Universität Bielefeld

3 Logik und Inferenz

7. Vorlesung: Schlussfolgern im Prädikatenkalkül; Deduktion; Skolemisierung

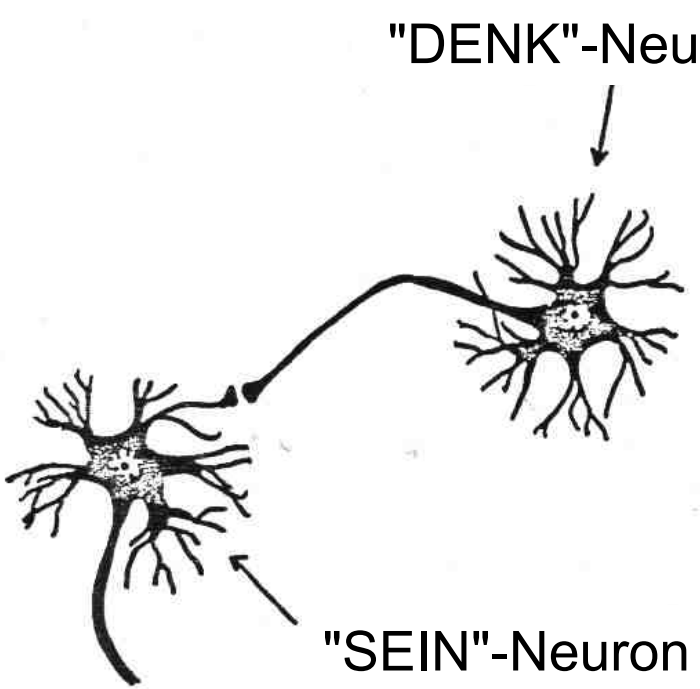
Methoden der Künstlichen Intelligenz

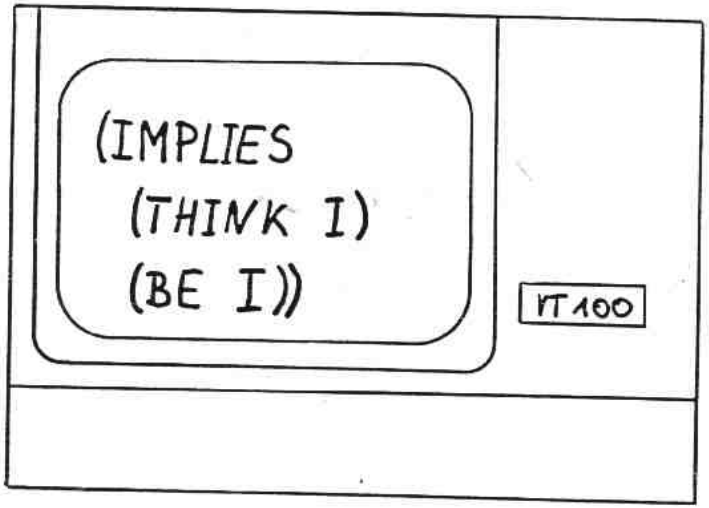
Ipke Wachsmuth WS 2010/2011

Kernfragen der Vorlesung

1. **Wie lässt sich Wissen symbolisch repräsentieren?**
2. **Wie lassen sich Probleme durch (geschickte) Suche lösen?**
3. **Wie lassen sich maschinell Schlussfolgerungen aus Annahmen ziehen?**
4. **Wie lassen sich auch bei unsicherem und unvollständigem Wissen Schlüsse ziehen?**
5. **Wie lassen sich Kommunikationsfähigkeiten für Maschinen realisieren?**

Logisches Schließen





NEIN!

JA!

Wissensrepräsentation

Wissen ist eine abstrakte Qualität.
Es muss an eine symbolische Repräsentation gekoppelt sein, um einsatzfähig zu sein.

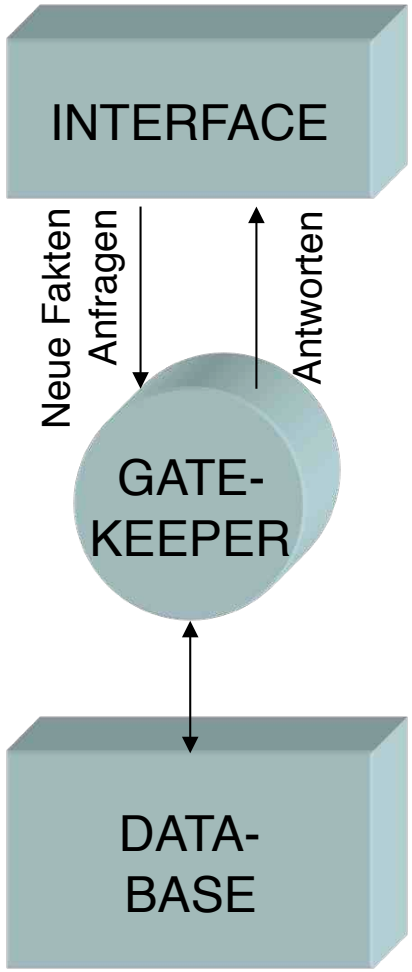
Logikformalismen mit Theorem-beweisern können als eine Repräsentation für einen intelligenten Agenten dienen.

(A. Newell)

Erste Idee:
Benutze eine Logiksprache zur symbolischen Repräsentation

Zweite Idee:
Statte den Agent mit einem Theorem-beweiser zum Schlussfolgern aus

Schlussfolgern („reasoning“)



bezeichnet kognitive Prozesse, mit denen aus vorhandenem Wissen bzw. Annahmen oder Vermutungen neues Wissen bzw. Annahmen oder Vermutungen gewonnen werden (Inferenzen).

***"neu":* jetzt verfügbar, vorher *nicht unmittelbar* verfügbar.**

Deduktive Inferenzen

sind formale Schlussfolgerungsprozesse eines bestimmten Typs (darüber hinaus gibt es abduktive und induktive Inferenzen).

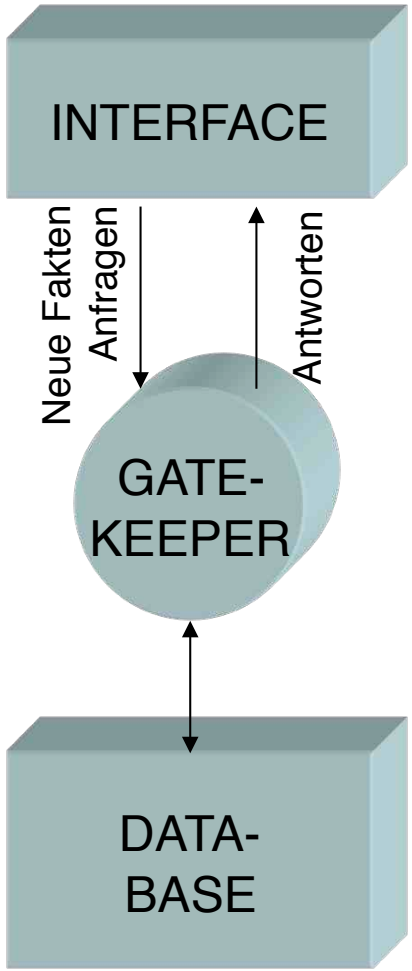
Gatekeeper oder Inferenzmaschine

ist zuständig für das Hinzufügen (oder Löschen) von Assertionen in einer Datenbasis und das Gewinnen von Schlussfolgerungen.

Datenbasis (database)

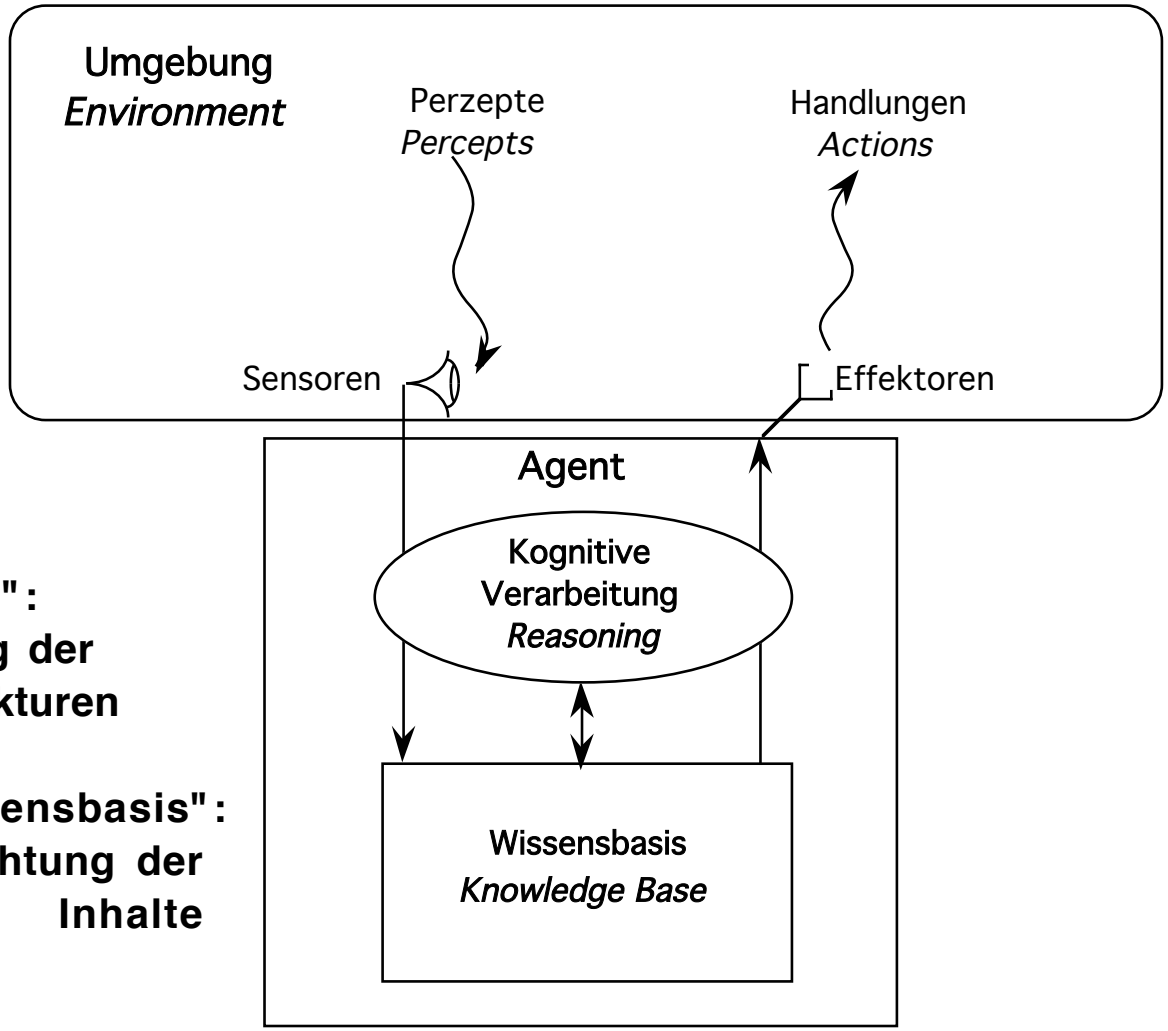
enthält eine organisierte Menge symbolischer Datenstrukturen, die die aktuellen "beliefs" (Überzeugungen, für wahr gehaltene Aussagen) eines Agenten darstellen.

Ziel: Schlussfolgernder Agent



"Datenbasis":
Betrachtung der Symbolstrukturen

"Wissensbasis":
Betrachtung der Inhalte



Zu unterscheidende Ebenen

Optionen:

1) Assertionsbasierte Repräsentation (z.B. Prädikatenlogik)
vs.
2) objektbasierte Repräsentation (z.B. assoziative Netzwerke)

Im weiteren zunächst Fall (1) betrachtet.

- **Fakten- oder Wissensebene**
  bezieht sich auf darzustellende Sachverhalte (eines betrachteten Weltausschnitts)
- **Ebene der abstrakten Repräsentation**
  Darstellung von Sachverhalten in symbolischen Ausdrücken **Fokus: *"Inhalt"***
- **Ebene der konkreten Repräsentation**
  maschinengeeignete und -verarbeitbare Darstellung (Implementierung) von Ausdrücken als Datenstrukturen
  Fokus: *"Index"*

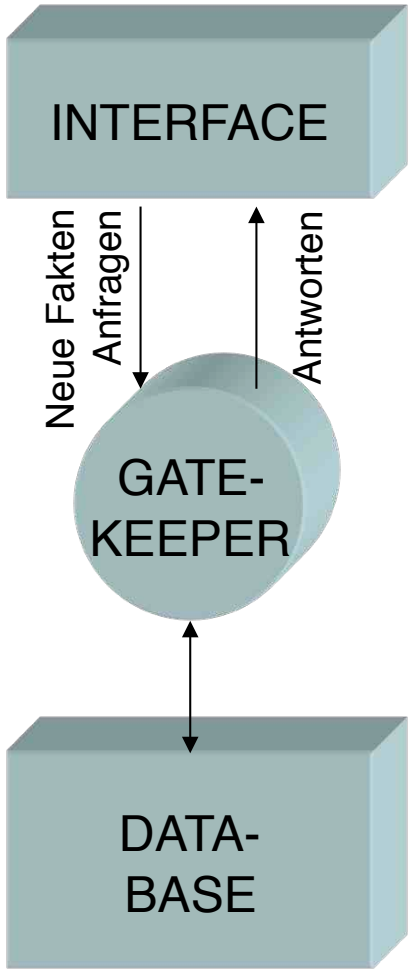
Gebrauch des Prädikatenkalküls

```
(forall (x) (if (in x antarctica)(temperature x cold)))
(forall (x) (if (person x)(exists (y)(head-of x y))))
(forall (t1 t2 t3)(if (and (before t1 t2)(before t2 t3))
                      (before t1 t3)) )
```

<u>Schlüsselidee</u>: Jeder Ausdruck notiert oder "denotiert" etwas.

- **Terme denotieren Individuen oder Klassen von Individuen**
- **Formeln denotieren Sachverhalte (Propositionen)**
- **Die Wahrheit einer Formel kann "weltabhängig" sein, nämlich davon, ob in der betrachteten Welt der denotierte Sachverhalt wahr oder falsch ist.**

Database und Gatekeeper



Einfachstes Design:

DATABASE hat Datenstrukturen in Form von PL-Formeln.

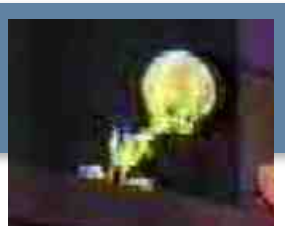
GATEKEEPER kann die gewöhnlichsten Inferenzregeln ausführen.

(Zweite Idee)
Statte den Agent mit einem Theorembeweiser zum Schlussfolgern aus

Basisbefehle zur Benutzung der DATABASE durch GATEKEEPER:

`assert`	**fügt jeweils eine Proposition in DATABASE ein**
`retract`	**zieht eine Proposition aus DATABASE zurück**
`query`	**fügt eine Frage-Formel in DATABASE ein und versucht, darauf passende Antworten zu deduzieren**

Tweety-Beispiel



```
assert:  (forall(x) (if (inst x canary)(color x yellow)))
assert:  (inst tweety canary)
query:   (color tweety yellow)
```

Ist Tweety gelb?

(Die Antwort sollte positiv sein.)

Zwei Möglichkeiten für GATEKEEPER (wann die Inferenz durchzuführen ist):

a) sobald `(inst tweety canary)` assertiert wurde:
assertion-time inference / forward chaining

b) erst wenn `(color tweety yellow)` gefragt wird:
query-time inference / backward chaining

Beide Wege sind gängig, aber jeweils mit unterschiedlichen Auswirkungen.

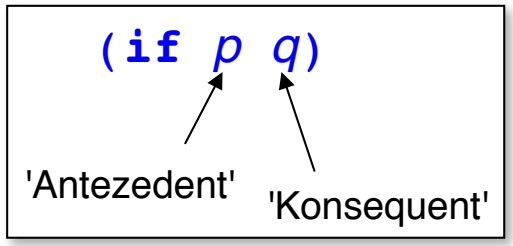
Am INTERFACE sollte man davon nichts merken.

Aber wie ...?

Formeln mit Variablen: 1. Weg

(I) Modus Ponens

Aus `p` **und** `(if p q )` **inferiere** `q` **.**



Keine Inferenz möglich bei:

```
(*) (forall(x) (if (inst x canary)(color x yellow)))
    (inst tweety canary)
```

p'

(Muster passt nicht!)

Betrachte jetzt eine weitere Inferenzregel:

(II) Universelle Einsetzung

Aus `(forall(x) p )` **inferiere** `p` **mit allen Vorkommen der Variable** `x` **durch den gleichen Term eingesetzt.**

Dann folgt mit der Regel (II) aus `(*)` **und** `x` **ersetzt durch** `tweety` **:**

```
(if (inst tweety canary)(color tweety yellow))
```

(if p q)

und mit Modus Ponens (I) : `(color tweety yellow)`

Das Problem mit Regel (II)

BEDENKE: Das Anliegen der traditionellen Logik ist es, den einfachsten Mechanismus zu finden, der alle legitimen Inferenzen berechnet. (Effizienz oder Interessantheit sind dabei nicht von Belang.)

Auf gleiche Weise können mit Regel (II) -zig andere variablenfreie Formeln legitim berechnet werden, die aber nutzlos sind, z.B.:

```
(if (inst bill-22 canary)(color bill-22 yellow))
(if (inst block-1 canary)(color block-1 yellow))
(if (inst block-21 canary)(color block-21 yellow))
(if (inst blue canary)(color blue yellow))
(if (inst box-104 canary)(color box-104 yellow))
(if (inst brick-33 canary)(color brick-33 yellow))
(if (inst brick-47 canary)(color brick-47 yellow))
(if (inst eiffeltower canary)(color eiffeltower yellow))
(if (inst fred canary)(color fred yellow))
(if (inst jack-1 canary)(color jack-1 yellow))
(if (inst mary canary)(color mary yellow))
(if (inst sylvia-21 canary)(color sylvia-21 yellow))
(if (inst tweety canary)(color tweety yellow))
(if (inst yellow canary)(color yellow yellow))
   .......
```

Erste Abhilfe zur Eindämmung

Vorwärtsverkettung (forward chaining)

am Beispiel:

Sowie `(inst` *thing* `canary)` **assertiert wird,**
wird `(color` *thing* `yellow)` **sofort inferiert (und sonst nichts)**

Dazu wird Regel (II) gezielt mit *thing* **auf (*) angewandt und sodann Regel (I) angewandt.**

Ein geschickteres Vorgehen ist die Unifikation;
dazu sind aber erst Vorbereitungen zu treffen...

Quantorenelimination

implicit-quantifier form

1) **Allquantifizierte Formeln werden wie folgt umgeschrieben:**

von: `(forall(x) (if (inst x canary)(color x yellow)))`

zu: `(if (inst ?x canary)(color ?x yellow))`

2) **Existenzquantifizierte Formeln (mit freistehenden Existenzquantor) werden wie folgt umgeschrieben:**

von: `(exists(x) (and (nudist x)(party x uni-bielefeld)))`

zu: `(and (nudist sk-1)(party sk-1 uni-bielefeld))`

Idee: Wenn ein solcher Mr. X existiert, kann man ihn formal benennen.
(Wenn mehrere existieren, ist der Benannte einer davon stellvertretend.)

Skolemisierung

(so benannt nach dem Logiker Thoraf Skolem)

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

Beispiel oben: Konstante **`sk-1`** (genauer: eine nullstellige Skolemfunktion)

Grundidee:

Existenzquantifizierte Variablen werden

(i) durch neue Konstantensymbole oder

(ii) durch Funktionsausdrücke der Art `(neue-funktion ?x)`

ersetzt (oder auch mehrerer Variablen).

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

Vorbemerkungen

- Vor der Skolemisierung werden die prädikatenlogischen Formeln erst durch Äquivalenzumformungen in sog. *Pränexform* gebracht, d.h. alle Quantoren stehen vorn (dabei wird der "wirkliche Typ eines Quantors" geklärt – kommt noch...)
- Durch *Skolemisierung* werden die Formeln zwar nicht äquivalent, aber erfüllbarkeitsäquivalent umgeformt (→ Schöning)
- Namensmehrdeutigkeiten sind zu *bereinigen (Standardisierung)*
- Allquantifizierte Variablen werden als (auf alle Terme passende) Match-Variablen kodiert.

formale Grundlage z.B.:

- Schöning, Logik für Informatiker, Seite 64ff

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)))
```

Wie skolemisieren?

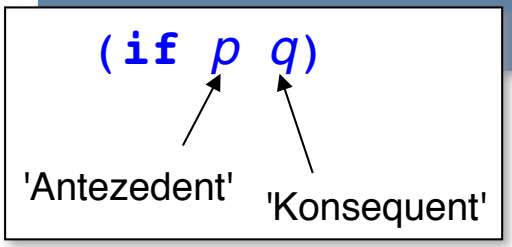
```
(not (divine sk-4))      ????

(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**

Negation und Quantorentyp



noch einmal: welches ist der "wirkliche" Quantortyp?

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 `nots` 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 Auto ohne Räder ist nichts wert."

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

(2) *p* — (1)

q

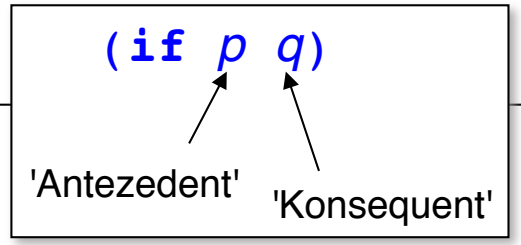
wird skolemisiert als:

```
(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“.

Skolemisierung – allgemein

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



Beispiel: „Jede Person hat einen Kopf.“

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

wird skolemisiert als:

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

eine Matchvariable

eine (1-stellige) Skolemfunktion

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`).**

- **Zuweilen wird aber auch von Hand skolemisiert; dann wählt man "sprechende Namen".**

Bereinigung von Variablennamen

Beispiel: `(on ?x table)`

`(if (on big-bertha ?x)(collapses ?x)`

Kann man `(collapses table)` folgern?

Standardisierung („bereinigte Form“):

In einer der Formeln `x` z.B. durch `y` ersetzen.

So jedenfalls nicht,

da einmal `x = big-bertha`

und einmal `x = table` substituiert werden müsste,

was nicht gleichzeitig möglich ist.

Jedoch: Die Formeln repräsentieren voneinander unabhängige Ausdrücke, die *zufällig* gleich benannte Variablen haben.

Ausweg: Wenn gleiche Variablen in zwei verschiedenen Formeln auftreten, benenne alle Variablen der einen Formel „brandneu“.

Noch einmal: Bereinigte 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."

bereinigt:

```
(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)))
```

Bemerkung:

- **Nicht ganz einfach ist es, die Negation einer skolemisierten Formel zu bestimmen.**
- **Verfahren: *Entskolemisieren, negieren, reskolemisieren.***

Quantorenelimination: Allgemeines Verfahren

Vorbereitung für maschinelle Formelverarbeitung

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

   (Die Allquantifizierung bleibt implizit dadurch gegeben, dass eine solche Variable mit allen Termen "matcht".)

Vorschau nächste Vorlesung

- Unifikation; Vorwärts- und Rückwärtsverkettung (chaining)
- Rückwärtsverkettung – zielorientiertes Inferenzverfahren in Verbindung mit Goal-tree-Suche und Unifikation
- allgemeine Resolutionsregel
- Antwortsubstitution

Leseempfehlung heute:

- Charniak & McDermott, Kapitel 6.1 und 6.3 z.Tl.

und natürlich:

- Kurztext zu Teil 3 (digitaler Semesterapp.)

alternativer Text:

- Russell & Norvig, Kapitel 9, Seite 265 ff