DAI-Labor
TU Berlin

Grundlagen der Künstlichen Intelligenz

Logikbasierte Agenten 2

Prädikatenlogik

24.11.2005

Stefan Fricke
stefan.fricke@dai-labor.de

AIOIT
Agententechnologien in betrieblichen Anwendungen und der Telekommunikation

Gliederung

- **Einleitung**
- **Prädikatenlogik**
  - Formalismus
  - Beispiele
- **Inferenzen in der Prädikatenlogik**
  - Propositionale Inferenz
  - Unifikation
  - Resolution
- **Zusammenfassung**

Aussagenlogik revisited

- **Repräsentation der Wissensbasis durch Formeln der Aussagenlogik**
  - Formeln bestehen aus mit Junktoren verknüpften Aussagen
    - inklusive $\top$ und $\bot$
- **Interpretation I weist jeder Aussage einen Wahrheitswert zu**
- **Modell ist I mit für $\top$ für jede Formel in Wissensbasis**
- **Logische Folgerbarkeit ist durch Wahrheitstabellen beweisbar**

Aussagenlogik revisited

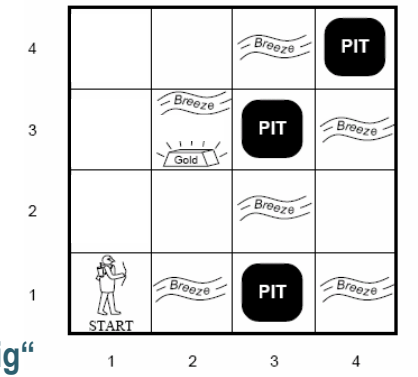
- ⇨ **WB $\models \phi$ durch Widerspruch beweisen:** $WB \cup \neg \phi \models \bot$
- ⇨ **Resolutionsmethode ist ein Inferenzmechanismus basierend auf Formeln konjunktiver Normalformen**
  - → Aus 2 Formeln wird eine neue Formel generiert und der Klauselmenge hinzugefügt
- ⇨ **Resolutionsmethode ist konstruktiv, vollständig und korrekt**

Resolution arbeitet auf Formeln und nicht auf Termen

2) Resolution ersetzt nicht sondern *erzeugt* aus zwei Formeln eine neue Formel, die dann zur Klauselmenge hinzugefügt wird.

Schwächen der Aussagenlogik

- **Modellierung von Umgebungen mit sehr vielen Objekten ist schwierig**
  - $\phi_1 : b[1,1] \Leftrightarrow (p[1,2] \vee p[2,1])$
  - …
  - $\phi_{16}: b[4,4] \Leftrightarrow (p[4,3] \vee p[3,4])$



- **Gesucht ist eine kompaktere Form, wie „Nachbarfelder von Falltüren sind windig“**
  - analog zur menschlichen Sprache, die für die Beschreibung von Situationen Objekte und Relationen / Eigenschaften verwendet.

Für jedes Quadrat und jede „Eigenschaft“ b muss eine Regel formuliert werden.

analog zur menschlichen Sprache, die im Wesentlichen auf Objekten und Relationen (auch: Eigenschaften) beruht, wenn es um Sachverhalte / Situationen geht.

Generell gilt: Aussagenlogik ist sehr ausdruckschwach.

Von der Aussagenlogik zur Prädikatenlogik

- **Aussagenlogik beschreibt die Welt als eine Menge von Fakten**
  - Symbole mit Verknüpfungen: $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$

- **Prädikatenlogik beschreibt die Welt durch**
  - Objekte: Orte, Personen, …
  - Relationen: links_von/2, windig, …
  - Funktionen: links_von/1, successor/1

Gliederung

- **Einleitung**
- **Prädikatenlogik**
  - Formalismus
  - Beispiele
- **Inferenzen in der Prädikatenlogik**
  - Propositionale Inferenz
  - Unifikation
  - Resolution
- **Zusammenfassung**

Prädikatenlogik (First Order Logic, FOL) — Terme

- **Konstanten und Variablen sind Terme.**
  - Konstanten beginnen mit Großbuchstaben, z.B. A, B, Wert
  - Variablen beginnen mit Kleinbuchstaben, z.B. x, y, variable

- **Falls *f* ein Funktionssymbol der Stelligkeit *k* ist und falls t_1, … t_k Terme sind, so ist auch $f(t_1,\ldots,t_k)$ ein Term.**

Wir sprechen im Folgenden von der Prädikatenlogik erster Stufe, auch First Order Logic (FOL) genannt. In der Prädikatenlogik erster Stufe beziehen sich Quantoren auf Objekte. In der Prädikatenlogik zweiter Stufe sind hingegen auch Quantifizierungen von Eigenschaften von Objekten erlaubt. Beispielsweise lässt sich der Satz „Zwei Objekte sind genau dann gleich, wenn sie sich durch keine Eigenschaft unterscheiden“ nicht in der FOL, wohl aber in der Prädikatenlogik zweiter Stufe darstellen.

Prädikatenlogik (First Order Logic, FOL) — Formeln

- Falls *P* ein Prädikatsymbol der Stelligkeit *k* ist, und falls t_1, ... t_k Terme sind, dann ist $P(t_1,\ldots,t_k)$ eine (atomare) Formel.
- Für alle Formeln F und G sind auch
  $\neg F$, $F \wedge G$, $F \vee G$, $F \Rightarrow G$, $F \Leftrightarrow G$ Formeln
- Für alle Formeln F und G ist auch $F = G$ eine Formel.
- Falls x eine Variable ist und F eine Formel, so sind auch
  $\exists x\ F$ und $\forall x\ F$ Formeln.

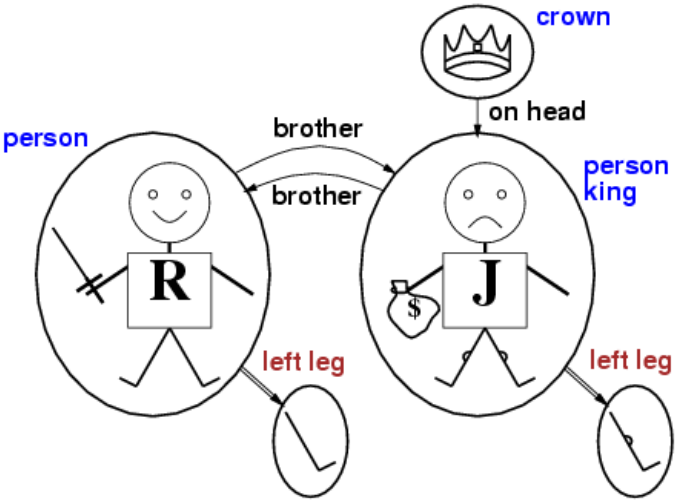
Die Begriffe „Folgerbarkeit“, „Modell“ und „erfüllbar“ lassen sich direkt aus der Aussagenlogik übertragen.

- **Formeln sind wahr hinsichtlich einer Interpretation**
- **Interpretation bildet ab:**
  - Konstanten → Objekte
  - Prädikatensymbole → Relationen
  - Funktionssymbole → Funktionen
- **Eine Formel $P(t_1,\ldots,t_k)$ ist wahr, genau dann wenn die referenzierten Objekte $t_1,\ldots,t_k$ die Relation des Prädikats P darstellen.**

Formeln sind wahr hinsichtlich einer Interpretation (Modellbegriff wie in der Aussagenlogik)

FOL — Beispiel

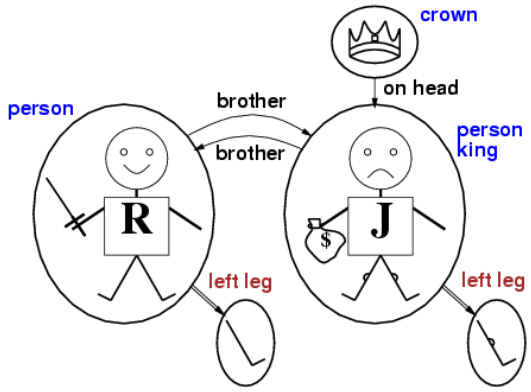
- 5 Objekte {R,J, 2x left_leg, crown}
- 2-stellige Relationen {brother, on_head}
- Unäre Relationen {person, crown, king}
- Unäre Funktion {left_leg}



5 Objekte, 2 zweistellige Relationen, 3 einstellige Relationen (blaue Label an den Objekten), 1 einstellige Funktion (left_leg)

Beispiel: Interpretation

- **Intendierte Interpretation:**
  - R ⟶ Richard Löwenherz, J ⟶ König John, Crown ⟶ Krone
  - brother ⟶ Bruderrelation, usw.
- **Viele weitere Interpretationen möglich!**



Allein für die 5 Objekte existieren 25 Permutationen.

- **Eigenschaften von Objekten aufzählen ist sehr aufwändig:**
  z.B. „Ein König ist immer auch eine Person“
  - King(J) $\Rightarrow$ Person(J).
  - King(Gustaf) $\Rightarrow$ Person(Gustaf).
  - …
- **Mittels Quantoren lassen sich Eigenschaften von Mengen von Objekten eleganter ausdrücken, z.B.**
  - $\forall x$ King(x) $\Rightarrow$ Person(x). (Alle Könige sind Personen)
  - $\exists x$ Crown(x) $\wedge$ OnHead(x, J) (J trägt eine Krone)

2 Quantoren in der FOL: der Allquantor $\forall x$ (für alle möglichen x der Wissensbasis gilt: …) und der Existenzquantor $\exists x$ (für mindestens ein x der Wissensbasis gilt: …)

$\forall x$ P(x) ist wahr genau dann, wenn P(x) wahr für jedes Objekt x.

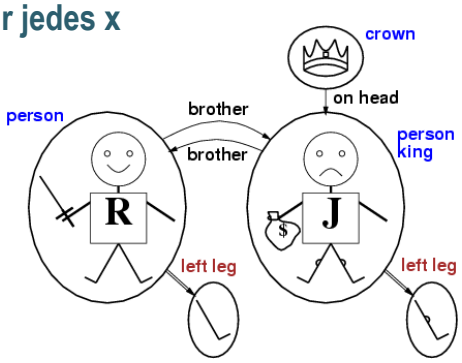
- 5 mögliche Instanzen für x: R, J, Crown, left_leg(R), left_leg(J).

⇨ **$\forall x$ King(x) $\Rightarrow$ Person(x) ist wahr gdw.**
King(x) $\Rightarrow$ Person(x) ist wahr für jedes x

- Beweisführung über Wahrheitstabelle

⇨ **$\forall x$ King(x) $\wedge$ Person(x)**
ist falsch!

- z.B. für x = Crown unerfüllbar.



Quantoren binden schwach, d.h. sie erstrecken sich über den gesamten folgenden Ausdruck.

King(J) $\Rightarrow$ Person(J): Prämisse UND Konklusion wahr.

für alle andere x ist die Prämisse falsch und damit der Term wahr.

Die logische Implikation ist die “natürliche” Verknüpfung des Allquantors.

Ein anderes korrektes Beispiel: $\forall$x,y (Crown(x) $\wedge$ OnHead(x,y)) $\Rightarrow$ king(y)

∃x P(x) ist wahr gdw. P(x) wahr für mindestens ein Objekt x.

⇨ **∃x King(x) ∧ Person(x) erfüllbar für x=J**

⇨ **Die Verwendung der logischen Implikation ist üblicherweise nicht gewollt, da zu ausdrucksschwach.**

→ Beispiel:

∃x Crown(x) ⇒ OnHead(x, J) ist wahr, jedoch für viele Instanzen sinnlos, z.B. Crown(R) ⇒ OnHead(R, J).

Das logische und ist die „typische“ Verknüpfung in Verbindung mit dem Existenzquantor.

∃x At(x) ⇒ P(x) ist wahr, sofern ein x existiert, für das At(x) falsch ist.

Anderes korrektes Beispiel: ∃x Crown(x) ∧ OnHead(x, J) ist wahr für x = Crown.

- **$\exists x\ \forall y\ P(x, y)$ ist nicht dasselbe wie $\forall y\ \exists x\ P(x, y)$**
  - Z.B. Interpretation: Loves(x, y) ⟶ x liebt y
  - **$\exists x\ \forall y$ Loves(x, y)**
    „Es gibt eine Person, die alle Personen auf der Welt liebt“
- **Es gelten die deMorgan’schen Regeln:**
  - $\forall x\ \neg P \Leftrightarrow \neg\exists x\ P$
  - $\neg\forall x\ P \Leftrightarrow \exists x\ \neg P$
  - $\forall x\ P \Leftrightarrow \neg\exists x\ \neg P$
  - $\neg\forall x\ \neg P \Leftrightarrow \exists x\ P$

$\forall x$ Likes(x, IceCream)
$\Leftrightarrow$
$\neg\exists x\ \neg$Likes(x, IceCream)

Andere Beispiele:

$\forall y\ \exists x$ Loves(x,y) : „Jede Person wird von irgend einer Person geliebt“

$\exists x$ Likes(x,Broccoli) $\Leftrightarrow$ $\neg\forall x\ \neg$Likes(x,Broccoli)

- $t_1 = t_2$ ist wahr, gdw. $t_1$und t_2 auf dasselbe Objekt referenzieren.

- z.B.: “R hat mindestens 2 Brüder”:

  $\exists x,y\ \text{Brother}(x, R) \wedge \text{Brother}(y, R) \wedge \neg(x = y)$

- Hingegen bedeutet $\exists x,y\ \text{Brother}(x, R) \wedge \text{Brother}(y, R)$

  “R hat mindestens einen Bruder” (erfüllbar auch für x=y)

Gliederung

- **Einleitung**
- **Prädikatenlogik**
  - Formalismus
  - Beispiele
- **Inferenzen in der Prädikatenlogik**
  - Propositionale Inferenz
  - Unifikation
  - Resolution
- **Zusammenfassung**

Logikbasierte (wissensbasierte) Agenten, revisited

```
function KB-Agent( percept ) returns action
    static KB, t                          // Knowledge Base, time counter
    TELL( KB, Make_Percept_Sentence( percept, t ))
    action := ASK( KB, Make_Action_Query( t ))
    TELL( KB, Make_Action_Sentence( action, t ))
    t := t + 1
    return action
```

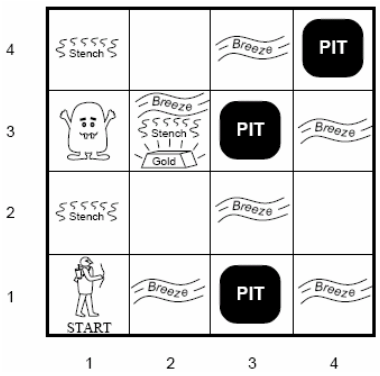

Die KB besteht aus einer Menge von Sätzen (z.B. Aussagen, Regeln), die in einer formalen Wissensrepräsentationssprache beschrieben sind und Aussagen über die Welt machen. Wissensbasierte Agenten leiten ihre Aktionen aus dieser KB ab. Sie müssen entsprechend in der Lage sein, Anfragen (ASK) an die WB zu stellen und auch neue Sätze hinzuzufügen (TELL). Von der Umgebung bekommt ein Agent Wahrnehmungen (percepts) mitgeteilt, die dieser in die KB einträgt. Agieren tut der Agent mittels der aus der WB abgeleiteten Aktion (action). Das 2. TELL wird durchgeführt, damit der Agent selbst weiss, dass er eine Aktion auszuführen gedenkt.

MAKE-PERCEPT-SENTENCE erzeugt den Satz, dass der Agent die Wahrnehmung zu einer bestimmten Zeit gemacht hat, MAKE-ACTION-QUERY erzeugt einen Satz (Anfrage), zur Durchführung einer Aktion zum aktuellen Zeitpunkt. Die Details des Inferenzmechanismus verstecken sich in TELL und ASK, sie werden später betrachtet. TELL fügt einen Satz der Wissensbasis hinzu, ASK durchsucht die Wissensbasis.

FOL-Beispiele

Agenten in der Wumpus Welt

- ⇨ **Sensoren** für Stench, Breeze, Glitter
- ⇨ **Aktionen**: Linksdrehung, Rechtsdrehung, Vorwärts laufen, Greifen
- ⇨ **Anfrage** nach bester Aktion



Charakterisierung der Wumpus-Umgebung:
¬beobachtbar ∧ deterministisch ∧ ¬episodisch ∧ statisch ∧ diskret ∧ single-agent

Bump: Agent läuft aus dem Feld (wird zurückgewiesen)

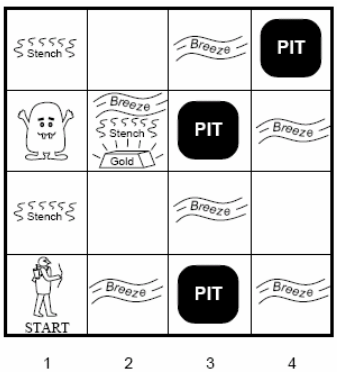
Percept([Stench,Breeze,None,None,None],5): Zum Zeitpunkt t=5 die Wahrnehmung auf dem Startfeld. Percept ist ein 2-stelliges Prädikat mit einer 5-elementigen Liste als erstes Argument. Wahrnehmung erfolgt immer von direkt benachbarten Felder, aber nicht von diagonalen “Nachbarfeldern”.

Charakterisierung der Wumpus-Welt: ¬beobachtbar (wegen lokaler Wahrnehmung); deterministisch (Effekt exakt spezifiziert); ¬episodisch (da Aktionen strikt sequenziell); statisch (Weder PIT noch Wumpus bewegen sich); diskret (keine kontinuierlichen Signale); single-agent (Wumpus ist kein Agent).

FOL-Beispiele

Repräsentation der Wumpus Welt

- ⇨ **Sensorik: Percept(Stench, Breeze, None, 5)**
- ⇨ **Aktionsterme: Turn(Right), Turn(Left), Forward, Grab.**
- ⇨ **Anfrage (beste Aktion): ∃a BestAction(a, 5)**
  - → ASK antwortet mit z.B. a = Forward
- ⇨ **Wahrnehmung**
  - → ∀b,g,t Percept(Smell, b, g, t) ⇒ Smell(t)
  - → ∀s,g,t Percept(s, Breeze, g, t) ⇒ Breeze(t)
  - → ∀s,b,t Percept(s, b, Glitter, t) ⇒ Glitter(t)



Bump: Agent läuft aus dem Feld (wird zurückgewiesen)

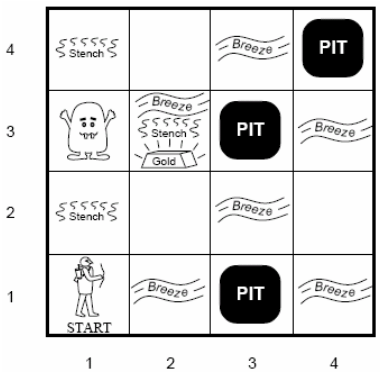
Percept([Stench,Breeze,None,None,None],5): Zum Zeitpunkt t=5 die Wahrnehmung auf dem Startfeld. Percept ist ein 2-stelliges Prädikat mit einer 5-elementigen Liste als erstes Argument. Wahrnehmung erfolgt immer von direkt benachbarten Felder, aber nicht von diagonalen “Nachbarfeldern”.

Wahrnehmung: Zeit t ist allquantifiziert. Aus Wahrnehmungen werden Eigenschaften von Orten abgeleitet.

FOL-Beispiele

Repräsentation der Wumpus Welt

- ⇨ **Reflex-Verhalten**
  - → $\forall t\ Glitter(t) \Rightarrow BestAction(Grab, t)$
- ⇨ **Reflex mit internem Zustand**
  - → $\forall t\ Glitter(t) \wedge \neg Holding(Gold, t) \Rightarrow BestAction(Grab, t)$
- ⇨ **Holding(Gold, t) kann nicht über Perceptions beobachtet werden, muss also im Zustand / in der Wissensbasis vermerkt werden**



Bump: Agent läuft aus dem Feld (wird zurückgewiesen)

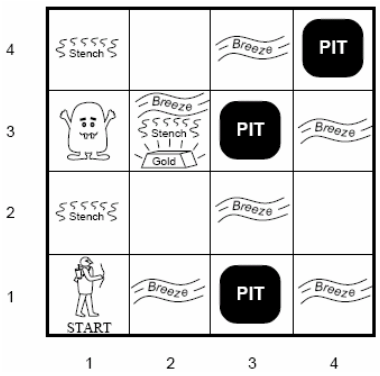
Percept([Stench,Breeze,None,None,None],5): Zum Zeitpunkt t=5 die Wahrnehmung auf dem Startfeld. Percept ist ein 2-stelliges Prädikat mit einer 5-elementigen Liste als erstes Argument. Wahrnehmung erfolgt immer von direkt benachbarten Felder, aber nicht von diagonalen “Nachbarfeldern”.

FOL-Beispiele

Repräsentation der Wumpus Welt

„Nachbarschaftsbeziehung“:

$\forall x,y,a,b\ Adjacent([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y],[x-1,y],[x,y+1],[x,y-1]\}$



(die Listennotation […] ist trivial auf FOL-Terme abbildbar)

(die Repräsentation von Mengen befindet sich im Anhang)

(Annahme: unzulässige Felder werden zurückgewiesen)

statt [a,b] könnte man auch field(a,b) schreiben. Die Nachbarschaftsbeziehung kann mittels FOL viel effizienter dargestellt werden als in der Aussagenlogik, wo für jedes Feld die Nachbarschaftsbeziehung explizit beschrieben werden musste.

Diagnostik: (verborgene) Ursache vom (beobachteten) Effekt schließen; Kausalik: Effekt aus Ursache schließen. Es gilt übrigens auch der umgekehrte Schluss (die Negation): $\forall s\ \neg Breese(s) \Rightarrow \neg\exists r\ Adjacent(r,s) \wedge Pit(r)$, sodass die Implikation eigentlich eine Äquivalenz ist: $\forall s\ Breese(s) \Leftrightarrow \exists r\ Adjacent(r,s) \wedge Pit(r)$

FOL-Beispiele — Versteckte Eigenschaften in der Wumpus-Welt

„Nachbarfelder von Grube sind windig“:

⇨ **“Diagnostische Regeln”**

- … schließen von beobachteten Effekten auf versteckte Ursachen
- $\forall s\ Breese(s) \Rightarrow \exists r\ Adjacent(r, s) \wedge Pit(r)$
- $\forall s\ \neg Breese(s) \Rightarrow \neg\exists r\ Adjacent(r, s) \wedge Pit(r)$

⇨ **“Kausale” Regeln**

- … schließen von (versteckten) Eigenschaften auf Wahrnehmungen
- $\forall r\ Pit(r) \Rightarrow (\forall s\ Adjacent(r, s) \Rightarrow Breeze(s))$
- $\forall s\ (\forall r\ Adjacent(r, s) \Rightarrow \neg Pit(r)) \Rightarrow \neg Breeze(s)$

statt [a,b] könnte man auch field(a,b) schreiben. Die Nachbarschaftsbeziehung kann mittels FOL viel effizienter dargestellt werden als in der Aussagenlogik, wo für jedes Feld die Nachbarschaftsbeziehung explizit beschrieben werden musste.

Diagnostik: (verborgene) Ursache (bzw. Zustand) aus (beobachteten) Effekt (bzw. Eigenschaft) folgern. Beispielsweise in der medizinischen Diagnostik von Symptomen auf Krankheiten schließen.

Kausalität: Effekt (bzw. Eigenschaft) aus Ursache (bzw. Zustand) schließen. Weil die 2. Implikation die Negation der ersten ist handelt es sich eigentlich um eine Äquivalenz: $\forall s\ Breese(s) \Leftrightarrow \exists r\ Adjacent(r, s) \wedge Pit(r)$

Kausalität: Ein Loch verursacht Wind in allen umgebenden Feldern. Bzw.: Wenn alle Nachbarfelder eines Feldes keine Löcher sind, dann ist das Feld ohne Wind.

Gliederung

- **Einleitung**
- **Prädikatenlogik**
  - Formalismus
  - Beispiele
- **Inferenzen in der Prädikatenlogik**
  - Propositionale Inferenz
  - Unifikation
  - Resolution
- **Zusammenfassung**

FOL mit propositionaler Inferenz

Idee

1. **Überführung einer FOL-Wissensbasis in eine Wissensbasis mit Aussagenlogik**
2. **Danach Anwendung eines aussagenlogischen Inferenzmechanismus**
   - → z.B. Model Checking oder Resolutionsmethode

FOL mit propositionaler Inferenz — Allquantoren auflösen

- **Inferenzregel für Allquantoren: Ersetze jede allquantifizierte Variable mit allen Ground-Termen der Wissensbasis:**
- **$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x) wird ersetzt durch:**

  King(**J**) $\wedge$ *Greedy*(**J**) $\Rightarrow$ *Evil*(**J**)
  King(**R**) $\wedge$ *Greedy*(**R**) $\Rightarrow$ *Evil*(**R**)
  King(**Brother(R,J)**) $\wedge$ *Greedy*(**Brother(R,J)**) $\Rightarrow$ *Evil*(**Brother(R,J)**)
  King(**Brother(J,R)**) $\wedge$ *Greedy*(**Brother(J,R)**) $\Rightarrow$ *Evil*(**Brother(J,R)**)
  usw.

Mit p k-stelligen Prädikaten und n Konstanten kommt man zu $p \cdot n^k$ Instantiierungen!

FOL mit propositionaler Inferenz — Existenzquantoren auflösen

- Inferenzregel für Existenzquantoren: Ersetze jede existenz-quantifizierte Variable mit einem neuen Konstantensymbol:
- $\exists x$ Crown(x) $\wedge$ OnHead(x, J) wird ersetzt durch:

  Crown(C1) $\wedge$ OnHead(C1, J).

- Die resultierende Wissensbasis WB' ist nicht äquivalent zur ursprünglichen WB, aber erfüllbar gdw. WB erfüllbar.

Die durch Auflösung von Existenzquantoren entstehende Wissensbasis ist erfüllbar genau dann, wenn auch die originale Wissensbasis (mit Quantoren) erfüllbar ist. C1 ist eine so genannten Skolem-Konstante (Vorgang der Skolemisierung, s.u.)

FOL mit propositionaler Inferenz — Propositionalisierung

⇨ **Alle Atome der Wissensbasis werden als Propositionen betrachtet.**

Die Wissensbasis

$\text{King}(J) \wedge \text{Greedy}(J) \Rightarrow \text{Evil}(J)$.

$\text{King}(\text{Brother}(R,J)) \wedge \text{Greedy}(\text{Brother}(R,J)) \Rightarrow \text{Evil}(\text{Brother}(R,J))$.

enthält die 6 Propositionen:

{ King(J), Greedy(J), Evil(J), King(Brother(R,J)),
Greedy(Brother(R,J)), Evil(Brother(R,J)) }

Prinzipiell lässt sich jede FOL-Wissensbasis in eine inferentiell äquivalente propositionale WB transformieren.

Probleme der Propositionalisierung: Funktionssymbole führen zu unendlich vielen Ground-Terms.

Gliederung

- **Einleitung**
- **Prädikatenlogik**
  - Formalismus
  - Beispiele
- **Inferenzen in der Prädikatenlogik**
  - Propositionale Inferenz
  - Unifikation
  - Resolution
- **Zusammenfassung**

Unifikation — Idee

⇨ **Aus der Wissensbasis**

$\text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$.

$\text{King}(\text{John})$.

$\text{Greedy}(\text{John})$.

kann Evil(John) direkt, ohne Generierung unnötiger Propositionen, abgeleitet werden, …

⇨ **… durch Substitution von Variablen.**

Idee der Unifikation: Ziel eines Widerspruchsbeweises wie bei der Resolution. Ergebnis ist Widerspruch/kein Widerspruch und eine Variablenbindung im Falle eines Widerspruchs.

Unifikation — Substitution

⇨ **Eine Substitution θ ist eine endliche Menge der Form $\{v_1/t_1, \ldots, v_n/t_n\}$, wobei jedes v_i eine Variable ist und jedes t_i ein Term, der verschieden ist von v_i und die Variablen $v_1, \ldots, v_n$ disjunkt sind.**

→ Jedes Element v_i/t_i stellt eine Bindung von v_i dar.

⇨ **Sei $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ eine Substitution und E ein Ausdruck. Dann ist Eθ der Ausdruck, der gewonnen wird aus E, indem man simultan jedes Vorkommen der Variablen v_i in E durch den Term t_i ersetzt.**

Ein *Ausdruck* ist ein Term, ein Literal, eine Konjuktion von Literalen oder eine Disjunktion von Literalen.

Beispiel:

θ = {y/Substitution, x/Ist(Eine), z/Einfache, w/Zulässige}; E = dies(x,z) ∧ w ⇒ y;

Eθ = dies(Ist(Eine), Einfache) ∧ Zulässige ⇒ Substitution

Unifikation — Unifikator

⇨ Sei S eine endliche Menge von einfachen Ausdrücken.
Eine Substitution θ ist ein **Unifikator** für S, wenn Sθ eine einelementige Menge ist.

Beispiel:

⇨ S = { a, P(b), P(Q(c)) } | S | = 3

⇨ θ = { a / P(Q(z)), b / Q(z), c / z }

⇨ Sθ = { P(Q(z)), P(Q(z)), P(Q(z)) }

= { P(Q(z)) } | Sθ | = 1

Beispiel: S = {a,p(b), p(q(c)),d}; θ = {a/p(q(z)), b/q(z), d/z}; Sθ = {p(q(z))}

Die Reihenfolge ist nur der Übersicht halber so gewählt, in Mengendarstellungen spielt sie natürlich keine Rolle. Der Unifikator sorgt dafür, dass alle Elemente der S-Menge identisch werden, sodass die S-Menge nur aus einem Element besteht (| Sθ | = 1).

Unifikation — Beispiel

p	q	θ
Knows(John, x)	Knows(John, Jane)	{ x/Jane }
Knows(John, x)	Knows(y, OJ)	
Knows(John, x)	Knows(y, Mom(y))	
Knows(John, x)	Knows(x, OJ)	

⇨ Unify(α ,β) = θ wenn $\alpha\theta = \beta\theta$

Unifikation

Beispiel

p	q	θ
Knows(John, x)	Knows(John, Jane)	{ x/Jane }
Knows(John, x)	Knows(y, OJ)	{ x/OJ, y/John }
Knows(John, x)	Knows(y, Mom(y))	
Knows(John, x)	Knows(x, OJ)	

Unifikation — Beispiel

p	q	θ
Knows(John, x)	Knows(John, Jane)	{ x/Jane }
Knows(John, x)	Knows(y, OJ)	{ x/OJ, y/John }
Knows(John, x)	Knows(y, Mom(y))	{ y/John, x/Mom(John) }
Knows(John, x)	Knows(x, OJ)	

Unifikation — Beispiel

p	q	θ
Knows(John, x)	Knows(John,Jane)	{ x/Jane }
Knows(John, x)	Knows(y, OJ)	{ x/OJ, y/John }
Knows(John, x)	Knows(y, Mom(y))	{ y/John, x/Mom(John) }
Knows(John, x)	Knows(x, OJ)	{ FAIL }

FAIL, weil nicht zugleich x durch OJ und durch John substituiert werden kann.

Unifikation — MGU

- **2 mögliche Substitutionen für unify(*Knows(John,x), Knows(y,z)*) :**
  - **(1) α = {y/John, x/z } und**
  - **(2) α = {y/John, x/John, z/John}.**
- **Unterschied: Unifier (1) ist genereller als (2).**
- **θ ist Most General Unifier (MGU), wenn für jeden Unifikator ρ von S eine Substitution γ existiert sodass ρ = θγ.**
  - → Für jede Unifikation existiert genau ein MGU

Ein Unifikator sub von L heißt allgemeinster Unifikator von L, falls für jeden Unifikator sub0 von L gilt, dass es eine Substitution s gibt mit sub0 = sub s. D.h., jeder andere Unifyer ist eine Spezialisierung des MGU.

Wenn es zwei MGUs gibt, dann sind diese Umbenennungen voneinander (bis auf Variablennamen identisch).

Im Beispiel ist der MGU = { y/John, x/z }

Unifikationsalgorithmus

Eingabe: eine Literalmenge L ≠ Ø
θ := { }; (leere Substitution)
while | Lθ | > 1 do
Suche in Lθ die erste Position, an der sich zwei Literale L_1 und L_2 unterscheiden
if keiner der beiden Terme ist eine Variable
then stoppe mit „nicht unifizierbar“
else Sei x die Variable in L_1 und t der Term in L_2
if x kommt in t vor
then stoppe mit „nicht unifizierbar“
else θ := θ ∪ { x/t }
Ausgabe: θ ist MGU

t (Term im anderen Literal) kann natürlich auch eine Variable sein.

Voraussetzungen für Unifizierbarkeit:

a) Gleiche Prädikatensymbole in den Literalen

b) Unifizierbarkeit zweier Terme erfordert, dass mindestens einer von ihnen eine Variable enthält, und dass diese Variable nicht im anderen Term vorkommt.

„ x kommt in t vor“: Bei der Unifikation einer Variablen v mit einem Term τ darf v nicht in τ vorkommen (engl. occurs check). Dieser Test ist wichtig, weil sonst inkorrekte Antworten generiert werden können: Aus der Formel *equal*(*X*, *X*) könnte z.B. die Formel *equal*(*N*, *s*(*N*)) abgeleitet werden. Durch Binden von x an f(x) entstünde eine zyklische Struktur.

Dieser occurs check macht aus dem ansonsten linear aufwändigen Algorithmus einen quadratisch aufwändigen (relativ zur Länge der Ausdrücke). Wird aus Effizienzgründen auf den O.-C. verzichtet, kann die Korrektheit dieser die Unifikation nutzenden Resolutionsmethode nicht mehr garantiert werden. Prolog-Interpreter z.B. führen keinen Occur-Check aus.

Unifikationsalgorithmus — Terminierung und Korrektheit

- In jedem Schritt wird entweder durch Substitution die Menge der Variablen in Lθ kleiner, oder es wird ein Paar nicht unifizierbarer Literale gefunden und das Verfahren bricht ab.
- Die while-Schleife wird nur dann erfolgreich verlassen, wenn $|L\theta| = 1$, d.h. wenn ein Unifikator gefunden ist.
- Das heißt, der Algorithmus terminiert und ist korrekt.

Unifikationsalgorithmus — Beispiel

0. $L = \{Q(x, f(A)), Q(B, y)\}$ $\theta = \{\}$ $|L\theta| = 2$

1. $L_1 = Q(x, f(A))$ $L_2 = Q(B, y)$
   $t_1 = x$ $t_2 = B$ (unterschiedliche Terme)
   $\theta = \{x/B\}$ (Substitution)

2. $L_1\theta = Q(B, f(A))$ $L_2\theta = Q(B, y)$ $|L\theta| = 2$
   $t_1 = f(A)$ $t_2 = y$
   $\theta = \{x/B, y/f(A)\}$

3. $L_1\theta = Q(B, f(A))$ $L_2\theta = Q(B, f(A))$ $|L\theta| = 1$

Dieses und ein komplexeres Beispiel (auf Folie #22) gefunden in http://www.informatik.uni-hamburg.de/WSV/f1/2001/VL-PDF/F1-12PL-Resolution.pdf

0. ist die Initialisierung, 1-3 sind die Schleifendurchläufe.

Gliederung

- **Einleitung**
- **Prädikatenlogik**
  - Formalismus
  - Beispiele
- **Inferenzen in der Prädikatenlogik**
  - Propositionale Inferenz
  - Unifikation
  - Resolution
- **Zusammenfassung**

Resolution, revisited

- **Robinson'sche Resolutionsregel der Aussagenlogik:**
  - Seien $L_1, \ldots, L_n$ und $L^*_1, \ldots, L^*_m$ Literale mit $L = L_i = \neg L^*_j$ für ein i mit $1 \leq i \leq n$ und ein j mit $1 \leq j \leq m$. Dann gilt:

$$(L_1 \vee \ldots \vee L_i \vee \ldots \vee L_n) \wedge (L^*_1 \vee \ldots \vee \neg L_i \vee \ldots \vee L^*_n)$$

$$\models$$

$$L_1 \vee \ldots \vee L_{i-1} \vee L_{i+1} \vee \ldots \vee L_n \vee L^*_1 \vee \ldots \vee L^*_{i-1} \vee L^*_{i+1} \vee \ldots \vee L^*_n$$

- **Eine Klauselmenge ist unerfüllbar genau dann, wenn die leere Klausel abgeleitet werden kann.**

Ausdrücke müssen in konkunktiver Normalform sein, damit die Resolutionsmethode angewendet werden kann.

FOL-Resolution

Idee

- **Anwendung des Resolutionsverfahrens für die Prädikatenlogik**
  - Umgang mit Quantoren
  - Umgang mit Variablen
- **Ansatz**
  1. Bildung der Konjunktiven Normalform
  2. Erweiterung der aussagenlogischen Resolution um die Behandlung von Variablen durch Unifikation

- **KNF besteht aus einer Konjunktion von Disjunktionen von Literalen.**
  - Variablen in Literalen werden als allquantifiziert angenommen

- **Alle Ausdrücke der Prädikatenlogik können in eine inferenziell äquivalente KNF transformiert werden**
  - Hauptproblem: Existenzquantoren eliminieren
  - Das Verfahren zur KNF-Transformation besteht aus 5 Schritten…

⇨ **“Jeder, der alle Tiere liebt, wird von irgendjemandem geliebt”:**

➙ $\forall x\,[\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

➙ y bezeichnet links und rechts der Implikation unterschiedliche Variablen!

1. Implikationen eliminieren

$$\forall x\,[\forall y\ \neg Animal(y) \vee Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$$

$$\Leftrightarrow$$

$$\forall x\,[\neg \forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$$

Zuerst die innere (linke), danach die äußere (rechte) Implikation auflösen.

→ $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2. Negierte Quantoren auflosen:

$\forall x\ [\exists y\ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ Loves(y,x)]$

$\Leftrightarrow \forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

$\Leftrightarrow \forall x\ [\exists y\ \ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

$\neg\forall x\ p \equiv \exists x\ \neg p$	$\neg\exists x\ p \equiv \forall x\ \neg p$

Den negierten Allquantor durch Existenzquantor ersetzen, danach die doppelte Negation des geklammerten Ausdrucks entfernen.

→ $\forall x\,[\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

3. **Variablen standardisieren**

→ $\forall x\,[\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists \mathbf{z}\ Loves(\mathbf{z},x)]$

4. **Existenzquantifizierte Variablen skolemisieren durch Skolem-funktion über alle beeinflussenden allquantifizierten Variablen**

→ $\forall x\,[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

Variablen standardisieren: Voneinander unabhängige Variablen unterschiedlich benennen. Im Beispiel: wird das y im letzten existenzquantifizierten Ausdruck durch ein neues z ersetzt.

Skolemisieren: Ersetzen einer durch einen Existenzquantor gebundene Variable durch eine Funktion. Eine existenzquantifizierte Variable wird durch eine Funktion aller beeinflussenden allquantorgebundenen Variablen ersetzt (entsprechend dem Geltungsbereich der Quantoren).

$\forall$ x $\exists$ y klingone (x) $\wedge$ bekämpft (x, y) → $\forall$ x klingone (x) $\wedge$ bekämpft (x, f(x))

→ $\forall x\,[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

5. Universalquantoren entfernen und KNF-Konnektoren bilden

$[Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee Loves(G(x),x)$

$\Leftrightarrow$

$[Animal(F(x)) \vee Loves(G(x),x)]$

$\wedge\,[\neg Loves(x,F(x)) \vee Loves(G(x),x)]$

Alle Variablen werden im Folgenden als allquantifiziert betrachtet. Daher können alle Allquantoren ohne Verlust der Semantik entfernt werden.

Danach die Regel $(A \wedge B) \vee C \Leftrightarrow (A \vee C) \wedge (B \vee C)$ anwenden und fertig ist die KNF.

- Es ist ein Verbrechen für einen Amerikaner, Waffen an feindliche Staaten zu verkaufen. Nono besitzt Raketen, die es sämtlich von Colonel West kaufte. Raketen sind Waffen. West ist Amerikaner. Nono ist Feind Amerikas.

- Frage:
  (Wie) kann bewiesen werden, dass West ein Verbrecher ist?

- $\forall x,y,z$ American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x, y, z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x).
- $\exists x$ Owns(Nono, x) $\wedge$ Missile(x)
- Missile(x) $\wedge$ Owns(Nono, x) $\Rightarrow$ Sells(West, x, Nono)
- Missile(x) $\Rightarrow$ Weapon(x)
- American(West)
- Enemy(Nono, America)
- Enemy(x, America) $\Rightarrow$ Hostile(x) (Hilfsimplikation)

FOL-Resolution — Beispiel in KNF

Wissensbasis in KNF:

⇨ $\neg$American(x) $\vee$ $\neg$Weapon(y) $\vee$ $\neg$Sells(x, y, z)
$\vee$ $\neg$Hostile(z) $\vee$ Criminal(x).
$\neg$Missile(x) $\vee$ $\neg$Owns(Nono, x) $\vee$ Sells(West, x, Nono).
$\neg$Enemy(x, America) $\vee$ Hostile(x).
$\neg$Missile(x) $\vee$ Weapon(x).
Owns(Nono, M). Missile(M).
American(West). Enemy(Nono, America).

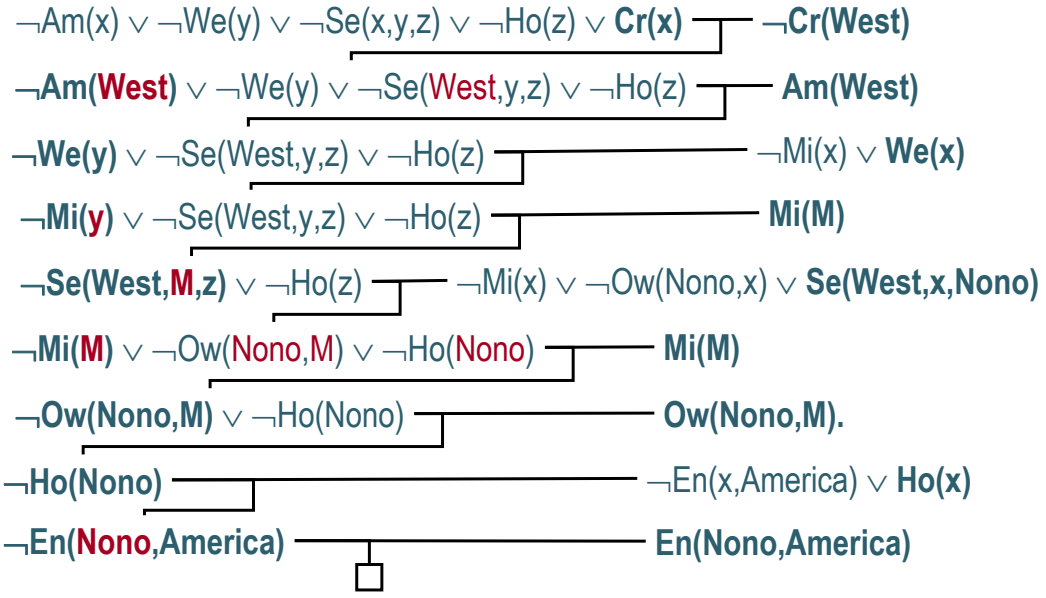
Anfrage negiert in WB aufnehmen:

⇨ $\neg$Criminal(West).

$\forall$x,y,z American(x) $\wedge$ Weapon(y) $\wedge$ Sells(x, y, z) $\wedge$ Hostile(z) $\Rightarrow$ Criminal(x). wurde, da alle Variablen allquantifiziert, einfach in KNF durch Entfernen des Allquantors und Auflösen der Implikation gebracht.

$\exists$x Owns(Nono, x) $\wedge$ Missile(x) wird durch Skolemisierung zu Owns(Nono, M). und Missile(M). M ist eine Konstante (argumentlose Funktion), da keine abhängigen (umgebenden) allquantifizierten Variablen zur berücksichtigen sind.

Resolutionsbeweis in FOL verwendet Unifikation



Am = American, We = Weapon, Se = Sells, Ho = Hostile, Cr = Criminal, Mi = Missile, Ow = Owns, En = Enemy.

Die komplementären Literale, die zur Anwendung der Resolution führen, sind **fett** gedruckt.

In roter Schrift stehen die von der Unifikation betroffenen Variablen.

Dieser Resolutionsbeweis ist durch Anwendung von Heuristiken effizient. Beispielsweise wird gleich Antwortprädikat ins Spiel gebracht und möglichst immer Resolutionen angewendet, die den untersuchten Ausdruck verkleinern (und nicht vergrößern, jedoch ist dies nicht immer möglich, wie insbesondere Schritt 5 belegt). Es existieren viele andere Wege, die umständlicher sind und dennoch zum Ziel kommen – klassische Suchverfahren wie Tiefen- oder Breitensuche kommen hier typischerweise zum Einsatz.

Gliederung

- ⇨ **Einleitung**
- ⇨ **Prädikatenlogik**
  - → Formalismus
  - → Beispiele
- ⇨ **Inferenzen in der Prädikatenlogik**
  - → Propositionale Inferenz
  - → Unifikation
  - → Resolution
- ⇨ **Zusammenfassung**

Zusammenfassung

- ⇨ **FOL besitzt stärkere Ausdruckskraft als Aussagenlogik**
  - → Objekte und Relationen als semantikbehaftete Primitive
  - → Hinreichend mächtig, um die Wumpus Welt zu beschreiben
- ⇨ **Unifikation macht logische Ausdrücke durch Substitution einander ähnlich**
  - → alle FOL-Inferenzverfahren nutzen Unifikation
- ⇨ **Konstruktion der KNF**
  - → insbesondere Skolemisierung existenzquantifizierter Variablen
- ⇨ **Resolution ist konstruktive Beweisführung**

Resolutionsbeweis zeigt die Schritte und ist also nachvollziehbar. Resolution ist Widerspruchsbeweis. Findet:

a) Widerspruch ja/nein

b) Variablenbindung im Falle eines Widerspruchs. Ergebnis enthält Variablenbelegungen (aber nur soweit notwendig (wegen jeweiliger Verwendung des MGU)).

Ausblick

- **Inferenzen mittels Forward Chaining und Backward Chaining, wie in der VL Planen besprochen, sind in FOL ebenso möglich.**
- **Komplexitätsproblem: Kombinatorische Explosion beim Durchsuchen des Resolutionsgraphen**
  - Zumeist sind nur wenige Resolventen zur Ableitung der leeren Klausel nötig
  - Heuristiken zur Reihenfolgebestimmung
  - Resolutionsrestriktionen (z.B. SLD-Resolution in PROLOG)
  - Weitere Heuristiken zur effizienteren Resolution in Russell/Norvig

Heuristiken: gleich mit der zu beweisenden (bzw. zu falsifizierenden) Anfrage beginnen (siehe Resolutionsbeispiel). Siehe Russell/Norvig für weitere Heuristiken, z.B. Stützmengenstrategie (Set of support).

DAI-Labor
TU Berlin

Grundlagen der Künstlichen Intelligenz

Logikbasierte Agenten 2: FOL

24.11.2005

nächster Termin: Nichtmonotones Schließen

Stefan Fricke
stefan.fricke@dai-labor.de

AIOIT
Agententechnologien in betrieblichen Anwendungen und der Telekommunikation

Anhänge

FOL-Beispiele

Verwandschaftsbeziehungen

- **Brüder sind Geschwister**
  $\forall x,y\ Brother(x, y) \Leftrightarrow Sibling(x, y)$
- **Geschwisterschaft ist symmetrisch**
  $\forall x,y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$
- **Eine Mutter ist ein weibliches Elternteil**
  $\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m, c))$
- **Ein Onkel (o) ist der Bruder (b) eines Elternteils (p)**
  $\forall x,o\ Uncle(o, x) \Leftrightarrow \exists p\ Parent(p, x) \wedge Brother(o, p)$

FOL-Beispiele

Mengen

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x|s_2\})$ (Konstruktion)
2. $\neg\exists x, s\ \{x|s\} = \{\}$ (Konsistenz)
3. $\forall x, s\ x \in s \Leftrightarrow s = \{x|s\}$ (Elemente)
4. $\forall x, s\ x \in s \Leftrightarrow \exists y, s_2\ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))$ (Elemente)
5. $\forall s_1, s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$ (Teilmenge)
6. $\forall s_1, s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$ (Gleichheit)
7. $\forall x, s_1, s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$ (Schnittmenge)
8. $\forall x, s_1, s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$ (Vereinigung)

(1) baut Mengen konstruktiv aus der leeren Menge und Elementen x auf.
(2) ist Konsistenzbedingung (Konstante „Leermenge“ enthält keine Elemente)
(3) Element-Definition
(4) konstruktiver Aufbau der Element-Relation
(5) Teilmengenrelation
(6) Gleichheit zwischen Mengen
(7) Schnittmengenrelation
(8) Vereinigungsmengenrelation

Unifikationsalgorithmus

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
  inputs: x, a variable, constant, list, or compound
          y, a variable, constant, list, or compound
          θ, the substitution built up so far

  if θ = failure then return failure
  else if x = y then return θ
  else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
  else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
  else if COMPOUND?(x) and COMPOUND?(y) then
      return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
  else if LIST?(x) and LIST?(y) then
      return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
  else return failure
```

Unifikationsalgorithmus

function UNIFY-VAR(var, x, θ) **returns** a substitution
  inputs: var, a variable
    x, any expression
    θ, the substitution built up so far

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Forward chaining algorithm

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

repeat until *new* is empty
  new $\leftarrow \{\}$
  for each sentence r **in** KB **do**
    $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
    for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
        for some $p'_1, \ldots, p'_n$ in KB
      $q' \leftarrow$ SUBST(θ, q)
      if q' is not a renaming of a sentence already in KB or *new* **then do**
        add q' to *new*
        $\phi \leftarrow$ UNIFY(q', α)
        if ϕ is not *fail* **then return** ϕ
  add *new* to KB
return *false*

Backward chaining algorithm

function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    goals, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\}$
  local variables: *ans*, a set of substitutions, initially empty

  if *goals* is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST(*goals*))
  for each r **in** KB where STANDARDIZE-APART(r) = $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
    ans $\leftarrow$ FOL-BC-ASK(KB, $[p_1, \ldots, p_n | \text{REST}(goals)]$, COMPOSE($\theta, \theta'$)) $\cup$ *ans*
  return *ans*

SUBST(COMPOSE(α_1, α_2), p) = SUBST(α_2, SUBST(α_1, p))