Universität Bielefeld

6 Logik und Deduktion

16. Vorlesung: Skolemisierung Rückwärtsverkettung und Goal Trees

Methoden der Künstlichen Intelligenz

Ipke Wachsmuth WS 1999/2000

Universität Bielefeld

Übersicht

- Skolemisierung – allgemeines Verfahren, um PL-Formeln für maschinelle Deduktion vorzubereiten
- Rückwärtsverkettung – zielorientiertes Inferenzverfahren (in Verbindung mit Goal-tree-Suche und Unifikation)
- allgemeine Resolutionsregel
- Antwortsubstitution

Universität Bielefeld

Skolemisierung – Motivation

Skolemisierung ist ein Verfahren, um die Existenzquantoren aus prädikatenlogischen Formeln (1. Ordnung) zu eliminieren.

Zusammen mit der Ersetzung von allquantifizierten Variablen x durch Matchvariable ?x erhält man quantorenfreie Formeln (in implicit-quantifier form), wie sie von der Inferenzmaschine verarbeitet werden können.

Grundidee:

Existenzquantifizierte Variable y werden
(i) durch neue Konstantensymbole oder
(ii) durch Funktionsausdrücke der Art (`neue-funktion ?x`)
ersetzt.

Universität Bielefeld

Vorbemerkung

- Vor der Skolemisierung werden die prädikatenlogischen Formeln erst durch Äquivalenzumformungen in *Pränexform* gebracht, d.h. alle Quantoren stehen vorn (dabei wird der "wirkliche Typ eines Quantors" geklärt)
- Durch die *Skolemisierung* werden die Formeln zwar nicht äquivalent, aber erfüllbarkeitsäquivalent umgeformt.
- Zum Nachlesen weiterer Details
  -> Schöning: Logik für Informatiker, S. 64 ff

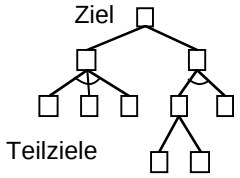
Universität Bielefeld

Rückwärtsverkettung – Idee

Die meisten Inferenzen in einem deduktiven System werden zur Query Time vorgenommen.

- **GATEKEEPER wird eine Implikation `(if p q)` zwar assertieren, aber nichts damit anstellen, bis eine Frage (query) der Form *q'* gestellt wird. Dann wird subquery (subgoal) *p'* gestellt.**
- **Das Stellen von subgoals, sub-subgoals etc. wird durch backward chaining (Rückwärtsverkettung) bewerkstelligt.**
- **Der Basismechanismus des Backward Chaining bezieht sich auf *nichtkonjunktive goals:* Verkettung von Implikationen (rückwärts).**
- **Als Suchstruktur bei *konjunktiven goals* von der Form `(if (and p1 p2) q)` werden Goal Trees eingesetzt.**

Goal Trees (Zielbäume)



Universität Bielefeld

Skolemisierung

(nach dem Logiker Thoraf Skolem)

- allgemeine Technik zur Elimination von Quantoren: liefert eine Darstellung quantifizierter Formeln in DATABASE, die ohne explizite Verwendung von Quantoren auskommt (implicit-quantifier form)

  Beispiele vom letzten Mal (einfache Fälle):

  ```
  (if (inst ?x canary)(color ?x yellow))
  (and (nudist sk-1) (party sk-1 uni-bielefeld))
  ```

- Allquantifizierte Variablen wurden als (auf alle Terme passende) Pattern-matching-Variablen kodiert; die führende existenzquantifizierte in diesem Spezialfall durch eine Konstante **`sk-1`** (genauer: eine nullstellige Skolemfunktion) kodiert.

Universität Bielefeld

Skolemisierung – allgemein

- jede existenzquantifizierte Variable muß in eine Funktion umkodiert werden, deren Argumente diejenigen allquantifizierten Variablen sind, deren Skopi den des Existenzquantors umfassen.

Beispiel: „Jede Person hat einen Kopf.“

```
(forall (x) (if   (inst x person)
                  exists (y) (and (inst y head)
                                  (partof y x))))
```

wird skolemisiert als:

```
              (if   (inst ?x person)
                    (and   (inst (head-of ?x) head)
                           (partof (head-of ?x) ?x)))
```


Universität Bielefeld

Problem: Namenseindeutigkeit!

- **Im Prädikatenkalkül müssen Terme eindeutige Entitäten benennen.**
- **Wenn die Funktion (Skolemfunktion) `head-of` eingeführt wird, muß klar sein, daß sie nirgendwo in anderer Weise verwendet wird.**
- **Deshalb führen Skolemisierungsalgorithmen jeweils brandneue Funktionssymbole der Form `sk-`*n* ein (also statt `head-of` z.B. `sk-17`).**
- **Häufig wird aber auch von Hand skolemisiert; dann wählt man "sprechende Namen".**

Universität Bielefeld

Scheinbarer/wirklicher Quantor

Häufig ist nicht einfach zu erkennen, ob ein *Existenz-* oder ein *Allquantor* zu skolemisieren ist.

Beispiel: "Nichts ist göttlich"

```
(not (exists (x) divine x)))
(forall (x) (not (divine x)))
```

Skolemisierung?

```
(not (divine sk-4))      ????
(not (divine ?x))
```

- hieße ja: „etwas ist nicht göttlich“
- heißt: „alle x sind nicht göttlich“

- **Keines Objektes Existenz wird hier assertiert.**
- **Also "wirklicher" Typ: Allquantor**

Universität Bielefeld

Negation und Quantorentyp

noch einmal: welches ist der "wirkliche" Quantor?

Allgemein hängt der "wirkliche" Quantortyp von der Anzahl der Negationen ab, die darauf Einfluß haben.

Um den wirklichen Typ zu bestimmen:

Zähle die Anzahl der `not`s und der `if`-Antezedenten, worin er vorkommt.

Falls *ungerade:* Tausche den Quantor um (also All- in Ex.- bzw. Ex.- in All-); falls *gerade:* so lassen.

Beispiel: "A car without wheels is not valuable." "Ein Wagen ohne Räder ist nix wert."

```
(forall (c)
   (if (and (inst c car)
            (not (exists (x)(and (inst x wheel)
                                 (attached x c)))))
       (not (valuable c))))
```

wird skolemisiert als:

```
(if (and (inst ?c car)
         (not (and (inst (sk-5 ?c) wheel)
                   (attached (sk-5 ?c) ?c))))
    (not (valuable ?c)))
```

Der Existenz-Quantor tritt innerhalb einer Negation und eines `if`-Antezedenten auf (gerade Anzahl), ist also ein „wirklicher“.

Und noch einmal zum Problem gleich benannter (unabhängiger) Variablen:

```
(forall(y)(if (and(exists(x)(just-left x y))
                  (exists(x)(just-left y x)))
              (crowded y)))
```

als Formalisierung von "Wenn man jemand links von sich und jemand rechts von sich hat, sitzt man eingeklemmt."

Bemerkung:

- **Besonders schwierig ist es, die Negation einer skolemisierten Formel zu bestimmen.**
- **Verfahren:** ***Entskolemisieren, negieren, reskolemisieren.***

```
(forall(y)(if (and(exists(xl)(just-left xl y))
                  (exists(xr)(just-left y xr)))
              (crowded y)))
```

skolemisiert:

```
          (if (and (just-left ?xl ?y))
                   (just-left ?y ?xr)))
              (crowded ?y)))
```


Skolemisierung - allg. Verfahren

1. Bestimme, welche Variablen existenz- und welche allquantifiziert sind (den "wirklichen“ Quantorentyp).
2. Ersetze jede existenzquantifizierte Variable durch eine Skolem-funktion. Die Argumente dieser Funktion sind alle diejenigen allquantifizierten Variablen, in deren Skopus der Existenzquantor liegt.
3. Falls zwei verschiedene allquantifizierte Variablen den gleichen Namen haben, benenne eine davon (brandneu) um.
4. Ersetze schließlich jede allquantifizierte Variable *v* durch eine mit "?" markierte Pattern-matching-Variable *?v* .

   (Die Allquantifizierung bleibt implizit dadurch gegeben, daß eine solche Variable mit allem "matcht".)

Universität Bielefeld

Forward vs. backward chaining

Forward Chaining

Aus p' und (if p q)
inferiere q'

wobei p mit p' unifiziert
mit MGU θ und $q' = q\theta$

- Inferenz zur Assertion Time
- "Die Assertion resolviert mit der Implikation."

Backward Chaining

(if p q) sei assertiert.

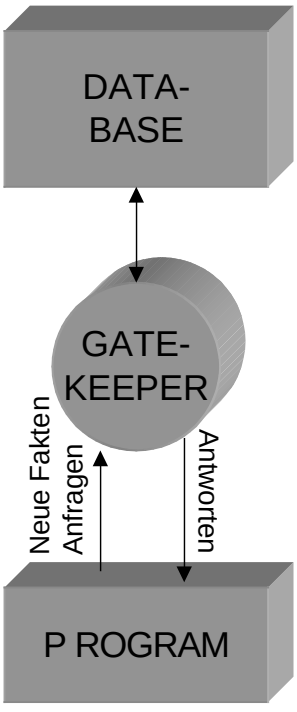
Wenn nach q' gefragt wird (als goal)
und q, q' haben MGU θ
wird $p' = p\theta$ als subgoal aufgeworfen.

- Inferenz zur Query Time
- "Das goal resolviert mit der Implikation."

(Voraussetzung: die Formeln in DATABASE sind standardisiert.)

Universität Bielefeld

Goal-Formeln kennzeichnen:



Verwende (Show:*formula*) zur Kennzeichnung von goal-Formeln.

Hier am Beispiel (Show: (color tweety yellow))
bzw. subgoal (Show: (inst tweety canary))
(bezogen auf früheres Beispiel)

- Show: ist kein logischer Junktor!
  (manchmal Pseudo-Junktor genannt)
- hängt aber zusammen mit dem
  Junktor not (kommt später)

Wenn eine Anfrage (query) an DATABASE gestellt wird, muß die entsprechende Formel gekennzeichnet werden, um sie von den "beliefs" (assertierten Formeln) unterscheiden zu können.

Universität Bielefeld

Goal-Formeln skolemisieren:

Für goals werden die Konventionen für das Skolemisieren umgedreht (entsprechend dem Verfahren bei not):

```
  (Show: (exists (y) (color y yellow)))
```
wird skolemisiert als
```
  (Show: (color ?y yellow))
```

```
  (Show: (forall (y) (color y yellow)))
```
wird skolemisiert als
```
  (Show: (color sk-8 yellow))
```

intuitiv: Wenn es für ein gewisses neues `sk-8` gezeigt werden kann, kann es für beliebige Instanzen gezeigt werden.

Universität Bielefeld

Show: *versus* not

`(Show: (color tweety yellow))` **(goal)**
kann als Versuch verstanden werden,
`(color tweety yellow)` **durch Widerspruch zu beweisen:**

Logisch gesehen ist `Show:` gleichbedeutend mit `not` (die entsprechenden Ausdrücke werden in gleicher Weise skolemisiert)

pragmatisch gesehen entspricht `Show:` einer Annahme (probeweise gemachte Assertion)

`(not (color tweety yellow))`
wird probeweise assertiert - *als Annahme;*

schon vorhanden seien folgende "geglaubte" Assertionen:

`(inst tweety canary)`

`(if (inst ?x canary)(color ?x yellow))`
gleichbedeutend:
`(or (not(inst ?x canary))(color ?x yellow))`

Das negierte goal resolviert mit der Implikation zu
`(not (inst tweety canary))`

Widerspruch! **Also gilt das ursprüngliche goal.**

Universität Bielefeld

Allgemeine Resolutionsregel

mit forward und backward chaining als Spezialfällen:

Schreibe `(if p q)` **als** `(or (not p) q)`

Chaining	Aus	und	inferiere
Forward	`p'`	`(or (not p) q)`	`q'`
Backward	`(not q')`	`(or q (not p))`	`(not p')`
Allgemein	`not m'`	`(or m n)`	`n'`
Forward	`m =(not p)`		`n = q`
Backward	`m = q`		`n=(not p')`

Vergleiche:
6. Vorlesung, bei weiterem Interesse: Abschnitt 6.6 in Charniak/ McDermott

Verallgemeinerung zur allgemeinen Resolutionsregel:

(V) Aus `(or` n_1 `(not m'))` **und** `(or m` n_2`)`
inferiere `(or` n_1' n_2'`)`

(beide Regel-Antezedenten dürfen disjunktiv sein)

Universität Bielefeld

Zentral: Antwortsubstitutionen

Allgemein:

Gegeben das goal `(Show:q')`
mit `(if p q)` **in DATABASE und** `q, q'` **haben MGU** θ .

Produziere subgoal `(Show:p`θ`)`
und falls das eine Antwort ψ **hat, ist** $\theta \cup \psi$ **eine Antwort auf das ursprüngliche goal.**

„theta“ *„psi“*

Beispiel: Assertiert seien

(I) `(inst tweety canary)`

(II) `(if (inst ?x canary)(color ?x yellow))`

`(Show: (color ?y yellow))` **als goal**

resolviert mit (II) zum subgoal
`(Show: (inst ?y canary))` θ = {x = ?y}

dies unifiziert mit (I) zu
`(inst tweety canary)` ψ = {y = tweety}

und damit bestätigt sich das ursprüngliche goal zu
`(color tweety yellow)` $\theta \cup \psi$ = {x = tweety}

D.h. die Frage *"Existiert etwas, das gelb ist?"*
führt zu der Antwort *"Ja,* `tweety` *ist gelb":* `{x = tweety}`

Antwortsubstitution

Universität Bielefeld

Goal Trees für subgoal-Suche

Nichtkonjunktives Goal: **Verkettung von Implikationen (rückwärts)**

Konjunktives Goal: **Finde Antworten für ein Konjunkt, die auch Antworten für die anderen Konjunkte sind.**

Algorithmus: Theorembeweiser ("Deductive Retriever")

---> Goal Trees

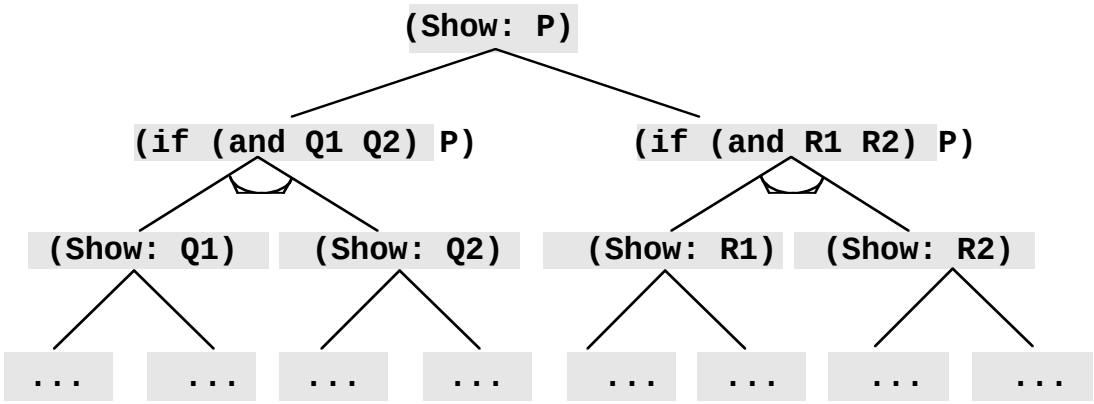
Bild siehe nächste Folie

Die UND-Knoten eines Goal Trees enthalten *constraints:* Randbedingungen an die Lösungen für jede ihrer Komponenten.

Hier sind das die folgenden: Die Variablenbindungen für gleich benannte Variablen in den Teillösungen müssen identisch sein!

Universität Bielefeld

Theorembeweiser-Goal Tree



Universität Bielefeld

Das bekannte Prolog-Beispiel

(vergleiche 6. Vorlesung)

Eine Definition von "Großvater":

Für alle x, für alle y, für alle z gilt:
x ist Großvater von y, wenn
x Vater von z ist und z Vater von y.

oder ganz formal:

$\forall x \forall y \forall z$ `(grossvater x y)`
$\leftarrow$ `(vater x z)(vater z y)`

Die entsprechende Definition "über die Mutter":

$\forall x \forall y \forall z$ `(grossvater x y)`
$\leftarrow$ `(vater x z)(mutter z y)`

In PROLOG gelten alle Variablen als implizit allquantifiziert (d.h. die operationale Semantik erlaubt Ersetzungen durch beliebige, in der Wissensbasis vorhandene Terme); Quantoren werden weggelassen:

```
   (grossvater ?x ?y) 1
← (vater ?x ?z)(vater ?z ?y)
```

```
   (grossvater ?x ?y) 2
← (vater ?x ?z)(mutter ?z ?y)
```

Seien jetzt diese beiden Implikationen (Regeln) assertiert, zusammen mit folgenden Fakten:

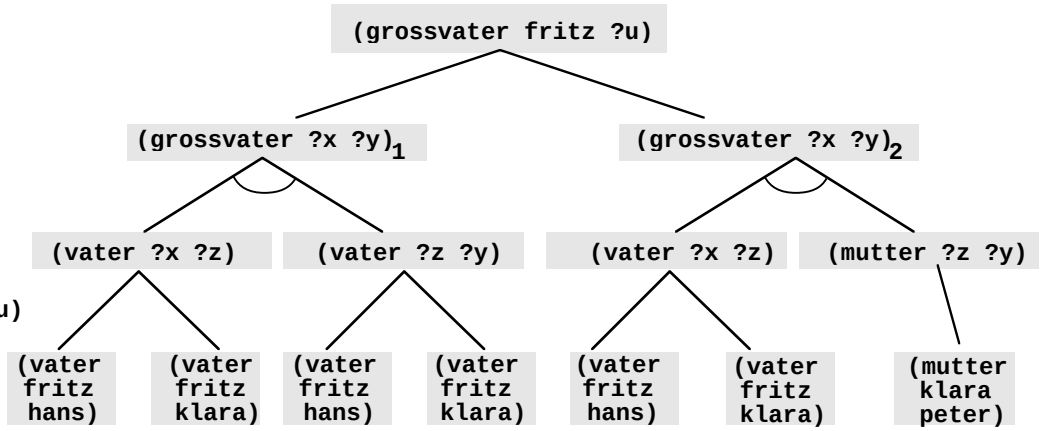
```
(vater fritz hans)
(vater fritz klara)
(mutter klara peter)
```

Ist Fritz ein Großvater?
`(grossvater fritz ?u)`

Universität Bielefeld

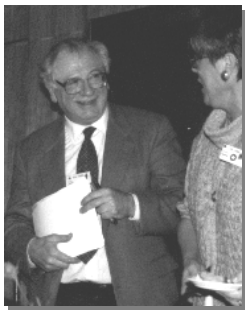
Goal-Tree Suche in PROLOG

```
show:     (grossvater fritz ?u)
show:     (vater fritz ?z)
success:  (vater fritz hans)
show:     (vater hans  ?y)
redo:     (vater hans  ?y)
fail:     (vater hans  ?y)
redo:     (vater fritz ?z)
success:  (vater fritz klara)
show:     (vater klara ?y)
redo:     (vater klara ?y)
fail:     (vater klara ?y)
redo:     (grossvater fritz ?u)
show:     (vater fritz ?z)
success:  (vater fritz hans)
show:     (mutter hans ?y)
fail:     (mutter hans ?y)
redo:     (vater fritz ?z)
success:  (vater fritz klara)
show:     (mutter klara ?y)
success:  (mutter klara peter)
success:  (grossvater fritz peter)
```



Was sind hier die Constraints an den AND-KNOTEN?

Bemerkungen



J.A. Robinson, 1996

ACHTUNG: Viele Algorithmen zum Theorembeweisen sind unvollständig (existierende Beweise werden nicht immer gefunden)!

Versuche, einen Theorembeweiser vollständig(er) zu machen, werden i.a. teuer mit Effizienzverlusten bezahlt.

Es gibt etliche Varianten des Resolutionsverfahrens und speziellen Strategien dafür, die im Gebiet "Automatisches Beweisen" untersucht werden*.

Sie alle gehen auf das allgemeine Resolutionsverfahren ("complete resolution") von ROBINSON (1965) zurück.

———

***Vertiefung: Kap. 2 in Görz, Einführung in die künstliche Intelligenz**

Leseempfehlung heute

- Charniak & McDermott, Kapitel 6, Seite 349-360; 378ff
- Vertiefung: Görz, Kap. 2