ist eine Art Konjunktion. Alle SWEZ der 1. verdeckten Schicht und die $SWEP^1_X$ werden mit der neuen Einheit verbunden und der Schwellwert ist gleich der Anzahl der Verbindungen.



[4]Die Eingabeeinheiten sind nur aus Gründen der Übersichtlichkeit doppelt dargestellt.

Die neue Einheit berechnet demzufolge

$$SWEP_X^2(\pi_c) = 1 \Leftrightarrow c \in (\mathbf{p_1} \cap \mathbf{q_2}) \wedge (\mathbf{p_1} \cap \mathbf{q_2}) \neq \emptyset \wedge (\mathbf{p_2} \cap \mathbf{q_1}) \neq \emptyset$$

und repräsentiert damit über einem Zyklus genau die Bindung $\mathbf{r}$ für diese Klausel. Möglicherweise existieren weitere Klauseln, die das Prädikat r im Klauselkopf enthalten. Dann müssen die Bindungen der abgeleiteten Formeln vereinigt werden, damit eine Einheit die Gesamtmenge $\mathbf{r}$ der Bindungen des Prädikates darstellt. Deshalb wird der Knoten $SWEP_X^2$ mit der Einheit $SWEP_r^O$ in der Ausgabeschicht verbunden. Die Ausgabeeinheit hat den Schwellwert 1 und ist aktiv, wenn eine ihrer Eingaben in einer Phase aktiviert ist. Sie realisiert damit eine Disjunktion.
Die rekurrente Verbindung von $SWEP_r^O$ zu $SWEP_r^I$ sorgt dafür, daß die abgeleiteten Formeln wieder im nächsten Inferenzschritt beteiligt sind.

Mit den Erkenntnissen des beschriebenen Beispiels soll nun der Konstruktionsalgorithmus für das 4-schichtige vorwärtsgerichtete neuronale Netz des dynamischen RNN-Modells festgehalten werden.

Konstruktionsalgorithmus:

- Erstelle für jede Konstante a eine Einheit $SWEP_a^I$ mit dem Schwellwert $\theta = 1$ in der Eingabeschicht des Netzes.
- Erstelle für jedes Prädikat p mit der Stelligkeit s jeweils s Einheiten $SWEP_{p_k}^I$ und $SWEP_{p_k}^I$ $(1 \leq k \leq s)$ in der Eingabeschicht und der Ausgabeschicht des Netzes. Der Schwellwert der Einheiten ist $\theta = 1$.
- Sei $K\colon \forall X_1, X_2, \ldots, X_m[B_1(\ldots) \wedge B_2(\ldots) \wedge \ldots \wedge B_n(\ldots) \rightarrow A(\ldots)]$ eine Programmklausel in P_R, dann führe folgende Schritte aus:
  - Erstelle für jede Variable X_i, die im Klauselkörper von K auftritt, eine Einheit $SWEZ_{X_i}^1$ in der 1. verdeckten Schicht. Verbinde die Einheit jedes Argumentes p_k, an das X_i im Klauselkörper gebunden ist, mit dieser Einheit. Der Schwellwert θ ist gleich der Anzahl dieser Verbindungen.
  - Erstelle für jede Konstante a, die in K auftritt, eine Einheit $SWEZ_a^1$. Verbinde die Einheit jedes Argumentes p_k, an das a im Klauselkörper gebunden ist, und die Einheit von a in der Eingabeschicht mit $SWEZ_a^1$. Der Schwellwert θ dieser Einheit ist gleich der Anzahl ihrer Verbindungen.
  - Erstelle für jede Einheit $SWEZ_{X_i}^1$ oder $SWEZ_a^1$ aus den beiden vorherigen Schritten eine „Zwillingseinheit" vom Typ $SWEP$, wenn die Variable X_i bzw. die Konstante a auch im Klauselkopf von K auftritt. Die Zwillingseinheit hat Verbindungen von denselben Eingabeeinheiten und denselben Schwellwert.
  - Sei s die Stelligkeit des Atoms $A = p(\ldots)$ im Kopf der Klausel. Dann erstelle genau s Einheiten $SWEP_{p_k}^2$ $(1 \leq k \leq s)$ in der 2. verdeckten Schicht des Netzes. Verbinde alle SWEZ der 1. verdeckten Schicht mit jeder dieser Einheiten.

- Ist die Variable X_i oder die Konstante a an der Stelle k im Atom A gebunden, so erstelle eine Verbindung von der Einheit $SWEP^1_{X_i}$ bzw. $SWEP^1_a$ zu der Einheit $SWEP^2_{p_k}$. Der Schwellwert der Einheit für p_k in der 2. Schicht wird gleich der Anzahl ihrer Verbindungen gesetzt.
- Erstelle eine Verbindung von der Einheit $SWEP^2_{p_k}$ zu der Einheit $SWEP^O_{p_k}$ in der Ausgabeschicht für alle $1 \leq k \leq s$.

Beispiel 6.2.2 *Sei folgendes logische Programm P gegeben:*

$$\begin{aligned} &\rightarrow p(a,b) \\ p(a,b) &\rightarrow q(a,b) \\ q(X,Y) &\rightarrow p(Y,X) \\ p(X,Y) \wedge q(Y,X) &\rightarrow r(X) \end{aligned}$$

Die Berechnung von T_P ergibt:

$$\begin{aligned} T_P(\emptyset) \uparrow 0 &= \emptyset \\ T_P(\emptyset) \uparrow 1 &= \{p(a,b)\} \\ T_P(\emptyset) \uparrow 2 &= \{p(a,b), q(a,b)\} \\ T_P(\emptyset) \uparrow 3 &= \{p(a,b), q(a,b), p(b,a)\} \\ T_P(\emptyset) \uparrow 4 &= \{p(a,b), q(a,b), p(b,a), r(b)\} \\ T_P(\emptyset) \uparrow 5 &= \{p(a,b), q(a,b), p(b,a), r(b)\} = lfp(T_P) \end{aligned}$$

Die Wissensbasis besteht aus dem Faktenwissen $P_F = \{p(a,b)\}$ und den Regeln $P_R = \{p(a,b) \rightarrow q(a,b); q(X,Y) \rightarrow p(Y,X); p(X,Y) \wedge q(Y,X) \rightarrow r(X)\}$. Für die Menge der Regeln wird nach dem gegebenen Konstruktionsalgorithmus das RNN-Modell erstellt. Abbildung 6.2 zeigt den Aufbau des vorwärtsgerichteten Netzes. Die Klausel aus Beispiel 6.2.1 ist auch in P_R enthalten.

Bei der Betrachtung des Netzes fällt auf, daß einige Einheiten und Verbindungen redundant sind. Dies gilt insbesondere für alle SWEP der ersten verdeckten Schicht, die einen Schwellwert von 1 haben. Sie berechnen stets dieselbe Ausgabe, wie ihr Vorgänger. Solche Einheiten können aus dem Netz entfernt werden. Es existieren weitere Möglichkeiten, das vorwärtsgerichtete Netz zu minimieren. Beim Entfernen von Verbindungen müssen dazu aber auch Schwellwerte verändert werden. Die Äquivalenz der vereinfachten Netze läßt sich mit der Funktion der Netzeinheiten beweisen. Man kann aber auch ausgehend von der zur Klausel korrespondierenden Datenbankoperation das Netz schon vereinfacht aufbauen.
Der Vorteil der Minimierung des Netzes ist, daß das Modell dadurch deutlich einfacher und übersichtlicher wird. Mit der Anzahl der Einheiten sinkt außerdem der Berechnungsaufwand bzw. die Anzahl der für die Parallelverarbeitung benötigten Prozessoren.

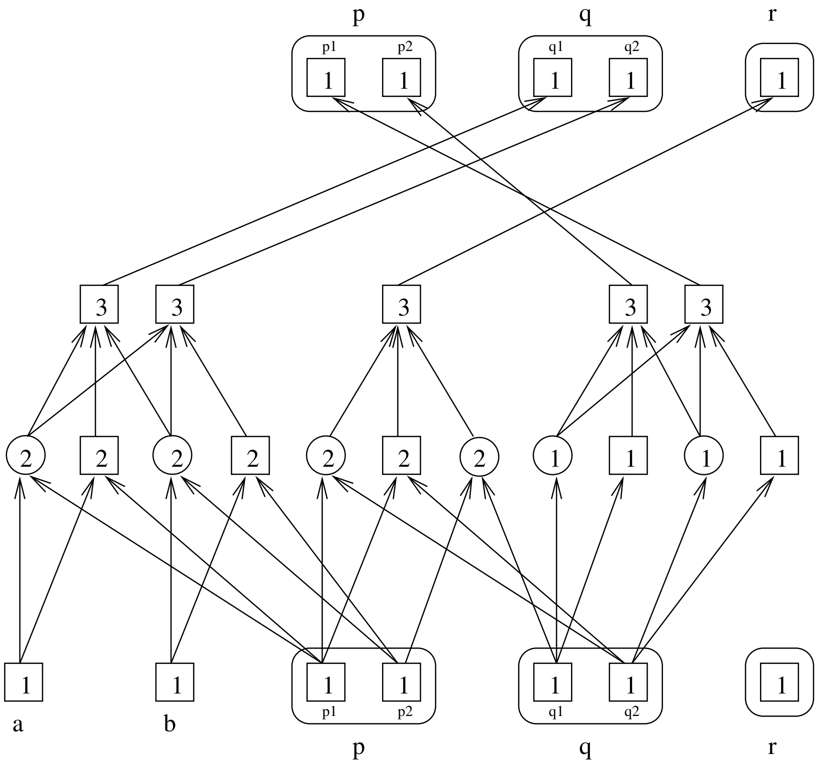


Abbildung 6.2: Das 4-schichtige vorwärtsgerichtete Netz für das Regelwissen aus Beispiel 6.2.2.

Als Nachteil ist zu nennen, daß die strenge Schichtenstruktur verlorengeht. Es entstehen Verbindungen zwischen nicht benachbarten Schichten, die sogenannten SHORTCUTS. Unter der Betrachtung, daß das Propagieren von Aktivierungen über eine Schicht genau $\Delta\omega$ Zeit in Anspruch nimmt, muß für einen SHORTCUT gelten, daß die Zeit zum Propagieren der Aktivierungen direkt proportional zu der Anzahl übersprungener Schichten ist. Eine Verbindung von der Eingabeschicht zur 2. verdeckten Schicht benötigt demzufolge $2 \times \Delta\omega$ Zeiteinheiten zum Propagieren der Phasen. Wichtig ist, daß die schichtweise Berechnung des Modells erhalten bleibt.
Abbildung 6.3 zeigt eine Vereinfachung zu dem Netz aus Abbildung 6.2. Die Äquivalenz der Modelle ist offensichtlich und kann mit der Funktion der Netzknoten und der Kodierung bewiesen werden. Weitere Minimierungen sind auch in der 2. verdeckten Schicht des Netzes möglich, aus Gründen der Übersichtlichkeit soll diese Darstellung jedoch beibehalten werden.

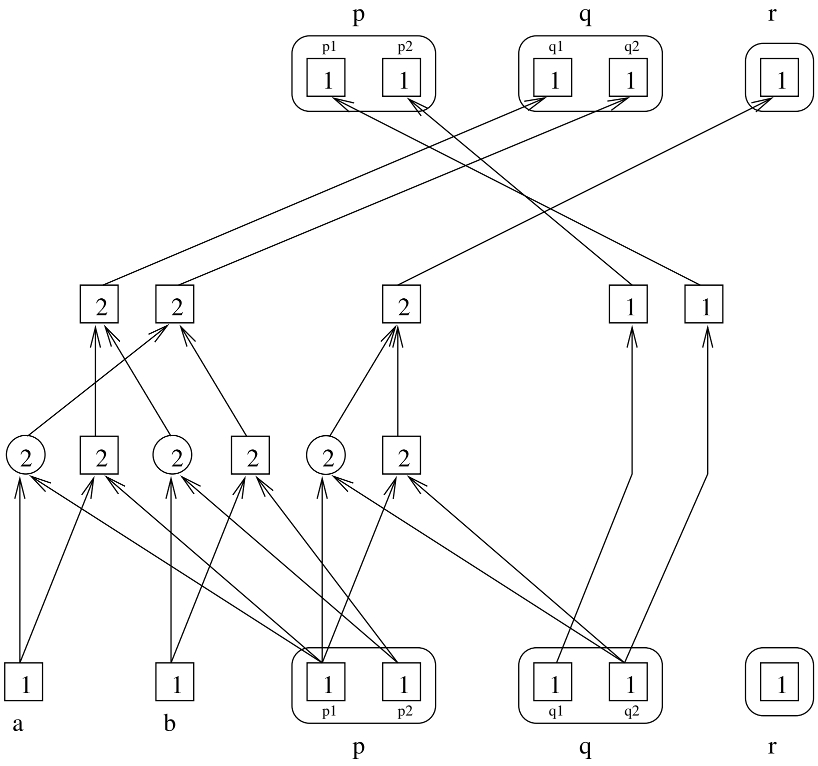


Abbildung 6.3: Ein vereinfachtes neuronales Netz für das RNN-Modell zu Beispiel 6.2.2.

6.3 Aufwandsabschätzung

Platzkomplexität

- Die Anzahl der Einheiten in der Ein- und Ausgabeschicht ist jeweils gleich der Anzahl der Konstanten und der Stelligkeit aller Prädikate im Programm P
- Für jede Programmklausel werden Einheiten in den verdeckten Schichten erstellt:
  - Die Anzahl der SWEZ ist gleich der Anzahl an Variablen und Konstanten im Klauselkörper
  - Die Anzahl der SWEP-Knoten ist direkt abhängig von der Stelligkeit des Prädikates im Kopf der Klausel

Sei C die Zahl der Konstanten und D die Zahl der Prädikate im Programm P. Die Anzahl der Klauseln in P wird bezeichnet mit K. Mit der durchschnittlichen Anzahl Variablen und Konstanten $\bar{v}$ im Klauselkörper und mit der durchschnittlichen Stelligkeit der Prädikatssymbole $\bar{s}$ ist die Zahl der Netzeinheiten im RNN-Modell beschränkt durch

$O(C+\bar{s}\times D+(\bar{v}+\bar{s})\times K)$. Vergleicht man dieses Ergebniss mit der Diskussion in Abschnitt 4.5, dann fällt auf, daß SHRUTI in der Grundversion dieselbe Platzkomplexität wie das RNN-Modell hat.
Für den Spezialfall einer Wissensbasis aus einstelligen Prädikatssymbolen ($\bar{s}=1$), für den der BUR-Kalkül korrekt ist, ist die Anzahl der Netzknoten durch $O(C+D+\bar{v}\times K)$ beschränkt.

- Die Anzahl der rekurrenten Verbindungen von der Ausgabeschicht zur Eingabeschicht ist gleich der Anzahl von Konstanten und der Gesamtstelligkeit aller Prädikate in P
- Für jede der Klauseln ist eine Anzahl von Verbindungen in den verdeckten Schichten erforderlich, die eine Komplexität von $O((\bar{n}+\bar{v})\times\bar{s})$ hat. $\bar{n}$ steht dabei für die durchschnittliche Anzahl der Prädikate im Klauselkörper.

Im allgemeinen Fall gilt für die Verbindungen im Netz eine Komplexität von $O(C+\bar{s}\times D+(\bar{n}+\bar{v})\times\bar{s}\times K)$. Bei Verwendung einstelliger Prädikatssymbole beschränkt sich dieser Aufwand auf $O(C+D+(\bar{n}+\bar{v})\times K)$ Verbindungen.

Aus der Abschätzung des Platzaufwandes ist bezüglich der Netzeinheiten und der Verbindungen zwischen ihnen festzustellen, daß das neue RNN-Modell lineare Komplexität bezüglich der Größe der Wissensbasis P hat. Das entspricht genau der Zielsetzung, die im vorraus an das Modell gestellt wurde.

Zeitkomplexität

Für die schichtweise Berechnung des Modelles sind 4 Schritte notwendig, die jeweils für die Propagierung der Aktivierungen über eine Schicht stehen. Unter Verwendung der $\max(O(C+\bar{s}\times D), O((\bar{s}+\bar{v})\times K))$ Prozessoren kann die Berechnung parallel durchgeführt werden. Dann ist die Zeitkomplexität parallel linear. Weil im allgemeinen Fall die Beziehung $I_d \supseteq T_P(\emptyset)\uparrow d$ gilt, kann ein stabiler Zustand im Netz schon vor dem Ende der längsten möglichen Inferenzkette erreicht werden. Im ungünstigsten Fall liegt der Fixpunkt bei 2^{B_P}. Die Zeit zum Erreichen des stabilen Zustandes ist trotzdem durch $O(4\times l\times\Delta\omega)$ begrenzt, wobei l die längste Inferenzkette der Wissensbasis P darstellt. Der Zeitaufwand ist damit linear abhängig von der Inferenzkette.

6.4 Beweisführung

Entsprechend den Ausführungen zur Beweisführung im BUR-Kalkül in Abschnitt 5.3.3 soll das am zugehörigen RNN-Modell demonstriert werden. Dazu beziehen wir uns erneut auf Beispiel 6.2.2 und das zugehörige vereinfachte Netz in Abbildung 6.3 für die Kodierung von P_R.
Zum Beginn des Inferenzvorganges werden wie beschrieben die Konstanten a und b zyklisch aktiviert und das Faktenwissen P_F unter Nutzung des Synchronitätsprinzips an der Eingabeschicht geclampt. Den Inferenzvorgang bis zum Fixpunkt zeigt die Abbildung

6.4. Die Eingabe für a, b und die Rollen p_1 und p_2 sind gesondert dargestellt. Die anderen Zeilen im Diagramm repräsentieren die Einheiten der Eingabeschicht des Netzes, also die Vereinigung von angelegtem Faktenwissen und den Ausgaben des vorwärtsgerichteten Netzes. Die Eingabeschicht steht damit für die abgeleiteten Formeln nach $\vdash_{BUR}$. Die Aktivierungen der Einheiten in den anderen Schichten werden nicht dargestellt, so daß zwischen den gezeigten Diagrammzyklen tatsächlich noch 3 weitere Berechnungszyklen liegen.

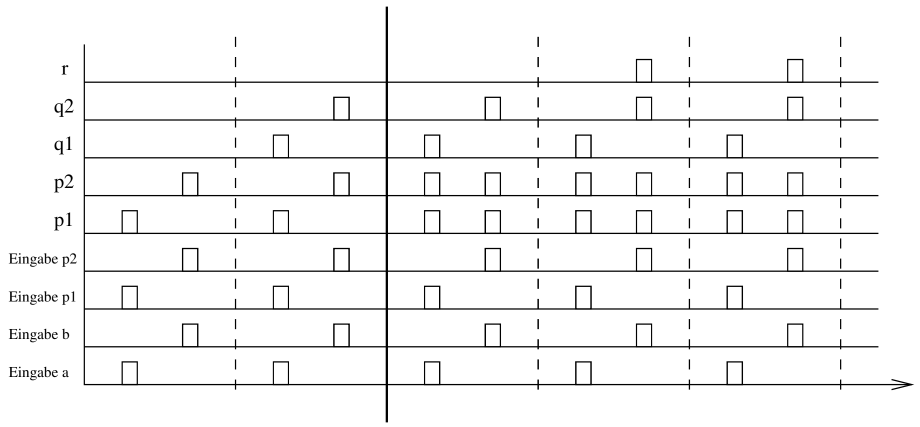


Abbildung 6.4: Der Inferenzvorgang im RNN-Modell für das Programm P aus Beispiel 6.2.2.

Der Fixpunkt der Berechnung ist nach 5 Zyklen erreicht. Er repräsentiert die Menge $I_5 = \{p(a,a), p(a,b), p(b,a), p(b,b), q(a,b), r(b)\}$. Man sieht jedoch, daß bis zum Erreichen des Fixpunktes CROSS-TALK aufgetreten sein könnte und deshalb nur $I_5 \supseteq T_P(\emptyset) \uparrow \infty$ gilt. Unter Nutzung des Korrektheitstheorems 5.3.5 kann festgestellt werden, daß dessen Gültigkeitsbedingung bis zum Inferenzschritt $d = 2$ erfüllt ist, was durch die durchgezogene Trennlinie dargestellt wird. Bis zu diesem Punkt gilt für jedes Prädikat, daß die Zahl der Bindungen nicht in mehr als einer Stelle größer als eins ist. Der Zeitpunkt repräsentiert die Menge $I_2 = \{p(a,b), q(a,b)\}$ und alle Elemente von I_2 sind logische Konsequenzen aus P. Bezogen auf den Fixpunkt verbleibt die Restmenge $\{p(a,a), p(b,a), p(b,b), r(b)\}$, über die nichts weiter ausgesagt werden kann. Es wird deshalb ein zweiter Inferenzprozeß gestartet, bei dem die logische Konsequenz $q(a,b)$ als Faktenwissen an der Eingabeschicht aktiviert wird. Abbildung 6.5 zeigt den Aktivierungsverlauf des RNN-Modells.

Der Fixpunkt dieser zweiten Berechnung ist nach 4 Inferenzschritten erreicht und genügt erfreulicherweise der Bedingung des Korrektheitstheorems. Alle im Fixpunkt repräsentierten Formeln sind damit logische Konsequenzen aus P_R und $q(a,b)$ und damit logische Konsequenzen von P. Dies ist ein Beweis für die abgeleitete Formelmenge $\{p(b,a), q(a,b), r(b)\}$. Aus den beiden dargestellten Berechnungen des RNN-Modells er-

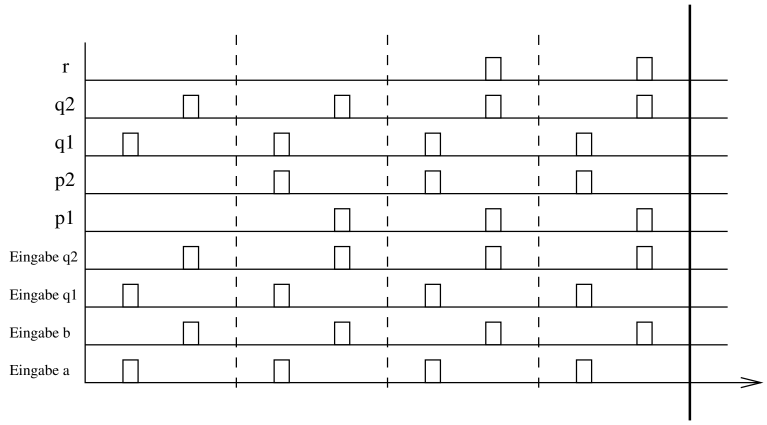


Abbildung 6.5: Der Inferenzvorgang bei Aktivierung der logischen Konsequenz $q(a,b)$ aus P.

gibt sich nun das Wissen, daß $P \models \{p(a,b), p(b,a), q(a,b), r(b)\}$ und es verbleibt eine Restmenge, über die wir nur wissen $P \vdash_{BUR} \{p(a,a), p(b,b)\}$. Diese Grundinstanzen von p können nicht durch weitere Berechnungen im RNN-Modell bzw. BUR-Kalkül bewiesen, aber auch nicht widerlegt werden. Mit zwei rückwärtschließenden Inferenzen (SHRUTI, Resolution, Konnektionsmethode, etc.) über P und den zwei im BUR-Kalkül abgeleiteten Formeln findet man mit parallel linearem Zeitaufwand einen Beweis, daß beide Formeln nicht logische Konsequenzen von P sind. Wegen der Vollständigkeit des BUR-Kalküls gilt nun $T_P(\emptyset) \uparrow \infty = \{p(a,b), p(b,a), q(a,b), r(b)\}$. Für die Ermittlung aller logischen Konsequenzen wurden somit zwei vorwärtsschließende und zwei rückwärtsschließende Inferenzen mit jeweils parallel linearem Aufwand benötigt.

6.5 Gegenüberstellung mit SHRUTI

Nun könnte man anführen, daß die Probleme im BUR-Kalkül bzw. RNN-Modell zu einem Großteil auch mit einem einzigen Inferenzvorgang in einem vorwärtsschließenden SHRUTI realisiert werden könnten. Das ist nicht ganz falsch. Tatsächlich existieren Probleme, die wegen seiner Korrektheit von SHRUTI gelöst werden können, aber im RNN-Modell mehrerer Inferenzprozeße oder zusätzlicher Rückwärtsschlüsse bedürfen. In diesem Zusamenhang sei aber erneut auf die Platzkomplexität[5], auf die Vielzahl von Netzeinheiten und auf die Vielzahl von Annahmen in SHRUTI verwiesen. Die Platzkomplexität im RNN-Modell ist stets linear und es gibt nur zwei Typen von Netzeinheiten. Außerdem können Regeln kodiert werden, bei denen Konstanten im Klauselkopf und im Klauselkörper auftreten, und allquantifizierte Variablen können auch nur im Klauselkopf

[5]siehe Abschnitt 4.5

auftreten. Zum Vergleich der beiden Modelle wollen wir nun noch einmal auf den Satz

„Klaus und sein Sohn John spielten Schach. Da kam John's Schwester Marie herein und fragte ihren Vater: ‚...'"

zurückgreifen, der in Beispiel 4.4.1 als Programm beschrieben wurde. Die Abbildungen 4.11 und 4.13 zeigen die Kodierung der Regeln und Fakten in SHRUTI unter Nutzung der Erweiterungen und verdeutlichen den damit verbundenen enormen Aufwand.
Im BUR-Kalkül wird der Satz interpretiert mit dem Regelwissen P_R:

$$\begin{aligned} Mann(Y) \wedge Sohn(X,Y) &\rightarrow Vater(Y,X) \\ Vater(X,Y) \wedge Schwester(Z,Y) &\rightarrow Vater(X,Z) \end{aligned}$$

und dem Faktenwissen P_F:

$$\begin{aligned} &\rightarrow Mann(Klaus) \\ &\rightarrow Sohn(John, Klaus) \\ &\rightarrow Schwester(Marie, John) \end{aligned}$$

Das implizite Wissen über die Sorte *Mann* wird nun dargestellt mit einem Prädikatssymbol derselben Bezeichnung, so daß der Fakt *Mann*(*Klaus*) zum ursprünglichen Wissen hinzugefügt wird. Das RNN-Modell für das Regelwissen P_R wird in Abbildung 6.6 dargestellt.

Es ist unschwer zu erkennen, daß das RNN-Modell wesentlich einfacher ist als das vergleichbare SHRUTI-Modell. Die Erweiterungen für Typhierarchie und mehrfache Prädikatsinstanzen in SHRUTI verschlechtern außerdem den Zeit- und Platzbedarf. Der Aufbau des RNN-Modells ist so einfach, daß ein Anlernen im Modell mit neuem Wissen und den bekannten Lernalgorithmen sehr gut vorstellbar ist, während dies in dem SHRUTI-Modell unmöglich erscheint.

Betrachten wir nun den Inferenzvorgang nach Abbildung 6.7 bei zyklischer Aktivierung der Konstanten *Klaus*, *John*, *Marie* und der Fakten P_F. Der Fixpunkt der Berechnung liefert für das Prädikat Vater nur in der Rolle *Kind* mehrere Bindungen. Auch alle anderen Prädikate genügen der Bedingung des Korrektheitstheorems 5.3.5. Damit gilt, daß alle abgeleiteten Formeln korrekte Folgerungen bzw. logische Konsequenzen des Programmes $P = P_F \cup P_R$ sind. Insbesondere ist das der Beweis dafür, daß *Klaus Vater* von *John* und auch *Vater* von *Marie* ist.

Das Beispiel hat gezeigt, daß Probleme, die augenscheinlich zur Klasse des reflexiven Schließens gehören, durchaus im RNN-Modell gelöst werden können. Dabei ist der Aufwand in einem vergleichbaren SHRUTI-System für dieses Beispiel sogar um ein Vielfaches höher.

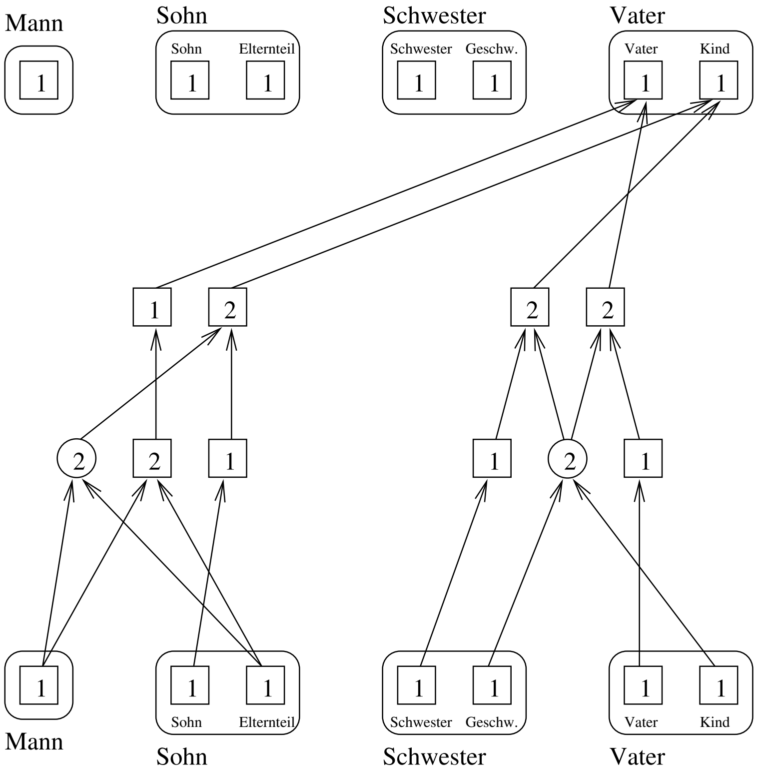


Abbildung 6.6: Das RNN-Modell für das Regelwissen der „Vater-Geschichte“.

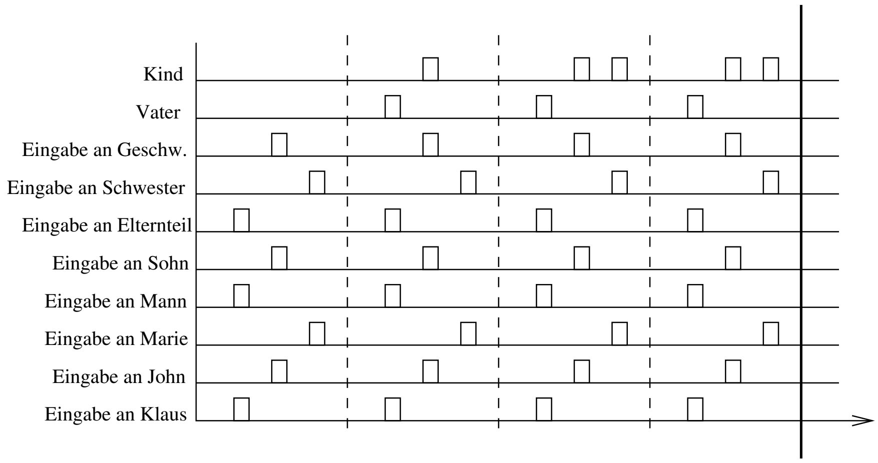


Abbildung 6.7: Der Inferenzprozeß für die „Vater-Geschichte“.

7. Zusammenfassung

In der vorliegenden Arbeit werden Möglichkeiten dargelegt, das von Hölldobler und Kalinke ([HK94]) vorgestellte RNN-Modell zur Berechnung von T_P für aussagenlogische Programme auf Prädikatenlogik zu erweitern. Dabei liegt der Fokus auf einem Modell, welches mit Konzepten aus dem konnektionistischen System SHRUTI arbeitet.

Die Darlegungen zum Rückwärtsschließen in SHRUTI durch Shastri und Ajjanagadde ([SA93]) ließen vermuten, daß das Vorwärtsschließen auch für eine große Klasse von Problemen mit linearem Platzaufwand und parallel linearem Zeitaufwand realisiert werden kann. Die detaillierte Erarbeitung der Kodierung und der Einschränkungen beim Vorwärtsschließen führten jedoch zu dem Ergebnis, daß im allgemeinen Fall die Platzkomplexität exponentiell und der Hauptgrund dafür das Kopieren von Regeln ist.

Stromaier liefert in [Str97] die Beweise, daß Rückwärtsschließen in SHRUTI der Anwendung von Reduktionen entspricht. Für das Vorwärtsschließen konnte dies in der Arbeit ebenfalls gezeigt werden. Als wichtigster Unterschied wurde festgestellt, daß Vorwärtsschließen das Generieren von Modellen ist, dem eine bestimmte Reihenfolge der Anwendung von Reduktionen zugrunde liegt.

Die Arbeit definiert den vorwärtsschließenden BUR-Kalkül, der ebenfalls durch eine festgelegte Priorität von Reduktionen Modelle generiert. Er basiert auf einer Hornklausellogik. Dabei erfolgt aber kein Kopieren von Regelwissen und die benötigte Platz- und Zeitkomplexität ist tatsächlich linear. Weil eine Reduktion in BUR nicht in Ihrer gültigkeitserhaltenden Orignialform verwendet wird, ist der Kalkül im allgemeinen Fall vollständig, aber nicht notwendigerweise korrekt. Es wird bewiesen, daß die Menge der in einem Inferenzschritt abgeleiteten Formeln eine Übermenge der Funktionswerte des Operators T_P bildet.

Es konnte außerdem ein Korrektheitstheorem bewiesen werden, welches aus den im BUR-Kalkül abgeleiteten Formeln entscheidet, ob diese garantiert korrekte Folgerungen aus der Wissensbasis sind. Mit der Kombination BUR-Kalkül, Korrektheitstheorem und Rückwärtsschließen kann so die Menge aller logischen Konsequenzen bestimmt werden.

Das neue RNN-Modell ist eine massiv parallele Implementierung des BUR-Kalküls. Es nutzt dynamische Bindungen mit zeitlicher Synchronität. Gegenüber SHRUTI hat es den Vorteil, daß lediglich zwei Typen von Netzeinheiten benötigt werden und die Platzkomplexität stets linear zur Größe der Wissensbasis und die Zeitkomplexität stets parallel linear zur Länge der längsten Inferenzkette ist. Das neue RNN-Modell kodiert nur Regelwissen. Ein einmal erstelltes Netz kann deshalb durch Aktivierung unterschiedlicher Fakten verschiedene Inferenzen realisieren.

Die weitere Arbeit auf diesem Gebiet sollte zunächst dahingehen, das neue RNN-Modell zu implementieren und auf typische Probleme der realen Welt anzuwenden. Dabei sollte vor allem festgestellt werden, ob das Korrektheitstheorem für eine große Klasse von Problemen ausreichend ist, um mit wenigen Inferenzen im RNN-Modell bzw. BUR-Kalkül alle logischen Konsequenzen zu ermitteln. Besonders interessant ist dabei der praktische Vergleich mit anderen Systemen, wie zum Beispiel SATCHMO ([MB88]). In Abhängigkeit von diesen Ergebnissen könnte dann an die Erweiterung des Modelles gedacht werden. Dort stellen sich Fragen bezüglich des Schließens über negiertem Wissen, der Wichtung von Verbindungen im Netz oder der Implementierung weiterer linearer Reduktionen. Interessant könnte auch ein Vergleich mit Methoden des approximativen Schließens sein, oder die Frage, wie mit den bekannten Lernalgorithmen das erstellte Netz trainiert werden kann.

A. Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet, sowie Zitate kenntlich gemacht habe.

Dresden, den 30.September 1998

Jörg Wunderlich

Literaturverzeichnis

[Ajj94] AJJANAGADDE, Venkat: Unclear Distinctions lead to Unnecessary Shortcomings: Examining the rule vs fact, role vs filler, and type vs predicate distinctions from a connectionist representation and reasoning perspective. **In:** *Proceedings of the AAAI National Conference on Artificial Intelligence*, 1994, S. 846–851

[Apt97] APT, Krzysztof R.: *From Logic Programming to Prolog*. Prentice Hall, 1997 (International Series in Computer Science)

[AS89] AJJANAGADDE, Venkat ; SHASTRI, Lokendra: Efficient Inference with Multi-Place Predicates and Variables in a Connectionist System. **In:** ERLBAUM, Lawrence (Hrsg.): *Proceedings of the Conference of the Cognitive Science Society*, 1989, S. 396–403

[BH93] BEHRINGER, Antje ; HÖLLDOBLER, Steffen: On the Adequateness of the Connection Method. **In:** *Proceedings of the AAAI National Conference on Artificial Intelligence*, 1993

[BHS93] BIBEL, Wolfgang ; HÖLLDOBLER, Steffen ; SCHAUB, Torsten: *Wissensrepräsentation und Inferenz: eine grundlegende Einführung*. Wiesbaden : Vieweg, 1993

[Bib87a] BIBEL, Wolfgang: Advanced Topics in Automated Deduction. **In:** NOSSUM, Rolf T. (Hrsg.): *Advanced Topics in Artificial Intelligence* Bd. 345. Springer-Verlag, 1987, S. 41–59

[Bib87b] BIBEL, Wolfgang: *Automated Theorem Proving*. 2. Wiesbaden : Vieweg, 1987

[Bib91] BIBEL, Wolfgang: Perspectives on Automated Deduction. **In:** BOYER, Robert S. (Hrsg.): *Automated Reasoning*. Kluwer Academic Publishers, 1991, Kapitel 4, S. 77–104

[Bib92] BIBEL, Wolfgang: *Deduktion*. Oldenbourg Verlag, 1992 (Handbuch der Informatik 6.2)

[Coo93] COOPER, Paul R.: Could static binding suffice ? **In:** *Behavioral and Brain Sciences* 16 (1993), S. 453–454. – (as a commentary to [SA93])

[DKRS93] DIETZ, Paul ; KRIZANC, Danny ; RAJASEKARAN, Sanguthevar ; SHASTRI, Lokendra: A Lower Bound Result for The Common Element Problem and Its Implification for Reflexive Reasoning / University of Pennsylvania, Computer and Information Science Department. 1993 (MS-CIS-93-73). – Forschungsbericht

[FHMV95] FAGIN, Ronald ; HALPERN, Joseph Y. ; MOSES, Yoram ; VARDI, Moshe Y.: *Reasoning about Knowledge*. MIT Press, 1995

[Fit94] FITTING, Melvin: Metric methods, three examples and a theorem. **In:** *Journal of Logic Programming* (1994), Nr. 21, S. 113–127

[Fun90] FUNAHASHI, K.-I.: On the approximate realization of continuous mappings by neural networks. **In:** *Neural Networks* (1990), Nr. 2, S. 183–192

[Gö95] GÖRZ, Günther (Hrsg.): *Einführung in die Künstliche Intelligenz*. 2. Addison-Wesley, 1995

[GR88] GIBBONS, Alan ; RYTTER, Wojciech: *Efficient Parallel Algorithms*. Cambridge University Press, 1988

[HK94] HÖLLDOBLER, S. ; KALINKE, Y.: Towards a New Massively Parallel Computational Model for Logic Programming. **In:** *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing*. Amsterdam, 1994, S. 68–77

[HKP91] HERTZ, John ; KROGH, Anders ; PALMER, Richard G.: *Introduction to the theory of neural computation*. Addison-Wesley, 1991 (Santa Fe Institute studies in the science of complexity. Computation and neural systems series.)

[HKS98] HÖLLDOBLER, Steffen ; KALINKE, Yvonne ; STÖRR, Hans-Peter: Approximating the Semantics of Logic Programs by Recurrent Neural Networks / Technische Universität Dresden. 1998. – Forschungsbericht

[Kal94] KALINKE, Yvonne: *Ein massiv paralleles Berechnungsmodell für normale logische Programme*, Technische Universität Dresden, Diplomarbeit, 1994

[Kal97] KALINKE, Yvonne: Using Connectionist Term Representations for First-Order Deduction - A Critical View. **In:** *CADE-14, Workshop on Connectionist Systems for Knowledge Representation and Deduction*. Townsville, Australia, 1997

[KR90] KARP, R. M. ; RAMACHANDRAN, V.: Parallel Algorithms For Shared-Memory Machines. **In:** LEEUWEN, Jan van (Hrsg.): *Handbook of Theoretical Computer Science* Bd. A - Algorithms and Complexity. Elsevier Science Publishers B.V., 1990, Kapitel 17, S. 869–941

[KS95] Kalinke, Yvonne ; Störr, Hans-Peter: Approximation der Funktion Tp normaler logischer Programme P durch 3-schichtige vorwärtsgerichtete Netze / Technische Universität Dresden. 1995. – Forschungsbericht

[KS96] Kalinke, Yvonne ; Störr, Hans-Peter: Rekurrente Neuronale Netze zur Approximation der Semantik akzeptabler logischer Programme. **In:** *Fortschritte in der Künstlichen Intelligenz*. Dresden University Press, 1996, S. 27

[LD89] Lange, Trent E. ; Dyer, Michael G.: High-Level Inferencing in a Connectionist Network / Artificial Intelligence Laboratory, Computer Science Department, University of California, Los Angeles. 1989. – Forschungsbericht

[Leh95] Lehmann, Helko: Vorwärtsschließen mit SHRUTI / TU Dresden. 1995. – Forschungsbericht

[Llo87] Lloyd, John W.: *Foundations of Logic Programming*. Second, Extended. Springer-Verlag, 1987

[Mas96] Massacci, Fabio: From Sound and Complete to Approximate Deduction. **In:** Gent, Ian (Hrsg.): *AISB-96 Workshop on Automated Reasoning ‚Bridging the Gap Between Theory and Practice'*, 1996. – Position paper for the panel ‚Implementing Temporal Reasoning'

[Mas97] Massacci, Fabio: A Proof Theory for Tractable Approximations of Propositional Reasoning. **In:** Lenzerini, Maurizio (Hrsg.): *Proceedings of the 5th Conference of the Italian Association for Artificial Intelligence (AI*IA-97)* Bd. 1321. Bd. 1321, Springer Verlag, 1997, S. 219–230

[MB88] Manthey, Rainer ; Bry, Francois: SATCHMO: a theorem prover implemented in PROLOG. **In:** *Proceedings on the 9th International Conference on Automated Deduction*, Springer, 1988, S. 415–434

[MS92a] Mani, D. R. ; Shastri, Lokendra: A Connectionist Solution to the Multiple Instantiation Problem using Temporal Synchrony. **In:** *Proceedings of the Fourteenth Conference of the Cognitive Science Society*, 1992

[MS92b] Mani, D. R. ; Shastri, Lokendra: Multiple Instantiation of Predicates in a Connectionist Rule-Based Reasoner / University of Pennsylvania. 1992 (MS-CIS-92-05). – Forschungsbericht

[MW98] Meyer-Wegener, Klaus: *Datenstrukturen und Datenbanken*. Vorlesungsskript, Technische Universität Dresden, 1998

[Rei97] Reichel, Horst: *Logik für Informatiker*. Vorlesungsskript, Technische Universität Dresden, 1997

[RN95] Russell, Stuart J. ; Norvig, Peter: *Artificial Intelligence: a modern approach*. Prentice-Hall, 1995

[SA93] SHASTRI, Lokendra ; AJJANAGADDE, Venkat: From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. **In:** *Behavioral and Brain Sciences* 16 (1993), S. 417–494

[SB95] SUN, Ron (Hrsg.) ; BOOKMAN, Lawrence A. (Hrsg.): *Computational Architectures Integrating Neural and Symbolic Processes*. Kluwer Academic Publishers, 1995

[SC95] SCHAERF, Marco ; CADOLI, Marco: Tractable Reasoning via Approximation. **In:** *AIJ* 74 (1995), S. 1–62

[Sei96] SEILER, Torsten: *Rekursiver Autoassoziativer Speicher und Holographisch Reduzierte Repräsentation*, Technische Universität Dresden, Diplomarbeit, 1996

[SG95] SHASTRI, Lokendra ; GRANNES, Dean J.: Dealing with negated knowledge and inconsistency in a neurally motivated model of memory and reflexive reasoning. (1995)

[Sha93] SHASTRI, Lokendra: A Computational Model of Tractable Reasoning - taking inspiration from cognition. **In:** *Proceedings of the IJCAI International Joint Conference on Artificial Intelligence*, 1993, S. 202–207

[SM97] SHASTRI, Lokendra ; MANI, D. R.: Massively parallel knowledge representation and reasoning: Taking a cue from the brain. **In:** GELLER, J. (Hrsg.): *Parallel Processing for Artificial Intelligence* Bd. 3. North-Holland, 1997

[Str97] STROHMAIER, Antje: *Dissertationen zur Künstlichen Intelligenz*. Bd. 169 : Logisches Schließen mit massiv parallelen Methoden. Infix, 1997

[Sun94] SUN, Ron: *Integrating Rules and Connectionism for Robust Commonsense Reasoning*. John Wiley and Sons, 1994

[Wun97] WUNDERLICH, Jörg: *Neuronale Netze zum induktiven Lernen über Aussagenlogischen Programmen*, Technische Universität Dresden, Großer Beleg, 1997