Projektarbeit im Rahmen des Fortgeschrittenenpraktikums Thema: Anforderungsanalyse für ein generisches Visualisierungssystem für verschiedene Deduktionsverfahren

Christian Braun

13. April 1999

- Version 9 -

Inhaltsverzeichnis

1 Einleitung

1.1 Ziel dieser Arbeit

In dieser Projektarbeit sollen die Anforderungsbeschreibung und ein detailliertes Konzept für das Design der Anwendung *SNARKS II* vorgestellt werden. Wie der Name des Programms bereits andeutet, handelt es sich um den Nachfolger eines bereits bestehenden Programms.

Der Vorgänger *SNARKS* ist ein graphisches Tool zur Analyse von Ableitungen, die mit Hilfe der *PUHR-Tableau-Methode* entwickelt wurden. Es gibt eine Reihe von Möglichkeiten die so erzeugten Bäume zu visualisieren.

Jedoch gibt es auch einige Punkte, die verbessert werden könnten:

- Erweiterung auf weitere Verfahren
  Bisher baut das System auf *PUHR-Tableaux* auf. Durch eine Verallgemeinerung auf weitere Verfahren könnte das System noch flexibler eingesetzt werden. Denkbar wären hier zum Beispiel andere *Tableaux*- oder *SLD*-Verfahren.

- Verbesserung der Darstellungsmöglichkeiten
  Es gibt einige Punkte zur Verbesserung der Visualisierung der Ableitungen.

Im nächsten Abschnitt wird eine vergleichende Beschreibung der erforderlichen Funktionalität erstellt. Anhand von verschiedenen Verfahren wird dann ein Rahmen für die Modellierung illustriert.

1.2 Mathematische Notationen

Vorab werden ein paar mathematische Vereinbarungen eingeführt: Die natürlichen Zahlen $\mathbb{N}$ bestehen aus den Zahlen $0, 1, 2, \ldots$.

Definition 1.1
Eine *Liste* ist wie folgt induktiv definiert: Liste

- Das Symbol ε, manchmal auch $<>$ notiert, ist die sogenannte leere Liste.

- Wenn L eine Liste und a ein Objekt ist, dann ist $< L, a >$ ebenfalls eine Liste.

Statt $< \ldots << \varepsilon, a_1 >, a_2 > \ldots, a_n >$ schreibt man einfach $< a_1, a_2, \ldots, a_n >$. Weitere Notationen sind:

- Die *Konkatenation* zweier Listen L_1 und L_2 wird $L_1 \circ L_2$ geschrieben.

- Das n-te ($n \in \mathbb{N} + 1$) Element einer Liste wird mit $L[n]$ bezeichnet.

- Man schreibt $a \in L$, um auszudrücken, dass a in der Liste L enthalten ist. Analog ist $a \notin L$ definiert.

- Die Anzahl der Elemente einer Liste L wird mit $|L|$ bezeichnet.

- Statt alle Elemente einer Liste aufzuzählen, ist es manchmal sinnvoller, deren Elemente durch eine Bedingung zu spezifizieren. Notiert wird dies $< E \mid B >$, wobei E ein Term oder eine Formel ist. Alle freien Variablen in E werden durch die Bedingung B gebunden. Zu beachten ist, dass die Reihenfolge der Elemente bei einer Liste relevant ist.
  Also beispielsweise ist $< 2^i \mid 0 \leq i \leq 8 >$ genauer definiert durch $< 2^i \mid i = 0, \ldots, 8 > = < 1, 2, 4, 8, 16, 32, 64, 128, 256 >$.

□

Definition 1.2
Für Substitutionen werden wir folgende Schreibweisen benutzen:

- Sei V die Menge aller Variablen und T die Menge aller Terme. Eine *Substitution* $\sigma : V \to T$ ist eine Abbildung, die Variablen in Terme abbildet und postfix geschrieben wird. Weiter definiert man $edom(\sigma) := \{X \mid X$ Variable und $X\sigma \neq X\}$ (*edom* steht für essential domain). Es wird davon ausgegangen, dass $edom(\sigma)$ stets endlich ist. Für σ schreibt man auch $[X_1\sigma/X_1, \ldots, X_n\sigma/X_n]$, falls $edom(\sigma) = \{X_i \mid 1 \leq i \leq n\}$. Substitution

- Die *Identitätssubstitution* $\sigma_{id} = []$ ist die Substitution, deren $edom = \emptyset$ ist.

- Sei φ eine Formel der Prädikatenlogik erster Stufe gegeben, σ eine Substitution und $edom(\sigma) = \{X_i \mid 1 \leq i \leq n\}$. Dann ist $\varphi\sigma := \varphi[X_1\sigma/X_1, \ldots, X_n\sigma/X_n]$ diejenige Formel, die aus φ entsteht, wenn alle freien Vorkommen von X_i in φ durch $X_i\sigma$ ersetzt werden.

- Die Anwendung einer Liste von Substitutionen $\Sigma =< \sigma_1, \ldots \sigma_m >$ auf eine einzelne Formel erzeugt $\varphi < \sigma_1, \ldots, \sigma_m >:= < \varphi\sigma_1, \ldots, \varphi\sigma_m >$.

- Ist $F = \{\varphi_1, \ldots, \varphi_k\}$ eine ganze Formelmenge, so definiert man weiter: $F < \sigma_1, \ldots, \sigma_m >:= \{\varphi_\nu < \sigma_1, \ldots, \sigma_m > \mid \nu \in \{1, \ldots, k\}\}$.

□

2 Anforderungen

In diesem Abschnitt sollen die Ziele und Anforderungen der zu schreibenden Anwendung dargestellt werden.

Ziel des zu entwerfenden Systems ist eine Verallgemeinerung von *SNARKS*, so dass insbesondere weitere Beweisverfahren dargestellt werden können. Das System soll aber nicht auf Beweisverfahren eingeschränkt sein. Es gibt vielfältige Anwendungen, die mit einem solchen System visualisiert werden könnten. Spiele wie Othello sind genauso denkbar wie Ableitungen aus einer Grammatik.
Zunächst ist es sinnvoll, einen allgemeinen begrifflichen Rahmen zu schaffen, mit dem alle angestrebten Verfahren beschrieben werden können.
Wir unterscheiden zwei Ebenen: eine Deduktions- und eine Darstellungsebene.

- *Deduktionsebene*
  Die Deduktionsebene beschäftigt sich mit möglichen Bestandteilen von Ableitungen, möglichen Beziehungen zwischen diesen Bestandteilen und den vorkommenden Ableitungsstrukturen.

- *Darstellungsebene*
  Die Darstellungsebene beschäftigt sich mit den wünschenswerten Möglichkeiten der Visualisierung dieser Ableitungen.

Auch in einer Implementierung ist eine solche Zweiteilung sinnvoll. Die Deduktionsebene interagiert direkt mit einem Deduktionssystem, das entweder in dieser Ebene eingebettet ist oder durch eine eigene, abgetrennte Komponente realisiert wird.
Die Darstellungsebene visualisiert die Strukturen, die sie von der Deduktionsebene erhält und bietet dem Benutzer die Möglichkeit, auf die Ableitung Einfluss zu nehmen.

Graphisch kann dies folgendermaßen veranschaulicht werden. In der ersten Darstellung ist das Deduktionssystem direkt in der Deduktionsebene enthalten.

Darstellungsebene
Deduktionsebene Deduktionssystem

Die Struktur des bisherigen *SNARKS* entspricht dem, allerdings sind die Deduktionsebene des Visualisierungssystems und das Deduktionssystem konzeptionell nicht klar unterschieden.
In der zweiten Darstellung ist das Deduktionssystem getrennt von der Deduktionsebene:

Darstellungsebene
Deduktionsebene
Deduktionssystem

Man muss zwei Arten von Benutzern unterscheiden: Konfiguratoren und Endanwender.

- Ein Konfigurator schließt ein Deduktionssystem an die Deduktionsebene bzw. die Darstellungsebene an. Er hat eine Reihe von Entscheidungen zu treffen, wie das zu visualisierende Verfahren in den begrifflichen Rahmen eingepasst wird und wie die Strukturen von Ableitungen den verfügbaren Darstellungsmitteln zugeordnet werden.

- Ein Endanwender benutzt das konfigurierte System zur Visualisierung und Analyse von Ableitungen in (möglicherweise) verschiedenen Verfahren.

Diese Projektarbeit richtet sich im besonderen Maße an einen Konfigurator. Zuerst wird ein allgemeiner begrifflicher Rahmen geschaffen und dann exemplarisch verschiedene Verfahren in diesem Rahmen dargestellt. Teilweise werden mehrere Varianten besprochen.

2.1 Die Deduktionsebene

Die darzustellenden Ableitungen werden mit Hilfe der folgenden allgemeinen Begriffen modelliert: Partikel, Partikelliste, initiale Partikelliste, Eigenschaften, Transformator und Transformatorenliste, Regel, Substitution und Substitutionsliste, Expansion, Teilexpansion, potentielle Teilexpansion und Strategie.

2.1.1 Die statischen Bestandteile der Deduktionsebene

Partikel

Ein *Partikel* ist ein elementarer Bestandteil von Ableitungen, der von einem Ableitungsschritt „erzeugt" und von anderen Ableitungsschritten verwendet werden kann. Ein Partikel ist normalerweise eine Formel einer Logik. Partikel haben i.A. eine innere Struktur, die dem Visualisierungssystem aber unbekannt ist. Partikel werden lediglich angezeigt.
Beispiele: Ein Partikel eines PUHR-Tableaux ist ein Grundatom oder eine Disjunktion von Grundatomen. Ein Partikel der vereinfachten SLD-Resolution ist eine atomare Anfrage. Ein Partikel eines Smullyan-Tableaux ist eine geschlossene Formel.

Partikelliste

Eine *Partikelliste* repräsentiert eine Gruppe von Partikeln, die von einem Ableitungsschritt expandiert werden kann. Die durch die Liste vorgegebene Reihenfolge der Partikel ist für manche Deduktionsverfahren relevant. Für manche Verfahren sind alle Partikellisten einelementig.

Beispiele: Eine Partikelliste eines PUHR-Tableaux steht für eine Menge von Grundatomen oder Disjunktionen von Grundatomen, wobei die Reihenfolge der Partikel unwesentlich ist. Eine Partikelliste der SLD-Resolution ist eine Anfrage. Die Reihenfolge spielt bei diesem Verfahren eine wichtige Rolle.

Initiale Partikelliste

Bei jedem Verfahren gibt es eine *initiale Partikelliste*. Je nach Verfahren ist sie konstant oder kann vom Benutzer beeinflusst werden.
Beispiele: Die initiale Partikelliste eines PUHR-Tableaux besteht immer aus dem Partikel $\top$. Bei der SLD-Resolution besteht sie aus der Anfrage, die der Benutzer gestellt hat.

Eigenschaften

Partikellisten können nach bestimmten *Eigenschaften* klassifiziert werden. So kann eine Partikelliste expandierbar sein oder auch nicht. Im Allgemeinen gibt es noch weitere Eigenschaften.
Beispiele: Geschlossen (steht für Widerspruch), offen, Modell, minimales Modell.

Transformator
Transformatorliste

Ein *Transformator* ist ein Bestandteil von Ableitungen, der in einem Ableitungsschritt verwendet, aber nicht „erzeugt" werden kann. Die *Transformatorliste* enthält alle Transformatoren, die global für die gesamte Ableitung zur Verfügung stehen. Es wird angenommen, dass es nur eine Transformatorliste zu einer Ableitung gibt. Manche Verfahren benötigen diesen Begriff nicht. Transformatoren werden vom Visualisierungssystem lediglich angezeigt, ihre interne Struktur wird nicht verarbeitet.
Beispiele: Eine Transformatorliste eines PUHR-Tableaux ist eine Menge von bereichsbeschränkten Klauseln in Implikationsform. Eine Transformatorliste der SLD-Resolution ist eine Liste von Programmklauseln. In Smullyan-Tableaux gibt es keine Transformatoren.

Regel

Eine *Regel* ist eine Schlussregel des jeweiligen Kalküls. Regeln werden lediglich benannt.
Beispiele: PUHR- und Splitting-Regel bei PUHR-Tableaux, SLD-Resolutionsregel bei der SLD-Resolution, β-Regel bei den Smullyan-Tableaux.

Substitution
Substitutionsliste

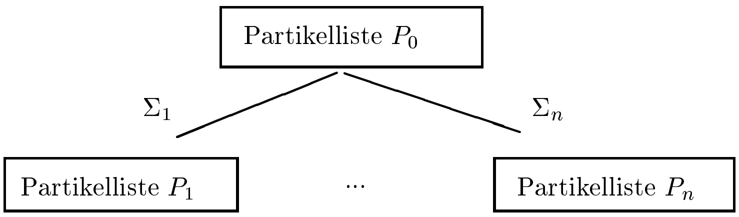
Eine *Substitution* ist eine Abbildung von Variablen auf Terme. Substitutionen werden vom Visualisierungssystem lediglich angezeigt, ihre interne Struktur wird nicht verarbeitet. Es gibt zwei Arten von *Substitutionslisten*:

1. Für jeden Ableitungsschritt eine lokale Substitutionsliste.

2. Eine globale Substitutionsliste, die den gesamten Baum betrifft.

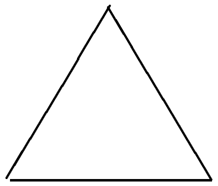
Substitutionen der ersten Art können sich auf ein oder mehrere Partikel und Transformatoren beziehen. Diese Information liefert das Deduktionssystem nicht explizit. Der Anwender kann dies aber aus dem Kontext erkennen. Die Interpretation einer Substitution ist somit vollständig dem Benutzer überlassen.

Folgende Abbildung illustriert Substitutionslisten Σ_i der ersten Art:



Die zweite Art bezieht sich immer auf den gesamten Baum. Der Baum steht dabei für mehrere Instanzen. Die Substitutionen geben an, wie man aus dem allgemeinen die speziellen Bäume erhält. Diese Substitutionsliste wird ebenfalls nur als Information ausgegeben. Der Anwender muss diese Angaben interpretieren. Ein Beispiel für derartige Substitutionen findet man bei DDB-trees, die im Abschnitt

2.1.9 skizziert werden.



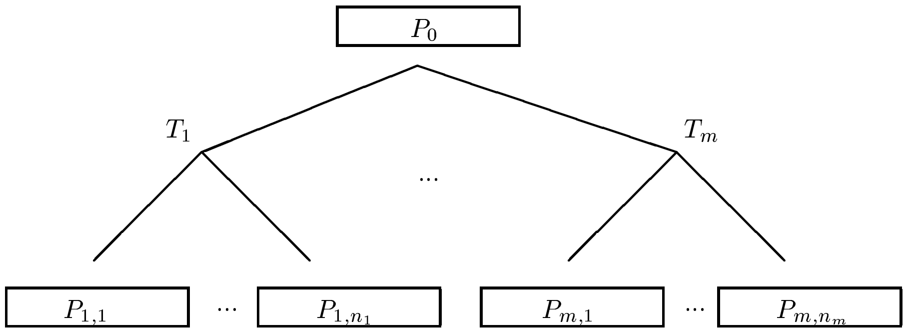
Allgemeiner Baum B

kann mittels $\Sigma =< \sigma_1, \ldots, \sigma_m >$ instantiiert werden:

Instanz $B\sigma_1$ Instanz $B\sigma_m$

Eine *Expansion* bildet eine Partikelliste in $k \in \mathbb{N}+1$ neue Partikellisten ab. Ausgehend von der initialen Partikelliste wird dadurch der Ableitungsbaum erzeugt. Expansion

Eine Expansion ist dabei eine Liste von *Teilexpansionen*, wie es die folgende Abbildung andeutet: P_0 wird in dieser Grafik in *einer* Expansion auf die Nachfolger $P_{1,1}$ bis P_{m,n_m} abgebildet. Die $T_1, \ldots, T_m$ stellen die Teilexpansionen dar. In diesem Fall ist $k = \Sigma_{i=1}^{m} n_i$. Teilexpansion



Die intendierte Bedeutung dieser Struktur ist, dass eine Teilexpansion eine Anwendung einer (Instanz einer) Kalkülregel modelliert. Dadurch ergibt sich in den meisten Fällen eine einzige Partikelliste als Nachfolger, aber für β-Regeln und ähnliche Regeln können es mehrere sein.

Die gesamte Expansion einer Partikelliste, also eine Liste von Teilexpansionen, modelliert dagegen alternative Anwendungen von verschiedenen (Instanzen von) Kalkülregeln. Dadurch kann der Suchraum des jeweiligen Verfahrens ganz oder teilweise dargestellt werden.

Bei jeder Teilexpansion sind folgende Informationen beteiligt:

1. die Vorgänger-Partikelliste
2. (a) Regel
   (b) ggf. der Transformator
   (c) Substitutionsliste
   (d) Verwendete Partikel

3. pro Nachfolger gibt es:
   (a) eine Bijektion „*korrespondiert*“ zwischen einer Teilmenge der Vorgänger-Partikelliste und einer Teilmenge der Nachfolger-Partikelliste.
   (b) eine Menge von „*abhängigen*“ Partikel aus der Nachfolger-Partikelliste. Jedes Paar (π_1, π_2), wobei π_1 verwendetes Partikel und π_2 abhängiges Partikel ist, steht in der Relation „*abhängig*“.

   Beispiele folgen in späteren Abschnitten.

Eine *potentielle Teilexpansion* enthält die Informationen unter 1. und 2., aber nicht die Informationen unter 3. Mit potentiellen Teilexpansionen kann modelliert werden, dass es weitere Alternativen für Anwendungen von Regeln gibt, die aber in der gegebenen Ableitung nicht repräsentiert sind.

potentielle Teilexpansion

Bemerkung
Nicht alle genannten Informationen sind immer interessant. Beispielsweise gibt es monotone Verfahren, bei denen jede Nachfolger-Partikelliste eine Obermenge ihrer Vorgänger-Partikelliste ist, beispielsweise bei PUHR-Tableaux. Einige Verfahren besitzen nur eine einzige Regel, ein Beispiel dafür ist die SLD-Resolution. □

2.1.2 Die dynamischen Bestandteile der Deduktionsebene

Bei einer gegebenen Partikelliste kann man im Allgemeinen mehrere Teilexpansionen anwenden. Die Auswahl der tatsächlichen Teilexpansion kann anhand einer *Strategie* erfolgen oder direkt durch den Benutzer gesteuert werden. Es gibt dabei folgende Möglichkeiten:

Strategie

- Auswahl der Regel
- Auswahl der Partikel
- Auswahl der Transformatoren

Auch hier gilt: Nicht jedes Verfahren bietet alle Varianten. Ist eine Strategie festgelegt worden, so wird die Auswahl automatisch getroffen. Soll jeder einzelne Schritt vom Benutzer selbst bestimmt werden oder möchte der Anwender die Alternativen zu einem Schritt sehen, so muss das System zu jeder Partikelliste eine Liste aller potentiellen Teilexpansionen liefern.

2.1.3 Modellierung von PUHR-Tableaux in diesem Rahmen

- *Partikel*:
  Grundatom oder Disjunktion von Grundatomen
- *Partikelliste*:
  Liste von solchen Partikeln. Jede Partikelliste definiert eine Interpretation, nämlich die Herbrand-Interpretation, die genau die in der Liste enthaltenen Grundatome erfüllt.
- *Initiale Partikelliste*:
  Die initiale Partikelliste besteht nur aus $\top$.
- *Eigenschaft*:
  Eine Partikelliste, die $\bot$ enthält, hat die Eigenschaft „geschlossen“. Eine nicht-expandierbare Partikelliste, die $\bot$ *nicht* enthält, hat die Eigenschaft „offen“.

- *Transformatoren*:
  Bereichsbeschränkte Implikationen der Form $A_1 \wedge \ldots \wedge A_n \Rightarrow B_1 \vee \ldots \vee B_k$.

- *Regeln*:
  PUHR- und Splitting-Regel.

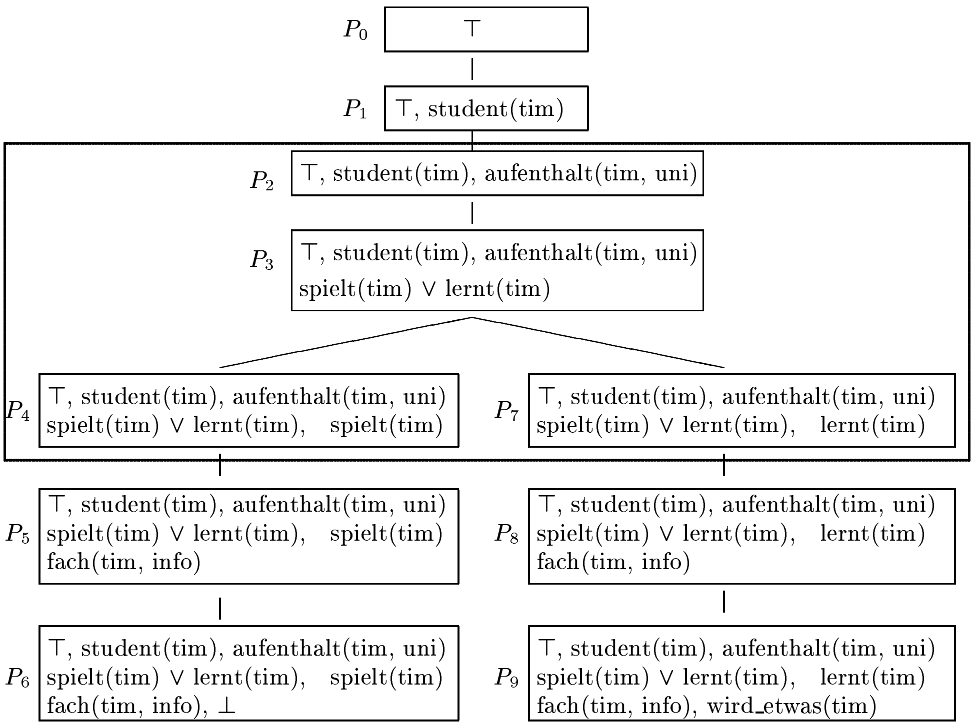
- *Substitutionen*:
  Es gibt nur die Substitutionen, die eine Teilexpansion betreffen. Jede Substitutionsliste ist einelementig.

- *Expansion*:
  Eine Teilexpansion modelliert eine Anwendung der PUHR-Regel oder eine Anwendung der Splitting-Regel. Im ersten Fall gibt es genau eine Nachfolger-Partikelliste, und diese ist eine Obermenge der Vorgänger-Partikelliste. Im zweiten Fall gibt es mehrere Nachfolger-Partikellisten, die alle Obermengen der Vorgänger-Partikelliste sind.

Beispiel 2.1
Sei die folgende Transformatorenliste T gegeben:

$$\begin{aligned}
lernt(Person) &\Rightarrow wird_etwas(Person). && (1)\\
student(Person) \wedge aufenthalt(Person, uni) &\Rightarrow spielt(Person) \vee lernt(Person). && (2)\\
assi(Person) \wedge aufenthalt(Person, uni) &\Rightarrow lehrt(Person) \vee forscht(Person). && (3)\\
fach(Person, informatik) \wedge spielt(Person) &\Rightarrow \bot. && (4)\\
\top &\Rightarrow student(tim). && (5)\\
\top &\Rightarrow fach(tim, info). && (6)\\
\top &\Rightarrow aufenthalt(tim, uni). && (7)
\end{aligned}$$

Daraus leitet sich beispielsweise folgendes PUHR-Tableau ab:



Die initiale Partikelliste P_0 besteht bei diesem Verfahren immer aus $\top$. Im ersten Schritt wird das Grundatom *student*(*tim*) mittels PUHR-Regel, Transformator 5

und der Identitätssubstitution hergeleitet. Alternativ hätten aber auch die Transformatoren 6 und 7 verwendet werden können. Auf Partikelliste P_1 wird nun die PUHR-Regel mit dem Transformator 7 angewendet.
Im folgenden Schritt könnte man sowohl Transformator 2, als auch Nummer 6 anwenden. Im Detail heißt das für den eingerahmten Teil der obigen Abbildung:

Partikelliste	P_2
Eigenschaften	ε
Teilexpansionen	$< P_2 \rightarrow P_3 >$
Potentielle Teilexpansionen	$< P_2 \rightarrow X >$

In diesem Fall wird die PUHR-Regel mit Transformator 2 und der Substitution $[tim/Person]$ angewendet. Als neues Partikel erhält man $spielt(tim) \vee lernt(tim)$.

Als Namen der Teilexpansionen werden hier mnemotechnische Bezeichner wie $P_2 \rightarrow P_3$ und $P_2 \rightarrow X$ gewählt, um die Lesbarkeit zu erleichtern. Die Bezeichner sind aber ansonsten beliebig.

Teilexpansion	$P_2 \rightarrow P_3$
Vorgänger	P_2
Regel	PUHR
Transformator	$T[2]$
Substitutionsliste	$< [tim/Person] >$
Verwendete Partikel	$< P_2[2], P_2[3] >$
Nachfolger	P_3
korrespondiert	$(P_2[i], P_3[i]),\ \ 1 \leq i \leq 3$
abhängig	$(P_2[2], P_3[4])$
	$(P_2[3], P_3[4])$

Die im obigen PUHR-Tableau nicht repräsentierte Alternative wird durch eine potentielle Teilexpansion modelliert:

Potentielle Teilexpansion	$P_2 \rightarrow X$
Vorgänger	P_2
Regel	PUHR
Transformator	$T[6]$
Substitutionsliste	$< [] >$
Verwendete Partikel	$< P_2[1] >$

Für die Nachfolger-Partikelliste P_3 hat man die Wahlmöglichkeit zwischen PUHR-Regel mit Transformator 6 oder Splitting Regel:

Partikelliste	P_3
Eigenschaften	ε
Teilexpansionen	$< P_3 \rightarrow P_4 P_7 >$
Potentielle Teilexpansionen	$< P_3 \rightarrow Y >$

In der obigen Ableitung wurde die Splitting-Regel gewählt und zwei Nachfolgerpartikellisten generiert.
Dies wird durch folgende Teilexpansion dargestellt:

Teilexpansion	$P_3 \rightarrow P_4 P_7$
Vorgänger	P_3
Regel Transformator Substitutionsliste Verwendete Partikel	Splitting $---$ $<[]>$ $<P_3[4]>$
Nachfolger korrespondiert abhängig	P_4 $(P_3[i], P_4[i]), \quad 1 \leq i \leq 4$ $(P_3[4], P_4[5])$
Nachfolger korrespondiert abhängig	P_7 $(P_3[i], P_7[i]), \quad 1 \leq i \leq 4$ $(P_3[4], P_7[5])$

Die nicht im PUHR-Tableau repräsentierte Alternative ist:

Potentielle Teilexpansion	$P_3 \rightarrow Y$
Vorgänger	P_3
Regel Transformator Substitutionsliste Verwendete Partikel	PUHR $T[6]$ $<[]>$ $<P_3[1]>$

In den Schritten $P_4 \rightarrow P_5$ und $P_7 \rightarrow P_8$ wird jeweils das Partikel $fach(tim, info)$ mit Hilfe der PUHR-Regel und des Transformators T[6] hinzugefügt. Im Schritt $P_5 \rightarrow P_6$ wird dann $\perp$ durch T[4] hergeleitet, was diesen Pfad schließt. Im Schritt $P_8 \rightarrow P_9$ kommt $wird\_etwas(tim)$ dazu. P_9 hat die Eigenschaft offen. In der tabellarischen Darstellung sieht das folgendermaßen aus:

Partikelliste	P_6
Eigenschaften Teilexpansionen Potentielle Teilexpansionen	$<geschlossen>$ ε ε

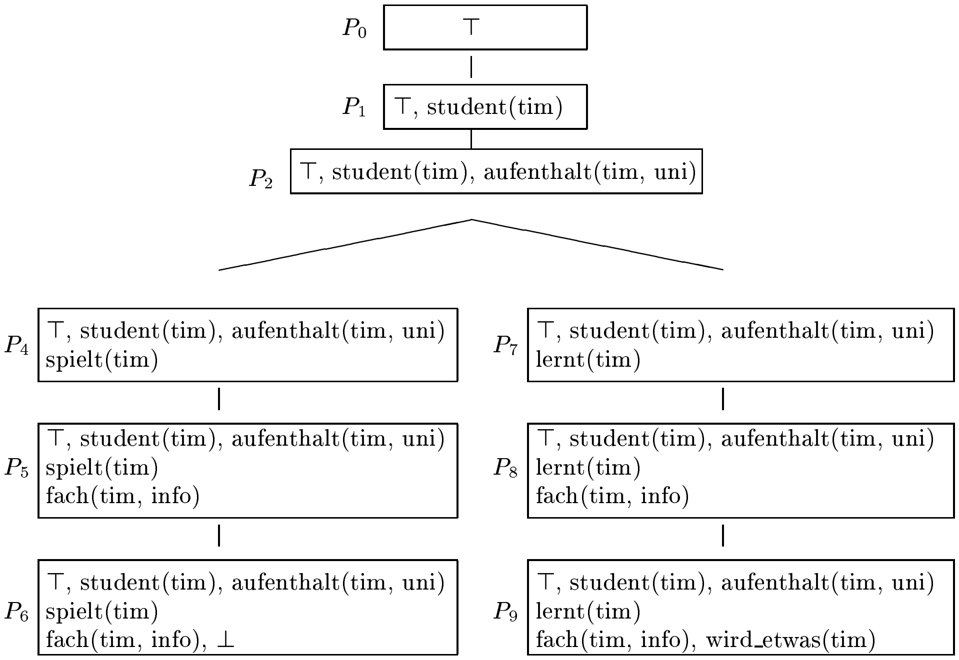
Partikelliste	P_9
Eigenschaften Teilexpansionen Potentielle Teilexpansionen	$<offen>$ ε ε

□

Eine Variante der PUHR-Tableau-Methode enthält eine einzige Regel, die die *PUHR- und Splitting-Regel kombiniert.* Die Anwendung eines Transformators $A_1 \wedge \ldots \wedge A_n \Rightarrow B_1 \vee \ldots \vee B_k$ auf eine Partikelliste P_0 erzeugt k neue Nachfolgerlisten, wobei $P_i = P_0 \circ <B_i\sigma>$ (für eine Substitutionsliste $<\sigma>$) anstatt ein Partikel $(B_1 \vee \ldots \vee B_k)\sigma$ hinzuzufügen, das möglicherweise in einer späteren Expansion gesplittet wird. PUHR + Splitting

Als Partikel kommen bei dieser Variante nur Grundatome und keine Disjunktionen vor. Die restlichen Bestandteile der Deduktionsebene bleiben unverändert.

Beispiel 2.2
Seien die Transformatoren wie in Beispiel 2.1 gegeben. Dann wird P_2 direkt zu P_4 und P_7 expandiert:



Die Partikelliste P_2 sieht dann folgendermaßen aus:

Partikelliste	P_2
Eigenschaften	ε
Teilexpansionen	$< P_2 \rightarrow P_4 P_7 >$
Potentielle Teilexpansionen	$< P_2 \rightarrow X >$

Der angewendete Schritt wird durch folgende Teilexpansion beschrieben:

Teilexpansion	$P_2 \rightarrow P_4 P_7$
Vorgänger	P_2
Regel	PUHR + Splitting
Transformator	$T[2]$
Substitutionsliste	$< [tim/Person] >$
Verwendete Partikel	$< P_2[2], P_2[3] >$
Nachfolger	P_4
korrespondiert	$(P_2[i], P_4[i]),\ 1 \leq i \leq 3$
abhängig	$(P_2[2], P_4[4])$
	$(P_2[3], P_4[4])$
Nachfolger	P_7
korrespondiert	$(P_2[i], P_7[i]),\ 1 \leq i \leq 3$
abhängig	$(P_2[2], P_7[4])$
	$(P_2[3], P_7[4])$

Bei der potentiellen Teilexpansion $P_2 \rightarrow X$ hat sich nur der Name der Regel geändert.

Potentielle Teilexpansion	$P_2 \rightarrow X$
Vorgänger	P_2
Regel Transformator Substitutionsliste Verwendete Partikel	PUHR + Splitting $T[6]$ $< [] >$ $< P_2[1] >$

□

Eine weitere Variante der PUHR-Tableau-Methode hat eine andere Splitting-Regel, nämlich die sogenannte *Complement-Splitting-Regel*. Wird P unter Verwendung eines Partikels $B_1 \vee \ldots \vee B_k$ expandiert, so ergeben sich k Nachfolger in einer Teilexpansion, mit $P_i = P \circ < B_i > \circ Neg_i$, wobei $Neg_i =< \neg B_j \mid i < j \leq k >$ ist[1].

Complement Splitting

Partikel können somit auch negierte Grundatome sein. Eine Partikelliste P hat dann die Eigenschaft geschlossen, wenn $\perp \in P$ oder es ein Grundatom A gibt, mit $A \in P$ und $\neg A \in P$.

Beispiel 2.3
Sei $\pi = B_1 \vee B_2 \vee B_3 \vee B_4$ in einer Partikelliste P enthalten. Durch Anwendung der obigen Complement-Splitting-Regel gäbe es 4 Nachfolger mit:

$$\begin{aligned} P_1 = P &\quad \circ < B_1, \neg B_2, \neg B_3, \neg B_4 > \\ P_2 = P &\quad \circ < B_2, \neg B_3, \neg B_4 > \\ P_3 = P &\quad \circ < B_3, \neg B_4 > \\ P_4 = P &\quad \circ < B_4 > \end{aligned}$$

Diese Nachfolger gehören zu einer Teilexpansion:

Partikelliste	P
Eigenschaften Teilexpansionen Potentielle Teilexpansionen	ε $< P \rightarrow P_1 P_2 P_3 P_4 >$ ε

Sei $l := |P|$.

[1] In einer anderen Variante der Regel werden die negierten Atome auch den rechts stehenden Partikellisten zugeordnet.

Teilexpansion	$P \to P_1 P_2 P_3 P_4$
Vorgänger	P
Regel Transformator Substitutionsliste Verwendete Partikel	Complement Splitting $---$ $< [] >$ $< B_1 \vee B_2 \vee B_3 \vee B_4 >$
Nachfolger korrespondiert abhängig	P_1 $(P[i], P_1[i]),\ 1 \leq i \leq l$ $(\pi, P_1[l+i]),\ 1 \leq i \leq 4$
Nachfolger korrespondiert abhängig	P_2 $(P[i], P_2[i]),\ 1 \leq i \leq l$ $(\pi, P_2[l+i]),\ 1 \leq i \leq 3$
Nachfolger korrespondiert abhängig	P_3 $(P[i], P_3[i]),\ 1 \leq i \leq l$ $(\pi, P_3[l+i]),\ 1 \leq i \leq 2$
Nachfolger korrespondiert abhängig	P_4 $(P[i], P_4[i]),\ 1 \leq i \leq l$ $(\pi, P_4[l+1])$

□

2.1.4 Modellierung von Smullyan-Tableaux in diesem Rahmen

- *Partikel*:
  Partikel sind geschlossene Formeln der Prädikatenlogik erster Stufe von beliebiger Komplexität.

- *Partikelliste*:
  Liste von Formeln, bei der die Reihenfolge keine Rolle spielt.

- *Initiale Partikelliste*:
  Die initiale Partikelliste besteht aus der vom Benutzer vorgegebenen Formel.

- *Eigenschaft*:
  Eine Partikelliste P hat die Eigenschaft „geschlossen“ genau dann, wenn es eine Formel ψ gibt, mit $\psi \in P$ und $\neg\psi \in P$.
  Eine nicht-expandierbare Partikelliste P hat die Eigenschaft „offen“ genau dann, wenn für jede Formel $\psi \in P$ gilt: $\neg\psi \notin P$.

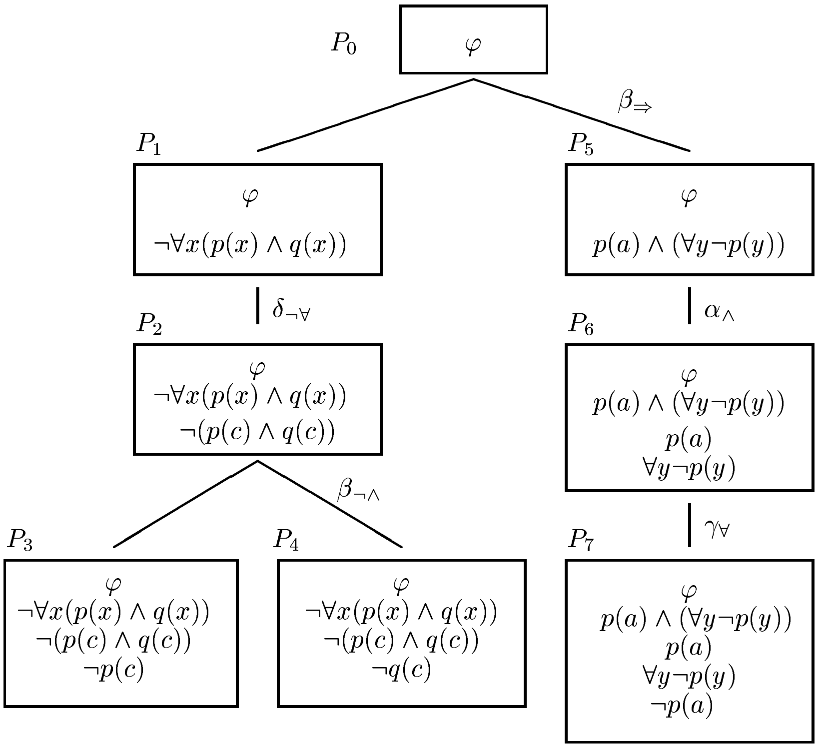
- *Transformatoren*:
  Es gibt keine Transformatoren.

- *Regeln*:
  Dies sind die $\alpha-, \beta-, \gamma-, \delta-$Regeln.

- *Substitutionen*:
  Es gibt nur die Substitutionen, die eine Teilexpansion betreffen und zwar bei einer Anwendung einer γ- oder δ-Regel. Jede Substitutionsliste ist einelementig.

- *Expansion*:
  Eine Teilexpansion, die eine Anwendung einer β-Regel beschreibt, hat zwei Nachfolger-Partikellisten, alle anderen Teilexpansionen haben nur eine.
  Das Verfahren ist monoton in dem Sinne, dass jede Nachfolger-Partikelliste Obermenge der Vorgänger-Partikelliste ist.

Beispiel 2.4
Sei die Formel $\varphi := \Big(\forall x(p(x) \wedge q(x))\Big) \Rightarrow \Big(p(a) \wedge (\forall y \neg p(y))\Big)$ gegeben. Dann ergibt sich beispielsweise folgendes Tableaux. (Aus Platzgründen wird in dieser Abbildung der Platzhalter φ verwendet.)



Die initiale Partikelliste enthält nur das Partikel φ. Darauf kann nur die Regel $\beta_\Rightarrow$ angewendet werden:

Partikelliste	P_0
Eigenschaften Teilexpansionen Potentielle Teilexpansionen	ε $< P_0 \to P_1 P_5 >$ ε

Daraus ergeben sich die zwei Nachfolger P_1 und P_5:

Teilexpansion	$P_0 \to P_1 P_5$
Vorgänger	P_0
Regel Transformator Substitutionsliste Verwendete Partikel	$\beta_\Rightarrow$ $---$ $<>$ $< P_0[1] >$
Nachfolger korrespondiert abhängig	P_1 $(P_0[1], P_1[1])$ $(P_0[1], P_1[2])$
Nachfolger korrespondiert abhängig	P_5 $(P_0[1], P_5[1])$ $(P_0[1], P_5[2])$

Auf Partikelliste P_1 kann nun die $\delta_{\neg\forall}$ - Regel angewendet werden, was zur Einführung einer neuen Konstante c führt:

Partikelliste	P_1
Eigenschaften	ε
Teilexpansionen	$< P_1 \to P_2 >$
Potentielle Teilexpansionen	ε

Teilexpansion	$P_1 \to P_2$
Vorgänger	P_1
Regel	$\delta_{\neg\forall}$
Transformator	$---$
Substitutionsliste	$< [c/x] >$
Verwendete Partikel	$< P_1[2] >$
Nachfolger	P_2
korrespondiert	$(P_1[i], P_2[i]),\ 1 \leq i \leq 2$
abhängig	$(P_1[2], P_2[3])$

Die nachfolgende Anwendung der $\beta_{\neg\wedge}$ - Regel, erzeugt die Partikellisten P_3 und P_4, die Modelle der Formel φ darstellen:

Partikelliste	P_3
Eigenschaften	$< offen >$
Teilexpansionen	ε
Potentielle Teilexpansionen	ε

Partikelliste	P_4
Eigenschaften	$< offen >$
Teilexpansionen	ε
Potentielle Teilexpansionen	ε

Betrachten wir noch die Teilexpansion $P_6 \to P_7$. Die einzige anwendbare Regel ist $\gamma_\forall$ auf das Partikel $\forall y \neg p(y)$. Die einzige in dem bisherigen Pfad vorkommende Konstante ist a. Somit wird a eingesetzt :

Partikelliste	P_6
Eigenschaften	ε
Teilexpansionen	$< P_6 \to P_7 >$
Potentielle Teilexpansionen	ε

Teilexpansion	$P_6 \to P_7$
Vorgänger	P_6
Regel	$\gamma_\forall$
Transformator	$---$
Substitutionsliste	$< [a/y] >$
Verwendete Partikel	$< P_6[4] >$
Nachfolger	P_7
korrespondiert	$(P_6[i], P_7[i]),\ 1 \leq i \leq 4$
abhängig	$(P_6[4], P_7[5])$

In P_7 haben wir den Fall, dass sowohl $p(a)$, als auch $\neg p(a)$ enthalten sind, was in diesem Verfahren einen Widerspruch repräsentiert. P_7 ist somit geschlossen:

Partikelliste	P_7
Eigenschaften	$< geschlossen >$
Teilexpansionen	ε
Potentielle Teilexpansionen	ε

In diesem Beispiel wird deutlich, dass es keine Transformatoren gibt. $\square$

Bemerkung
Es kann vorkommen, dass an einem Punkt mehrere Regeln anwendbar sind. Ferner ist es möglich, dass eine Partikelliste, die die Eigenschaft „geschlossen“ hat, noch expandierbar ist. □

Beispiel 2.5
Betrachten wir die Formel $(p \vee q) \wedge (p \wedge \neg p)$ und das folgende Tableau:

$$
\begin{array}{ll}
P_0 & \boxed{(p \vee q) \wedge (p \wedge \neg p)} \\
 & \qquad\qquad \alpha_\wedge \\
P_1 & \boxed{\begin{array}{c}(p \vee q) \wedge (p \wedge \neg p) \\ p \vee q \\ p \wedge \neg p\end{array}} \\
 & \qquad\qquad \alpha_\wedge \\
P_2 & \boxed{\begin{array}{c}(p \vee q) \wedge (p \wedge \neg p) \\ p \vee q \\ p \wedge \neg p \\ p \\ \neg p\end{array}}
\end{array}
$$

In der Expansion $P_0 \rightarrow P_1$ wird die $\alpha_\wedge$-Regel angewandt. Bei Partikelliste P_1 hat man dann die Wahlmöglichkeit zwischen der $\alpha_\wedge$-Regel mit dem Partikel $p \wedge \neg p$ und der $\beta_\vee$-Regel mit dem Partikel $p \vee q$. Im Schritt $P_1 \rightarrow P_2$ wurde die erste Möglichkeit ausgeführt.
Die Partikelliste P_2 hat nun die Eigenschaft „geschlossen“, da sowohl p, als auch $\neg p$ enthalten sind. Theoretisch könnte aber noch die $\beta_\vee$-Regel mit dem Partikel $p \vee q$ gefeuert werden, d.h. P_2 ist epxandierbar. Es gibt eine potentielle Teilexpansion, die diese Möglichkeit für P_2 repräsentiert und auch eine für P_1. □

2.1.5 Modellierung von EP-Tableaux in diesem Rahmen

- *Partikel*:
  Partikel sind PRQ-Formeln.

- *Partikelliste*:
  Liste von Formeln, bei der die Reihenfolge keine Rolle spielt.

- *Initiale Partikelliste*:
  Die initiale Partikelliste besteht aus der vom Benutzer vorgegebenen Formel.

- *Eigenschaft*:
  Eine Partikelliste P hat die Eigenschaft „geschlossen“ genau dann, wenn es eine Formel ψ gibt, mit $\psi \in P$ und $\neg\psi \in P$ oder wenn $\top \in P$.
  Eine nicht-expandierbare Partikelliste P hat die Eigenschaft „offen“ genau dann, wenn für jede Formel $\psi \in P$ gilt $\neg\psi \notin P$ und $\top \notin P$.

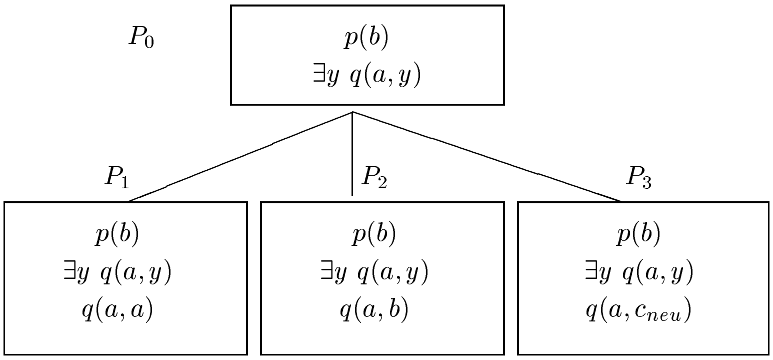
- *Transformatoren*:
  Es gibt keine Transformatoren.

- *Regeln*:
  Es gibt eine erweiterte PUHR-Regel, sowie Regeln für $\exists, \vee$ und $\wedge$.

- *Substitutionen*:
  Es gibt nur die Substitutionen, die eine Teilexpansion betreffen. Eine Substitutionsliste kann mehrelementig sein.

- *Expansion*:
  Eine Teilexpansion kann mehrere Nachfolger Partikellisten haben. Jede davon ist eine Obermenge der Vorgänger-Partikelliste.

Beispiel 2.6
Sei folgender Ausschnitt aus einem EP-Tableau gegeben:



In der Partikelliste P_0 kommen die Konstanten a und b vor, was bei der Anwendung der $\exists$-Regel in einer Teilexpansion zu 3 Nachfolger-Partikellisten führt: Je eine für jede vorkommende Konstante, sowie eine weitere mit einer neu eingeführten Konstante c_{neu}.

Partikelliste	P_0
Eigenschaften	ε
Teilexpansionen	$< P_0 \to P_1 P_2 P_3 >$
Potentielle Teilexpansionen	ε
Teilexpansion	$P_0 \to P_1 P_2 P_3$
Vorgänger	P_0
Regel	$\exists$-Regel
Transformator	$---$
Substitutionsliste	$< [a/y], [b/y], [c_{neu}/y] >$
Verwendete Partikel	$< P_0[2] >$
Nachfolger	P_1
korrespondiert	$(P_0[i], P_1[i]),\ 1 \leq i \leq 2$
abhängig	$(P_0[2], P_1[3])$
Nachfolger	P_2
korrespondiert	$(P_0[i], P_2[i]),\ 1 \leq i \leq 2$
abhängig	$(P_0[2], P_2[3])$
Nachfolger	P_3
korrespondiert	$(P_0[i], P_3[i]),\ 1 \leq i \leq 2$
abhängig	$(P_0[2], P_3[3])$

Zu beachten ist hier, dass es nur eine Teilexpansion gibt, die verschiedenen Substitutionen müssen deshalb in einer Liste angeordnet werden. Die i-te Substitution ist dabei dem i-ten Nachfolger zugeordnet.

□

2.1.6 Modellierung von SLD-Bäumen in diesem Rahmen

In diesem Abschnitt wird zunächst das SLD-Verfahren im Sinne von Lloyd behandelt, also insbesondere ohne Negation, Disjunktion und außerlogischen Prozeduren.

- *Partikel*:
  Partikel sind atomare Anfragen.

- *Partikelliste*:
  Liste von Partikeln, bei der die Reihenfolge eine wichtige Rolle spielt. Eine

solche Liste repräsentiert eine Konjunktion.

- *Initiale Partikelliste*:
  Die initiale Partikelliste besteht aus der vom Benutzer bestimmten Anfrage.

- *Eigenschaft*:
  Eine nicht-expandierbare Partikelliste P, die noch Elemente enthält, bezeichnet eine Liste von nicht beantwortbaren Anfragen und hat die Eigenschaft „gescheitert". Eine leere Liste stellt eine erfolgreich beantwortete Anfrage dar und hat die Eigenschaft „erfolgreich".

- *Transformatoren*:
  Die Transformatoren sind die Programmklauseln.

- *Regeln*:
  Die einzige Regel ist die SLD-Resolution.

- *Substitutionen*:
  Es gibt nur die Substitutionen, die eine Teilexpansion betreffen. Jede Substitutionsliste ist einelementig.

- *Expansion*:
  Eine Teilexpansion beschreibt die Ersetzung eines Partikels π der Liste mit Hilfe eines Transformators und einer Substitution durch $n \in \mathbb{N}$ Partikel, die an Stelle des π gesetzt werden.

Beispiel 2.7
Sei folgendes Prolog-Programm als Transformatorenliste gegeben:

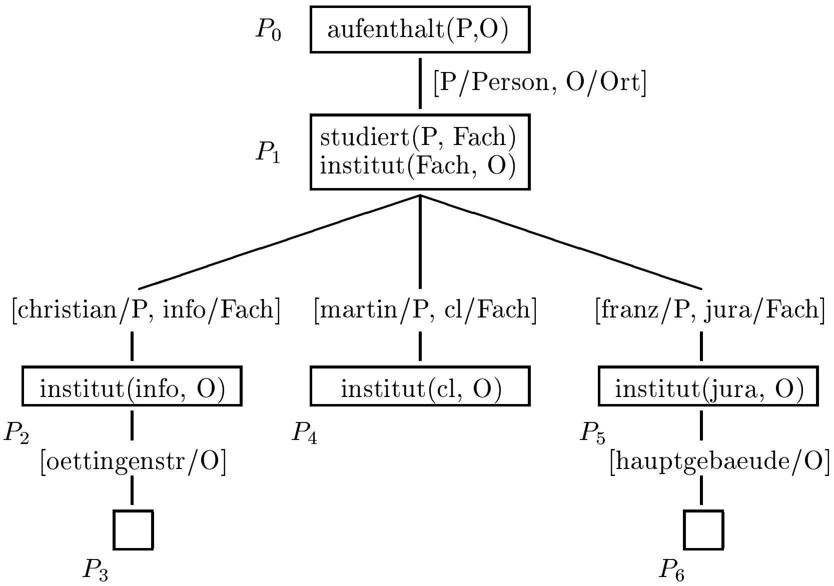
```
studiert(christian, info).                        (1)
studiert(martin, cl).                             (2)
studiert(franz, jura).                            (3)

institut(info, oettingenstr).                     (4)
institut(jura, hauptgebaeude).                    (5)

student(X) :- studiert(X, _).                     (6)

aufenthalt(Person, Ort)                           (7)
  :- studiert(Person, Fach), institut(Fach, Ort).
```

Ferner sei als Anfrage `aufenthalt(P,O)` gegeben, dann wird beispielsweise folgender SLD-Baum erzeugt:



In der obigen Abbildung enthalten die Rechtecke die Partikellisten, die Kanten sind mit den Substitutionen beschriftet, die bei der jeweiligen Teilexpansion angewendet wurden. Wir beschreiben hier die SLD-Resolution mit Selektion des ersten Partikels.

Partikelliste	P_0
Eigenschaften Teilexpansionen Potentielle Teilexpansionen	ε $< P_0 \to P_1 >$ ε

Die initiale Partikelliste enthält die Anfrage, die vom Benutzer spezifiziert wurde. Das Partikel `aufenthalt(P,O)` wird mit Hilfe des letzten Transformators auf die Partikelliste $<$ `studiert(P,Fach), institut(Fach, O)` $>$ abgebildet. Hierbei wird `Person` mit `P` und `Ort` mit `O` identifiziert:

Teilexpansion	$P_0 \to P_1$
Vorgänger	P_0
Regel Transformator Substitutionsliste Verwendete Partikel	SLD-Resolution $T[7]$ $< [P/Person, O/Ort] >$ $< P_0[1] >$
Nachfolger korrespondiert abhängig 	P_1 $---$ $(P_0[1], P_1[1])$ $(P_0[1], P_1[2]$

Für die resultierende Partikelliste P_1 ergibt sich:

Partikelliste	P_1
Eigenschaften Teilexpansionen Potentielle Teilexpansionen	ε $< P_1 \to P_2,\ P_1 \to P_4,\ P_1 \to P_5 >$ ε

Das erste Partikel `studiert(P, Fach)` wird für den nächsten Expansionsschritt verwendet. Die ersten drei Transformatoren ergeben :

Teilexpansion	$P_1 \rightarrow P_2$
Vorgänger	P_1
Regel	SLD-Resolution
Transformator	$T[1]$
Substitutionsliste	$< [christian/P, info/Fach] >$
Verwendete Partikel	$< P_1[1] >$
Nachfolger	P_2
korrespondiert	$(P_1[2], P_2[1])$
abhängig	ε

Teilexpansion	$P_1 \rightarrow P_4$
Vorgänger	P_1
Regel	SLD-Resolution
Transformator	$T[2]$
Substitutionsliste	$< [martin/P, cl/Fach] >$
Verwendete Partikel	$< P_1[1] >$
Nachfolger	P_4
korrespondiert	$(P_1[2], P_4[1])$
abhängig	ε

Teilexpansion	$P_1 \rightarrow P_5$
Vorgänger	P_1
Regel	SLD-Resolution
Transformator	$T[3]$
Substitutionsliste	$< [franz/P, jura/Fach] >$
Verwendete Partikel	$< P_1[1] >$
Nachfolger	P_5
korrespondiert	$(P_1[2], P_5[1])$
abhängig	ε

Zu beachten ist hier, dass diese Expansion aus drei verschiedenen Teilexpansionen besteht. (Wir hatten festgelegt, dass die SLD-Resolution stets das erste Partikel auswählt. Mit einer anderen Selektionsvorschrift hätte man auch mit dem zweiten Partikel `institut(Fach, 0)` fortfahren können, das sich mit den Transformatoren (4) und (5) unifizieren lässt. Dies sei dem Leser überlassen.)

Die resultierenden Partikellisten enthalten jeweils nur noch eine Anfrage. Das Partikel `institut(info, 0)` in P_2 und `institut(jura, 0)` in P_5 können erfolgreich beantwortet werden. Es ergeben sich die Partikellisten P_3 bzw. P_6:

Partikelliste	P_3
Eigenschaften	$< erfolgreich >$
Teilexpansionen	ε
Potentielle Teilexpansionen	ε

Partikelliste	P_6
Eigenschaften	$< erfolgreich >$
Teilexpansionen	ε
Potentielle Teilexpansionen	ε

Die Partikelliste $P_4 =<$ `institut(cl, 0)` $>$ hingegen kann nicht erfoglreich beantwortet werden. In diesem Fall ergibt sich eine nicht-leere und nicht-expandierbare Liste, was eine nicht beantwortbare Anfrage darstellt.

Partikelliste	P_4
Eigenschaften	$< gescheitert >$
Teilexpansionen	ε
Potentielle Teilexpansionen	ε

□

Bisher gingen wir von Partikeln aus, die atomare Formeln waren und haben Disjunktionen und Negationen nicht betrachtet.

Für die Behandlung von *Disjunktionen* muss man komplexere Partikel zulassen. Alle Partikel stammen aus einer Formelmenge F, die wie folgt definiert ist:

Disjunktion

- Alle Atome sind in F.
- Disjunktionen von Formeln aus F sind ebenfalls in F.
- Konjunktionen von Formeln aus F sind ebenso in F.

Beispielsweise sind `a;b` oder auch `a;(b,c)` in F enthalten.

Eine zusätzliche *Splitting-Regel* muss einen nicht-atomaren Partikel zerlegen, damit dieser mit Hilfe der Resolutionsregel weiterverarbeitet werden kann. Dies hat das Deduktionssystem zu leisten.

Beispiel 2.8

Sei das Partikel $((\mathtt{a},\mathtt{b});(\mathtt{c},(\mathtt{d};\mathtt{e})))$ gegeben, was für die Formel $\varphi = (a \wedge b) \vee (c \wedge (d \vee e))$ steht.

Eine Partikelliste $P = < ((\mathtt{a},\mathtt{b});(\mathtt{c},(\mathtt{d};\mathtt{e}))),\mathtt{f},\mathtt{g} >$ könnte durch eine Splitting-Regel beispielsweise auf folgendes expandiert werden:

$$\begin{aligned} P_1 &= < \mathtt{a},\mathtt{b},\mathtt{f},\mathtt{g} > \\ P_2 &= < \mathtt{c},(\mathtt{d};\mathtt{e}),\mathtt{f},\mathtt{g} > \end{aligned}$$

□

Die Behandlung von *negierten Formeln* erfordert die Möglichkeit von Nebenrechnungen. Wird ein Partikel $\pi = \mathtt{not}\ \varphi$ bearbeitet, wird in einer Nebenrechnung φ ausgewertet. Bei Erfolg scheitert π, ansonsten ist π erfolgreich. Für die Nebenrechnung sind keine anderen Modellierungsmittel erforderlich als die bisher vorgestellten. Zur Darstellung wäre es sinnvoll, die Nebenrechnung in einem separaten Fenster darzustellen.

Negation

Prolog wählt stets das erste Partikel für den nächsten Resolutionsschritt. Für die allgemeine SLD-Resolution könnte man eine *Selektionsregel* einführen, die vor jeder Anwendung der Resolution einen Partikel als Nächsten auszeichnet. Die Selektion wird durch eine eigene Teilexpansion modelliert.

In diesem Rahmen könnte diese Auszeichnung etwa durch Umsortierung der Partikelliste repräsentiert werden.

Selektion

Bisher haben wir zu jeder Teilexpansion die dazugehörige Substitution angegeben, ohne die vorangegangenen zu berücksichtigen. Ein zentraler Punkt bei diesem Verfahren ist aber das *Aufsammeln der Antwortsubstitution*. Dies kann mit unseren Mitteln durch Mitführen aller bisherigen Substitutionen durch das Deduktionssystem geschehen. Als Antwort werden dann die Bindungen der Variablen aus der

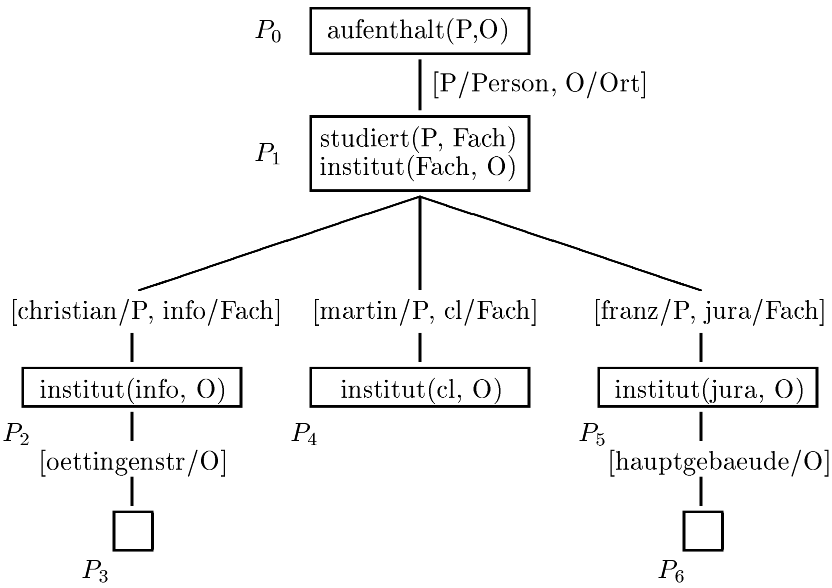
Aufsammeln der Antwortsubstitution

Anfrage aufgefasst.
Im Grunde gibt es drei Arten von Substitutionen bei der SLD-Resolution, die interessant für den Benutzer sein können:

- Lokale Substitutionen
- Kombination aller Substitutionen entlang eines Pfades
- Restriktion der Kombination aller Substitutionen auf die Variablen, die in der Anfrage enthalten sind. (Die eigentliche Antwortsubstitution)

Es sollte klar sein, dass alle drei Arten leicht vom Deduktionssystem geliefert werden können. An die Darstellungsebene könnten diese Informationen durch dreielementige Substitutionslisten weitergegeben werden. Die Interpretation dieser Daten ist - wie gehabt - dem Benuzter überlassen.

Beispiel 2.9
Betrachten wir noch einmal den SLD-Baum aus einem vorigen Beispiel:



An den Kanten sind jeweils die lokalen Substitutionen zu finden. Geht man den Pfad P_0, P_1, P_2, P_3 entlang, so ergeben sich nacheinander folgende Kombinationen (zur Vollständigkeit sind auch X/X aufgeführt, für Variablen $X \notin edom(\sigma)$):

$$
\begin{aligned}
\Sigma_1 &= < [P/P, O/O, P/Person, O/Ort, Fach/Fach] > \\
\Sigma_2 &= < [christian/P, O/O, christian/Person, info/Fach, O/Ort] > \\
\Sigma_3 &= < [christian/P, oettingenstr/O, christian/Person, info/Fach, \\
&\quad oettingenstr/Ort] >
\end{aligned}
$$

Die eigentliche Antwortsubstitution ist die Restriktion von Σ_3 auf P und O, also $[christian/P, oettingenstr/O]$. □

2.1.7 Modellierung von proof-trees in diesem Rahmen

Man betrachte ein Partikel $\pi \in P$ in einem SLD-Baum . Dann heißt ein Partikel π' aus einer Nachfolgerliste P' *direkter Nachfolger*, wenn π und π' in der Relation direkter Nachfolger

„abhängig“ oder „korrespondiert“ stehen.
Notiert wird dies kurz mit $\pi \rightarrow \pi'$. Ein π' ist dann ein *Nachfolger* eines Partikels π, wenn $\pi \rightarrow^{+} \pi'$ gilt, wobei $\rightarrow^{+}$ die transitive Hülle der Relation $\rightarrow$ bezeichnet.

Für ein festes $\pi \in P$ kann man nun die Nachfolger in den Pfaden zu den Blättern mit der Eigenschaft „erfolgreich“ betrachten und daraus eine Begründung für die Gültigkeit des Partikels π ableiten. Daraus ergibt sich aus einem SLD-Baum ein sogenannter *proof-tree*.

Die initiale Partikelliste dieses Baums enthält ein einziges Partikel. Hat eine Partikelliste $P =< \pi >$ die direkten Nachfolger $\pi_1, \ldots, \pi_n$, so gibt es in dem proof-tree eine Teilexpansion mit n Nachfolgerpartikellisten $P_i =< \pi_i >$.

- *Partikel*:
  Partikel sind atomare Formeln und stellen Anfragen dar.

- *Partikelliste*:
  Jede Partikelliste, die kein Blatt ist, ist einelementig. Die Blätter enthalten stets die leere Partikelliste. Die Nachfolger einer Partikelliste sind als Konjunktion aufzufassen.

- *Initiale Partikelliste*:
  Die initiale Partikelliste besteht aus der vom Benutzer bestimmten Anfrage.

- *Eigenschaft*:
  Eine Partikelliste, die ein Blatt ist, hat stets die Eigenschaft „erfolgreich“. Alle anderen Partikellisten sind expandierbar.

- *Transformatoren*:
  Die Transformatoren sind Programmklauseln.

- *Regeln*:
  Es gibt eine kombinierte SLD+Splitting-Regel. Die SLD-Resolutionsregel des Kalküls wird dabei kombiniert mit einer Splitting-Regel, die nicht zum Kalkül gehört, sondern nur zur Darstellung von Teilstrukturen der Ableitung in Form eines Baums dient. (Alternativ könnte man die beiden Regeln SLD-Resolution und Splitting auch trennen.)

- *Substitutionen*:
  Es gibt nur die Substitutionen, die eine Teilexpansion betreffen. Jede Substitutionsliste ist einelementig.

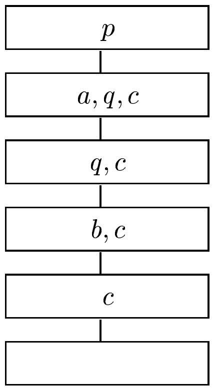
- *Expansion*:
  Es gibt zwei Arten von Teilexpansionen: Eine Teilexpansionsart beschreibt die Begründung eines Partikels (entspricht dem Kopf einer Instanz einer Programmklausel) durch eine Menge von einelementigen Nachfolgerpartikellisten (entspricht dem Rumpf deselben Klauselinstanz).
  Die andere Teilexpansionsart beschreibt Instantiierungen.
  Jede Partikelliste, die kein Blatt ist, ist Ausgangspunkt genau einer Teilexpansion.

Beispiel 2.10
Sei folgende Transformatorenliste gegeben:

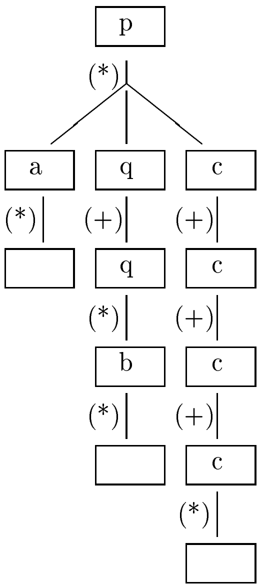
```
p :- a,q,c.
q :- b.
a.
b.
```

`c.`

Die Anfrage `p` liefert folgenden SLD-Baum:



Daraus leitet sich folgender proof-tree ab, der eine Begründung für die Gültigkeit von p liefert:



(*) Abhängigkeitsbeziehung
(+) Korrespondenzbeziehung

In diesem Beispiel ist die SLD-Resolutionsregel und die Splitting-Regel kombiniert, was sich darin auswirkt, dass die intiale Partikelliste drei Nachfolgerpartikellisten hat. Bei getrennten Regeln gäbe es noch eine Partikelliste dazwischen mit der Partikelliste $< a, q, c >$. □

2.1.8 Allgemeine Abhängigkeitsbäume

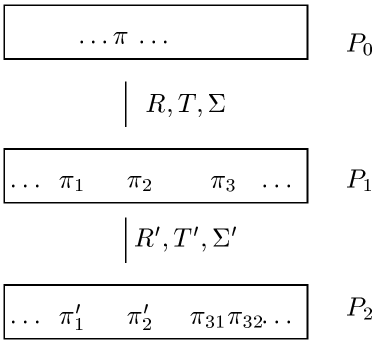
Die Konstruktion der proof-trees auf Basis der SLD-Bäume kann allgemein auch auf andere Bäume angewendet werden. Sei ein Ableitungsbaum mit folgenden Einschränkungen gegeben:

- Es gibt nur eine Nachfolgerpartikelliste pro Teilexpansion.
- Pro Teilexpansion wird genau ein Partikel verwendet.

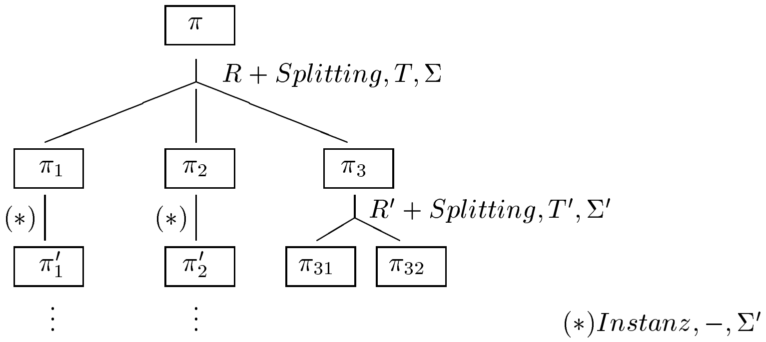
Anhand der Nachfolger-Relation kann man durch Einfügen einer Splitting-Regel einen Baum der Partikelabhängigkeiten erzeugen. (Die Splitting-Regel gehört wiederum nicht zum Kalkül, sondern dient nur zu Darstellungszwecken.)

Die initiale Partikelliste dieses Baums enthält ein einziges Partikel. Hat eine Partikelliste $P = <\pi>$ die direkten Nachfolger $\pi_1, \ldots, \pi_n$, so gibt es in dem Abhängigkeitsbaum eine Teilexpansion mit n Nachfolgerpartikellisten $P_i = <\pi_i>$.

Zur Veranschaulichung betrachten wir folgenden Teil eines Ableitungsbaums:



Von $\pi \in P_0$ sind die Partikel π_1, π_2, π_3 direkt abhängig. Im Schritt $P_0 \rightarrow P_1$ wurde Regel R mit Transformator T und Substitutionsliste Σ angewendet.
Im Schritt $P_1 \rightarrow P_2$ wurde die Regel R', mit Transformator T' und Substitutionsliste Σ' angewendet. Die Partikel π_1, π_2 korrespondieren mit π_1', π_2', während π_{31}, π_{32} von π_3 abhängen. Daraus ergibt sich folgender Abhängigkeitsbaum:



Eine andere Variante könnte die Abhängigkeitsrelation in Rückwärtsrichtung darstellen. Die oben aufgeführten Voraussetzungen müssen dann entsprechend abgeändert werden.

2.1.9 Modellierung von DDB trees in diesem Rahmen

DDB trees stehen für einen ganzen Wald von proof trees, die i.A. verschiedene Struktur aufweisen. Es ist möglich, diese Bäume mit unseren Mitteln darzustellen. Dazu sind insbesondere Substitutionslisten nötig.
Eine ausführliche Besprechung dieser Bäume würde den Rahmen der Projektarbeit sprengen, ist aber in [Mal97] zu finden.

2.1.10 Ein nicht-logisches Beispiel: Syntaxbäume

In diesem Abschnitt werden Syntaxbäume für kontextfreie Grammatiken betrachtet. Eine kontextfreie Grammatik G ist ein Viertupel (V, Σ, P, S) mit:

- V eine endliche Menge von Variablen, auch Nichtterminalzeichen genannt.
- Σ ein nicht-leere endliche Menge von Terminalzeichen. (Σ und V sind disjunkt.)
- P die Menge der Produktionen.
- $S \in V$ das Startsymbol.

Zu beachten ist hier, dass Σ und P in diesem Kontext ausnahmsweise nicht für Substitutionen und Partikellisten stehen! Man kann Syntaxbäume einer kontextfreien Grammatik folgendermaßen beschreiben:

- *Partikel*:
  Partikel sind Terminal- oder Nichtterminalzeichen, d.h. $\pi \in \Sigma \cup V$.
- *Partikelliste*:
  Partikellisten sind stets einelementig.
- *Initiale Partikelliste*:
  Die initiale Partikelliste enthält das Startsymbol.
- *Eigenschaft*:
  Eine Partikelliste ist genau dann expandierbar, wenn sie ein Nichtterminalzeichen enthält.
- *Transformatoren*:
  Die Transformatoren sind die Produktionen.
- *Regeln*:
  Es gibt nur eine Regel, die aus einem Nichtterminalzeichen A in einer Partikelliste und einer Produktion $A \rightarrow w_1 \dots w_n$ $(w_i \in \Sigma \cup V)$ n Nachfolger erzeugt, mit $P_i =< w_i >$.
- *Substitutionen*:
  Es gibt keine Substitutionen.
- *Expansion*:
  Eine Teilexpansion beschreibt eine Ableitung $A \rightarrow w_1 \dots w_n$.

2.1.11 Grenzen des begrifflichen Rahmens

Es wurden eine Reihe von vereinfachenden Annahmen getroffen, die zur Übersichtlichkeit eingeführt wurden. Zu nennen sind vor allem:

- Partikel sind unstrukturiert und können nur angezeigt werden.
- Substitutionen werden vom System ebenfalls nur angezeigt, nicht aber interpretiert.

Möglicherweise gibt es Situationen, in denen diese Annahmen eine zu große Einschränkung darstellen: Im Sequenzenkalkül beispielsweise könnte man durch strukturierte Partikel $\Gamma \Rightarrow \varphi$ den Kontext Γ und die Formel φ trennen. Andrerseits könnte es wünschenswert sein, Substitutionen interpretieren oder gar anwenden zu können, beispielsweise für die proof-trees.

Ferner gibt Verfahren mit Konstrukten, die mit den bisherigen Mitteln nicht beschrieben werden können. Ein Beispiel dafür ist die Einführung und Beseitigung von Annahmen im NK-Kalkül von Gentzen.

2.2 Die Darstellungsebene

Das vorrangige Ziel ist die Visualisierung der oben besprochenen Objekte in einem Baum. Für Nebenrechnungen, etwa für die Negation bei SLD-Bäumen, benötigt man separate Bäume.

Ein Darstellen aller Informationen ist im Allgemeinen der Übersichts halber nicht wünschenswert, so dass eine entsprechende Auswahl getroffen werden muss.

2.2.1 Darstellungsmöglichkeiten

Die folgenden Bestandteile des Baumes können beschriftet werden:

- Knoten
- Kanten

Zudem hat man auch einen „aktuellen“ Knoten, über den man weitere Informationen, beispielsweise in tabellarischer Form darstellen kann.

2.2.2 Korrespondenz zwischen Ableitungs- und Darstellungsbäumen

Man muss eine Korrespondenz zwischen Ableitungsbaum A und dargestellten Baum B finden. Es bietet sich an, den Knoten von B die Partikellisten von A gegenüberzustellen.
Das heißt aber nicht, dass ein Knoten nur mit Informationen zu Partikellisten beschriftet sein kann. Es ist durchaus wünschenswert, Informationen zu den bevorstehenden aber auch zu den gerade abgeschlossenen (potentiellen) Teilexpansionen darzustellen.
Analoges gilt für die Beschriftung der Kanten.

2.2.3 Die darstellbaren Informationen

Die darstellbaren Informationen kann man folgendermaßen klassifizieren:

- Informationen zu einer Partikelliste
  - Partikelliste
  - Inkrement der Partikelliste (bei monotonen Verfahren)
  - Änderung der Partikelliste (Inkrement und Dekrement)
  - Eigenschaften
- Informationen zu Teilexpansionen
  - Vorgängerpartikelliste
  - Verwendete Partikel für eine Teilexpansion.
  - Verwendete Regel bei einer Teilexpansion.
  - Transformator
  - Substitutionsliste
  - Relationen korrespondiert und abhängig.
- Informationen zu potentielle Teilexpansion
  - Vorgängerpartikelliste

- Verwendbare Regel bei einer potentiellen Teilexpansion.
- Transformator
- Substitutionsliste
- Verwendbare Partikel für eine potentielle Teilexpansion.

A Symbolverzeichnis

Symbol	Bedeutung
$P, P_1, \ldots$	Partikelliste
π	Partikel
φ	Formel
σ	Substitution
Σ	Substitutionsliste

Literatur

[Ket97] Mathias Kettner. Ein System zur Konstruktion, Modifikation und Untersuchung von PUHR-Tableaux. Diplomarbeit, Ludwig-Maximilians-Universität München, 1997.

[Llo97] J.W. Lloyd. Foundations of Logic Programming. Springer-Verlag, Heidelberg, 1987. Second edition.

[Mal97] S. Mallet, M. Ducassé. DDB trees: a basis for deductive database explanations. Universität von Rennes, Frankreich, 1997.

[Tor98] Sunna Torge. Überprüfung der Erfüllbarkeit im Endlichen: Ein Verfahren und seine Anwendungen, Dissertation, Ludwig-Maximilians-Universität München, 1998.