

---

## Semantische Sprachverarbeitung

---

**Carola Eschenbach**  
**Universität Hamburg, FB Informatik**  
**AB Wissens- und Sprachverarbeitung (WSV)**

**Sommersemester 2007**

---

---

## Semantische Sprachverarbeitung

---

### Sitzung 3

- Produktivität von Bedeutung
  - Techniken der Bedeutungserfassung
    - Logik : Semantik
- 

---

## Formale Darstellung von Bedeutungen

### Prädikatenlogik

- Formeln repräsentieren Wahrheitsbedingungen
- Prädikate, Funktions- und Relationssymbole repräsentieren Eigenschaften, Abbildungen und Relationen (die Inhalte)
- Konstanten repräsentieren Objekte
- Logische Symbole (Junktoren, Quantoren, Variablen) bieten Inventar für die Verknüpfung der Inhalte

### Lambda-Kalkül

- Lambda-Abstraktion erlaubt die Bildung von komplexen Prädikaten, Funktions- und Relationsausdrücken.
- Mächtiges Werkzeug für die Darstellung der Kombinationsregeln



---

## AN IMPORTANT JOB

### **This is a story about four people named Everybody, Somebody, Anybody and Nobody.**

There was an important job to be done and Everybody was sure that Somebody would do it.

Anybody could have done it, but Nobody did it.

Somebody got angry about that, because it was Everybody's job.

Everybody thought Anybody could do it, but Nobody realised that Everybody wouldn't do it.

It ended up that Everybody blamed Somebody when Nobody did what Anybody could have.

### Systematische Variationen (1)

<b>NP-Variation</b> [NP Peter] [VP lacht]. [NP Ein Junge] [VP lacht]. [NP Jeder Junge] [VP lacht]. [NP Kein Junge] [VP lacht].	<b>VP-Konjunktion</b> [NP Peter] [VP lacht und winkt]. [NP Ein Junge] [VP lacht und winkt]. [NP Jeder Junge] [VP lacht und winkt]. [NP Kein Junge] [VP lacht und winkt].
<b>Satz-Konjunktion</b> [NP Peter] [VP lacht] und [NP Peter] [VP winkt]. [NP Ein Junge] [VP lacht] und [NP ein Junge] [VP winkt]. [NP Jeder Junge] [VP lacht] und [NP jeder Junge] [VP winkt]. [NP Kein Junge] [VP lacht] und [NP kein Junge] [VP winkt].	

### Systematische Variationen (2)

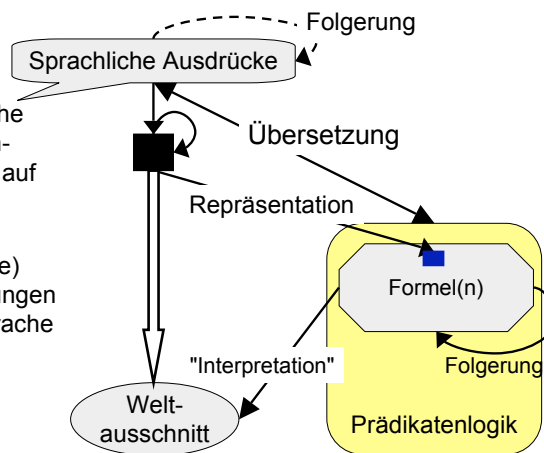
<b>NP-Variation</b> [NP Mausi] [VP meselt]. [NP Ein Kosi] [VP meselt]. [NP Jeder Kosi] [VP meselt]. [NP Kein Kosi] [VP meselt].	<b>VP-Konjunktion</b> [NP Mausi] [VP meselt und diest]. [NP Ein Kosi] [VP meselt und diest]. [NP Jeder Kosi] [VP meselt und diest]. [NP Kein Kosi] [VP meselt und diest].
<b>Satz-Konjunktion</b> [NP Mausi] [VP meselt] und [NP Mausi] [VP diest]. [NP Ein Kosi] [VP meselt] und [NP ein Kosi] [VP diest]. [NP Jeder Kosi] [VP meselt] und [NP jeder Kosi] [VP diest]. [NP Kein Kosi] [VP meselt] und [NP kein Kosi] [VP diest].	

Einige Aspekte der Satz-Bedeutung sind von der Bedeutung der Inhaltswörtern unabhängig

### Formale Linguistik

#### Modellierung von Bedeutung

- durch systematische Abbildung natürlicher Sätze auf prädikatenlogische Formeln,
- so dass (empirische) Folgerungsbeziehungen der natürlichen Sprache als nachweisbare Folgerungen in der Logik resultieren



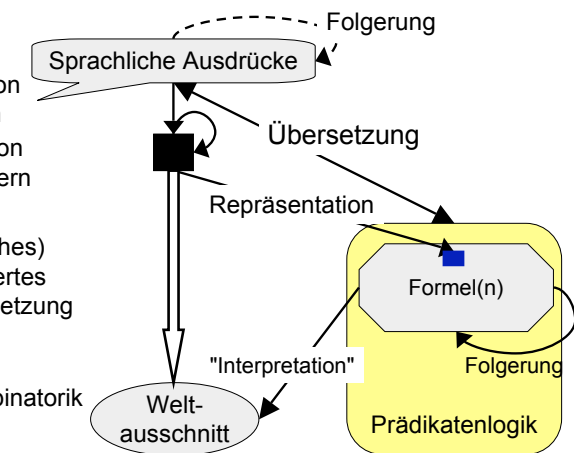
### Formale Linguistik

#### Logik dient

- zur Repräsentation der Bedeutungen
- zur Kommunikation zwischen Forschern

#### Ziel

- (möglichst einfaches) prinzipiengesteuertes Modell der Übersetzung
- Aufdeckung der Systematik der Bedeutungskombinatorik



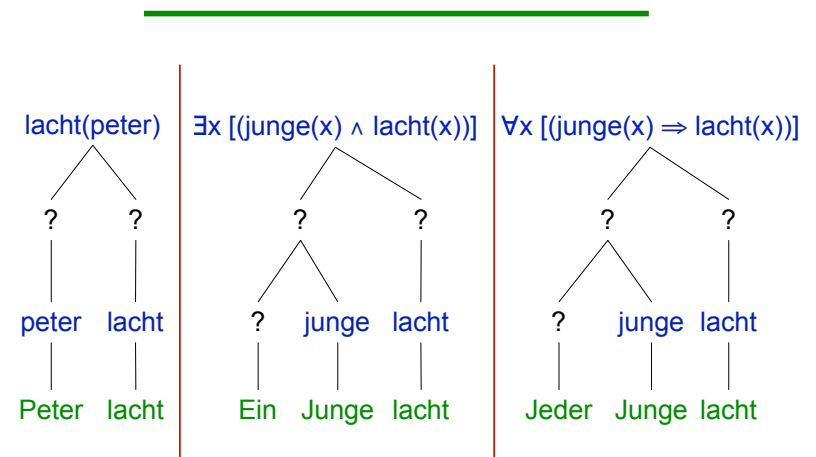
## Systematische Variationen

NP-Variation	VP-Konjunktion
[ <sub>NP</sub> Peter] [ <sub>VP</sub> lacht]. lacht(peter)	[ <sub>NP</sub> Peter] [ <sub>VP</sub> lacht und winkt]. (lacht(peter) ∧ winkt(peter))
[ <sub>NP</sub> Ein Junge] [ <sub>VP</sub> lacht]. ∃x [(junge(x) ∧ lacht(x))]	[ <sub>NP</sub> Ein Junge] [ <sub>VP</sub> lacht und winkt]. ∃x [(junge(x) ∧ (lacht(x) ∧ winkt(x)))]
[ <sub>NP</sub> Jeder Junge] [ <sub>VP</sub> lacht]. ∀x [(junge(x) ⇒ lacht(x))]	[ <sub>NP</sub> Jeder Junge] [ <sub>VP</sub> lacht und winkt]. ∀x [(junge(x) ⇒ (lacht(x) ∧ winkt(x)))]
[ <sub>NP</sub> Kein Junge] [ <sub>VP</sub> lacht]. ¬∃x [(junge(x) ∧ lacht(x))]	[ <sub>NP</sub> Kein Junge] [ <sub>VP</sub> lacht und winkt]. ¬∃x [(junge(x) ∧ (lacht(x) ∧ winkt(x)))]
Satz-Konjunktion	
[ <sub>NP</sub> Peter] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> Peter] [ <sub>VP</sub> winkt]. (lacht(peter) ∧ winkt(peter))	
[ <sub>NP</sub> Ein Junge] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> ein Junge] [ <sub>VP</sub> winkt]. ∃x [(junge(x) ∧ lacht(x)) ∧ ∃y [(junge(y) ∧ winkt(y))]]	
[ <sub>NP</sub> Jeder Junge] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> jeder Junge] [ <sub>VP</sub> winkt]. ∀x [(junge(x) ⇒ lacht(x)) ∧ ∀y [(junge(y) ⇒ winkt(y))]]	
[ <sub>NP</sub> Kein Junge] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> kein Junge] [ <sub>VP</sub> winkt]. ¬∃x [(junge(x) ∧ lacht(x)) ∧ ¬∃y [(junge(y) ∧ winkt(y))]]	

C. Eschenbach: Semantische Sprachverarbeitung

3 – 9

## Beispiel



C. Eschenbach: Semantische Sprachverarbeitung

3 – 10

## Prädikatenlogische Repräsentation: Inventar

### Nicht-logisches Inventar

- (freie, verfügbare Symbole)
- Konstanten: peter, ...
- Prädikate (einstellig): junge, lacht, winkt, ...

### Logisches Inventar

- (feste, logische Symbole)
- Junktoren, einstellig: ¬ (Negation)  
zweistellig: ∧ (Konjunktion), ∨ (Disjunktion), ⇒ (Implikation)
- Quantoren: ∀ (All-Quantor), ∃ (Existenzquantor)
- Variablen: x, y, z, ...

### Hilfssymbole

- Klammern, Komma

C. Eschenbach: Semantische Sprachverarbeitung

3 – 11

## Prädikatenlogische Repräsentation: formale Sprache

### Terme

- Konstanten
- Variablen
- (Kombinationen aus Funktionssymbolen und Termen)

### Formeln

- Kombinationen aus einem Prädikat und einem Term:  
lacht(peter), winkt(x), junge(y)
- Kombinationen aus n-stelligen Relationssymbolen und n Termen: lacht(laura, peter)
- Negation einer Formel: ¬lacht(peter)
- Kombination von zwei Formeln mit einem zweistelligen Junktoren: (winkt(peter) ∧ ¬lacht(peter))
- Kombination aus Quantor, Variable und Formel: ∃x [lacht(x)]

C. Eschenbach: Semantische Sprachverarbeitung

3 – 12

### Prädikatenlogik in PROLOG

- Formeln und Terme werden durch PROLOG-Terme repräsentiert
- Variablen
  - PROLOG-Variablen

#### Kein Symbol-Zeichensatz in PROLOG

- Junktoren  
&, v, ->, ~ anstelle von  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ ,  $\neg$
- definiert als (Infix-)Operatoren
- Quantoren  
exists(X, F), forall(Y, F) anstelle von  $\exists x [F]$ ,  $\forall y [F]$
- freie Symbole: Deklaration von Konstanten und Relationen  
constant(laura). relation(junge,1).

### Prädikatenlogik

#### Logische Eigenschaften und Relationen

- gelten in allen Interpretationen
- sind unabhängig von der Interpretation der freien Symbole
- Tautologie, gültige Formel
  - wahr unter jeder Interpretation  
z.B.  $(\text{lacht}(\text{peter}) \vee \neg \text{lacht}(\text{peter}))$
- Kontradiktion, widersprüchliche Formel
  - falsch unter jeder Interpretation  
z.B.  $(\text{lacht}(\text{peter}) \wedge \neg \text{lacht}(\text{peter}))$
- Folgerung: aus F folgt G, wenn unter jeder Interpretation, in der F wahr ist, auch G wahr ist
  - z.B.  $(\text{lacht}(\text{peter}) \wedge \text{winkt}(\text{peter})) \models \text{lacht}(\text{peter})$

### Logische Eigenschaften und Relationen

#### gelten unter allen Interpretationen

- in der Prädikatenlogik gibt es unendlich viele Interpretationen

#### können durch Beweisverfahren festgestellt werden

- z.B. Resolution ( $\rightarrow$  F1-Vorlesung)
- z.B. Tableau-Beweiser ( $\rightarrow$  LOS-Vorlesung)

#### aber: Prädikatenlogik (PL) ist nur Semi-Entscheidbar

- für jedes Beweisverfahren gibt es eine Aufgabe, die es nicht (in endlicher Zeit) bearbeiten kann
- man muss damit rechnen, kein Ergebnis zu erhalten

#### Bestimmte Fragmente der PL sind entscheidbar

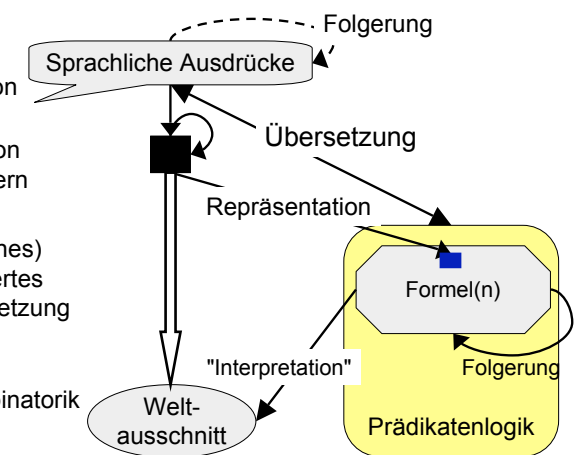
### Formale Linguistik

#### Logik dient

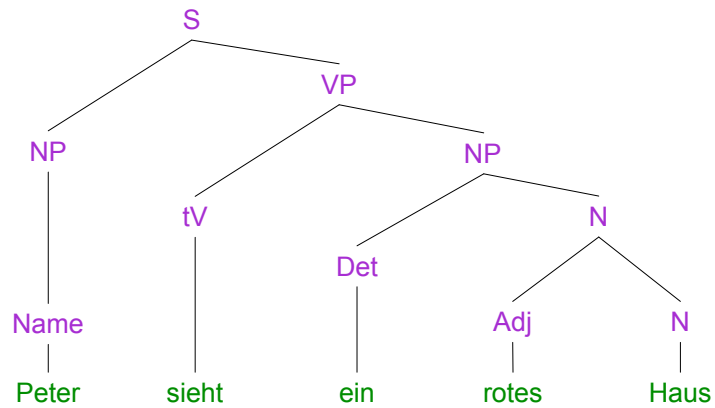
- zur Repräsentation der Bedeutungen
- zur Kommunikation zwischen Forschern

#### Ziel

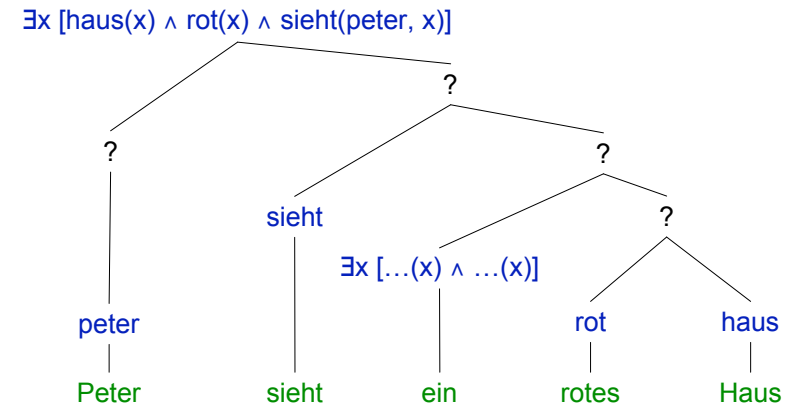
- (möglichst einfaches) prinzipiengesteuertes Modell der Übersetzung
- Aufdeckung der Systematik der Bedeutungskombinatorik



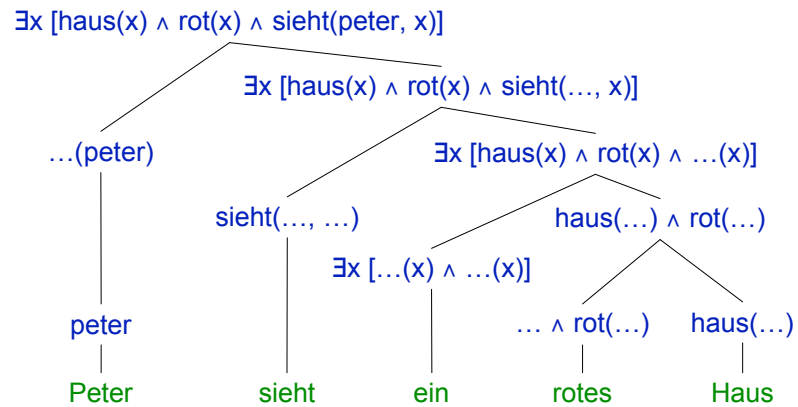
## Peter sieht ein rotes Haus.



## Wahrheitsbedingungen: Peter sieht ein rotes Haus.



## Wahrheitsbedingungen: Peter sieht ein rotes Haus.



## Korrespondenz Syntax - Semantik

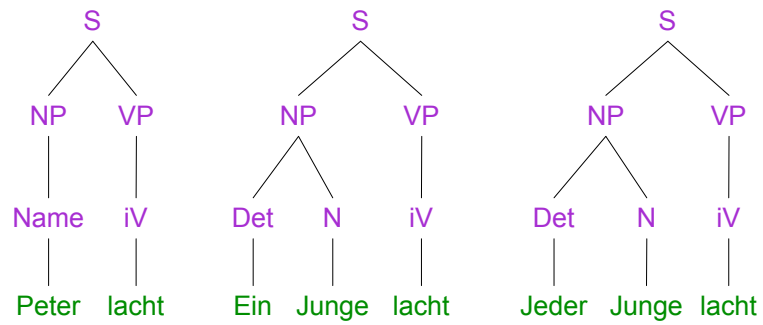
### Syntax

- Ausdrücke (Wörter, Konstituenten, Phrasen) gehören **syntaktischen Kategorien** an, die ihren grammatischen Positionsmöglichkeiten entsprechen.
  - Verb (intransitiv, transitiv, di-transitiv, ...), Verbalphrase
  - Nomen, Nominalphrase, Artikel, Determinator, Pronomen

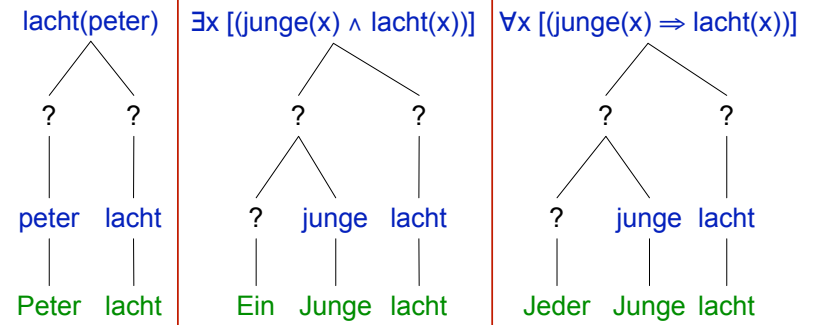
### Semantik

- Die Bedeutungen der Ausdrücke haben unterschiedliche **semantische Typen**, die ihrer Funktion im Bedeutungsaufbau entsprechen
  - Proposition, Menge / Eigenschaft, Relation, Quantor, ...

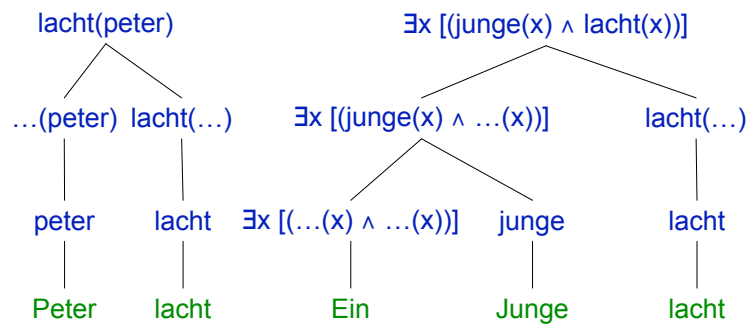
### Beispiel: Syntaktische Strukturen



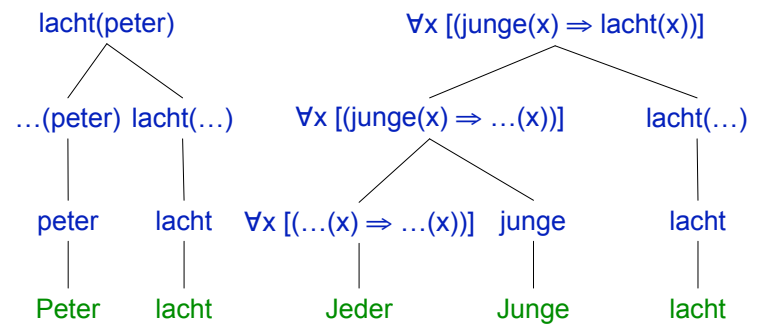
### Beispiel



### Beispiel



### Beispiel



## Grundidee

### die Bedeutung eines Ausdrucks

- beinhaltet einen eigentlichen, inhaltlichen Beitrag
- weist (wohldefinierte) Lücken auf, die der Kontext spezifiziert
- Beitrag und Lücken werden zu prädikatenlogischen Fragmenten zusammengesetzt

### der Typ des Ausdrucks

- determiniert, welche Lücken vom Kontext zu füllen sind

### Regeln

- determinieren, wie die Lücken gefüllt werden

### PROLOG-Umsetzung

- Lücken und Prädikatenlogik-Variablen werden durch Prolog-Variablen repräsentiert

## Konjunktionen (F2)

### verknüpfen zwei Sätze zu einem Satz

#### Lexikon

$\text{conj}(\text{conj}/(-\text{und}), [F, G]/(F \& G)) \rightarrow \text{lexem}(\text{und}).$   
 $\text{conj}(\text{conj}/(-\text{oder}), [F, G]/(F \vee G)) \rightarrow \text{lexem}(\text{oder}).$

#### Regeln

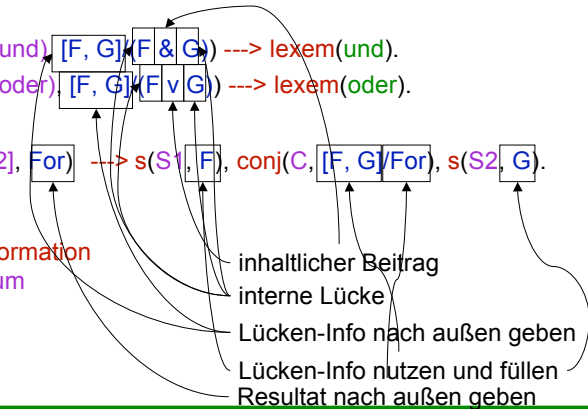
$s(s/[S1, C, S2], \text{For}) \rightarrow s(S1, F), \text{conj}(C, [F, G]/\text{For}), s(S2, G).$

Parser-Information

Syntaxbaum

Semantik

Wortform



## Konjunktionen (F2)

### verknüpfen zwei Sätze zu einem Satz

#### Lexikon

$\text{conj}(\text{conj}/(-\text{und}), [F, G]/(F \& G)) \rightarrow \text{lexem}(\text{und}).$   
 $\text{conj}(\text{conj}/(-\text{oder}), [F, G]/(F \vee G)) \rightarrow \text{lexem}(\text{oder}).$

#### Regeln

$s(s/[S1, C, S2], \text{For}) \rightarrow s(S1, F), \text{conj}(C, [F, G]/\text{For}), s(S2, G).$

$\text{vp}(\text{vp}/[\text{VP1}, C, \text{VP2}], [X]/\text{For}) \rightarrow \text{vp}(\text{VP1}, [X]/F),$   
 $\text{conj}(C, [F, G]/\text{For}), \text{vp}(\text{VP2}, [X]/G).$

Die Regel für Verbalphrasenkonjunktion unterscheidet sich nur in der Behandlung von Lücken.

## Nomen und Adjektive (F1, F2)

### Nomen drücken Eigenschaften aus

### Adjektive modifizieren Eigenschaften

#### Lexikon

$n(n/(-\text{haus}), [X]/(\text{haus}(X))) \rightarrow \text{lexem}(\text{haus}).$   
 $n(n/(-\text{junge}), [X]/(\text{junge}(X))) \rightarrow \text{lexem}(\text{junge}).$   
 $\text{adj}(\text{adj}/(-\text{rotes}), [X, F]/(\text{rot}(X) \& F)) \rightarrow \text{lexem}(\text{rotes}).$

#### Regel

$n(n/[\text{Adj}, N], [X]/\text{For}) \rightarrow \text{adj}(\text{Adj}, [X, F]/\text{For}),$   
 $n(N, [X]/F).$

## Nominalphrasen (F1, F2)

Determinatoren verknüpfen Eigenschaften von Nomen und VPs

### Lexikon

det(det/ (-ein), [X, F, G]/exists(X, F & G)) ---> lexem(ein).

det(det/ (-jeder), [X, F, G]/forall(X, F -> G)) ---> lexem(jeder).

det(det/ (-kein), [X, F, G]/(~ exists(X, F & G))) ---> lexem(kein).

pn(pn/ (-peter), peter) ---> lexem(peter).

pn(pn/ (-laura), laura) ---> lexem(laura).

### Regel

np(np/[Det, N], [X, G]/For) ---> det(Det, [X, F, G]/For), n(N,[X]/F).

np(np/[PN], [Name, For]/For) ---> pn(PN, Name).

## Verbalphrasen und Sätze (F1, F2)

### Lexikon

iv(iv/ (-lacht), [X]/lacht(X)) ---> lexem(lacht).

iv(iv/ (-winkt), [X]/winkt(X)) ---> lexem(winkt).

tv(tv/ (-sieht), [X, Y]/sieht(X, Y)) ---> lexem(sieht).

### Regeln

vp(vp/[IV], [X]/For) ---> iv(IV, [X]/For).

vp(vp/[TV,NP], [X]/For) ---> tv(TV, [X, Y]/G),  
np(NP, [Y, G]/For).

s(s/[NP,VP], For) ---> np(NP, [X, G]/For), vp(VP, [X]/G).

## Grammatik mit Syntaxbaum und Semantik (F2)

s(s/[NP,VP], For) ---> np(NP, [X, G]/For), vp(VP, [X]/G).

np(np/[PN], [Name, For]/For) ---> pn(PN, Name).

np(np/[Det, N], [X, G]/For) ---> det(Det, [X, F, G]/For), n(N,[X]/F).

n(n/[Adj, N], [X]/For) ---> adj(Adj, [X, F]/For), n(N, [X]/F).

vp(vp/[IV], [X]/For) ---> iv(IV, [X]/For).

vp(vp/[TV,NP], [X]/For) ---> tv(TV, [X, Y]/G),  
np(NP, [Y, G]/For).

vp(vp/[VP1,C,VP2], [X]/For) ---> vp(VP1, [X]/F),  
conj(C, [F, G]/For), vp(VP2,[X]/G).

s(s/[S1,C,S2], For) ---> s(S1, F), conj(C, [F, G]/For),  
s(S2, G).

## Lexikon mit Syntax und Semantik (F2)

n(n/ (-haus), [X]/(haus(X))) ---> lexem(haus).

n(n/ (-junge), [X]/(junge(X))) ---> lexem(junge).

adj(adj/ (-rotes), [X, F]/(rot(X) & F)) ---> lexem(rotes).

det(det/ (-ein), [X, F, G]/exists(X, F & G)) ---> lexem(ein).

det(det/ (-jeder), [X, F, G]/forall(X, F -> G)) ---> lexem(jeder).

det(det/ (-kein), [X, F, G]/(~ exists(X, F & G))) ---> lexem(kein).

pn(pn/ (-peter), peter) ---> lexem(peter).

pn(pn/ (-laura), laura) ---> lexem(laura).

iv(iv/ (-lacht), [X]/lacht(X)) ---> lexem(lacht).

iv(iv/ (-winkt), [X]/winkt(X)) ---> lexem(winkt).

tv(tv/ (-sieht), [X, Y]/sieht(X, Y)) ---> lexem(sieht).

conj(conj/ (-und), [F, G]/(F & G)) ---> lexem(und).

conj(conj/ (-oder), [F, G]/(F v G)) ---> lexem(oder).

## Welche Formeln werden erzeugt? (Auswahl)

---

<b>NP-Variation</b> [ <sub>NP</sub> Peter] [ <sub>VP</sub> lacht]. [ <sub>NP</sub> Ein Junge] [ <sub>VP</sub> lacht]. [ <sub>NP</sub> Jeder Junge] [ <sub>VP</sub> lacht]. [ <sub>NP</sub> Kein Junge] [ <sub>VP</sub> lacht].	<b>VP-Konjunktion</b> [ <sub>NP</sub> Peter] [ <sub>VP</sub> lacht und winkt]. [ <sub>NP</sub> Ein Junge] [ <sub>VP</sub> lacht und winkt]. [ <sub>NP</sub> Jeder Junge] [ <sub>VP</sub> lacht und winkt]. [ <sub>NP</sub> Kein Junge] [ <sub>VP</sub> lacht und winkt].
<b>Satz-Konjunktion</b> [ <sub>NP</sub> Peter] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> Peter] [ <sub>VP</sub> winkt]. [ <sub>NP</sub> Ein Junge] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> ein Junge] [ <sub>VP</sub> winkt]. [ <sub>NP</sub> Jeder Junge] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> jeder Junge] [ <sub>VP</sub> winkt]. [ <sub>NP</sub> Kein Junge] [ <sub>VP</sub> lacht] und [ <sub>NP</sub> kein Junge] [ <sub>VP</sub> winkt].	

Welche Aussagen/Formeln sind in welchen Modellen wahr ?

---

## Ausprobieren ?

---

### Paket 02PL

#### sem\_frag2de.pl

- Grammatik mit Erzeugung und Ausgabe der Syntaxbäume und prädikatenlogischen Formeln (Semantik)
- 

## Theoretischer Hintergrund

---

### Prolog-Umsetzung

- aber was wird hier umgesetzt ?
- was ist die Theorie dahinter ?
- wie ist das Beispiel zu verallgemeinern ?

---

## Typen und Lambda-Kalkül

---

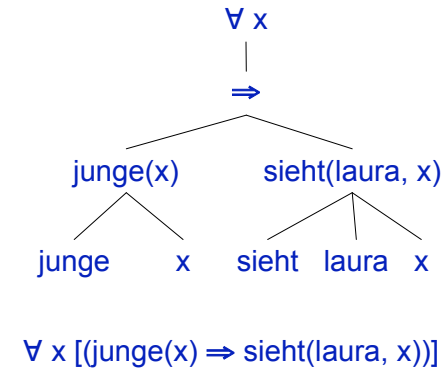
### Nächster Foliensatz

---

## Exkurs

### Modelltheoretische Semantik der Prädikatenlogik

## Darstellung der hierarchischen Struktur von Formeln



## Prädikatenlogische Repräsentation: Interpretation

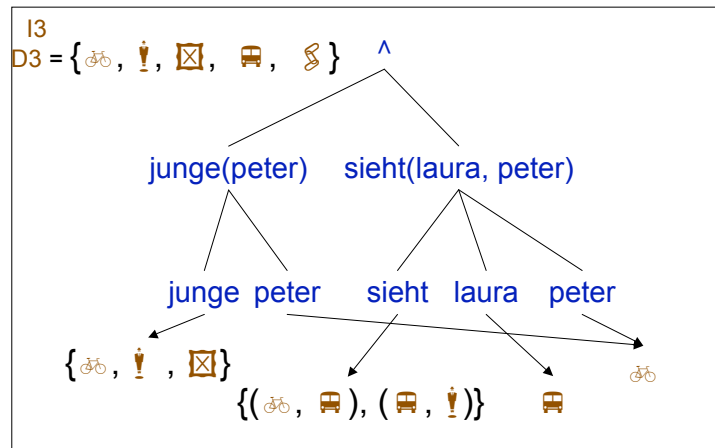
### Eine Interpretation einer prädikatenlogischen Sprache

- ist gegeben durch eine Abbildung der **frei verfügbaren Symbole** auf passende Entitäten
  - Konstanten  $\rightarrow$  Objekte
  - Prädikate  $\rightarrow$  Mengen von Objekten
  - Relationssymbole  $\rightarrow$  Mengen von Objekt-Tupeln
  - Funktionssymbole  $\rightarrow$  Funktionen
- determiniert **eindeutig**, welchen Wert ein komplexer geschlossener Ausdruck erhält.
  - prädikatenlogische Grammatik ist eindeutig
  - Interpretation der logischen Symbole ist fest vorgegeben

## Zwei Interpretationen

Domäne						
junge	✓		✓			✓
lacht	✓			✓	✓	✓
winkt		✓		✓		✓
peter	✓					
Domäne						
junge			✓			
lacht				✓		
winkt				✓		
peter				✓		

### Interpretation 3: freie Symbole



### Interpretation zusammengesetzter Ausdrücke

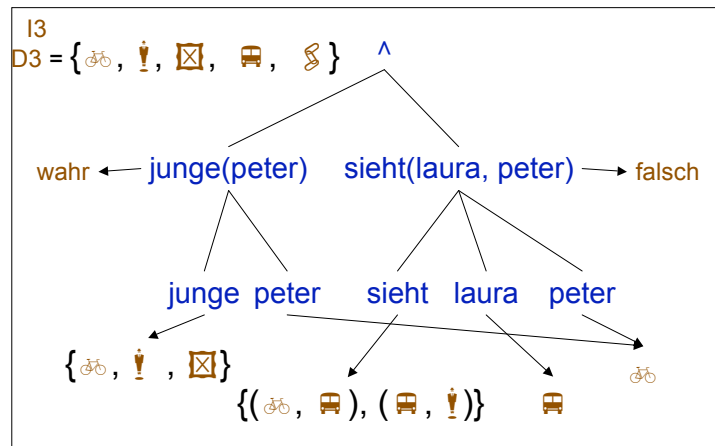
Sei  $D$  eine Menge und  $I$  eine Interpretation in  $D$  für die Symbole  $\text{peter}$ ,  $\text{laura}$ ,  $\text{lacht}$ ,  $\text{winkt}$ ,  $\text{junge}$ ,  $\text{haus}$ ,  $\text{sieht}$

- da  $\text{peter}$ ,  $\text{laura}$  Konstanten sind:  $I(\text{peter}), I(\text{laura}) \in D$
- da  $\text{lacht}$ ,  $\text{winkt}$ ,  $\text{junge}$  Prädikate sind:  $I(\text{lacht}), I(\text{winkt}), I(\text{junge}), I(\text{haus}) \subseteq D$
- da  $\text{sieht}$  ein zweistelliges Relationssymbol ist:  $I(\text{sieht}) \subseteq D^2$

Interpretation von atomare Formeln

- $I(\text{junge}(\text{peter})) = \text{wahr}$ , wenn  $I(\text{peter}) \in I(\text{junge})$ , sonst falsch
- $I(\text{lacht}(\text{peter})) = \text{wahr}$ , wenn  $I(\text{peter}) \in I(\text{lacht})$ , sonst falsch
- $I(\text{winkt}(\text{laura})) = \text{wahr}$ , wenn  $I(\text{laura}) \in I(\text{winkt})$ , sonst falsch
- $I(\text{sieht}(\text{laura}, \text{peter})) = \text{wahr}$ , wenn  $(I(\text{laura}), I(\text{peter})) \in I(\text{sieht})$ , sonst falsch

### Interpretation 3: Atomare Formeln

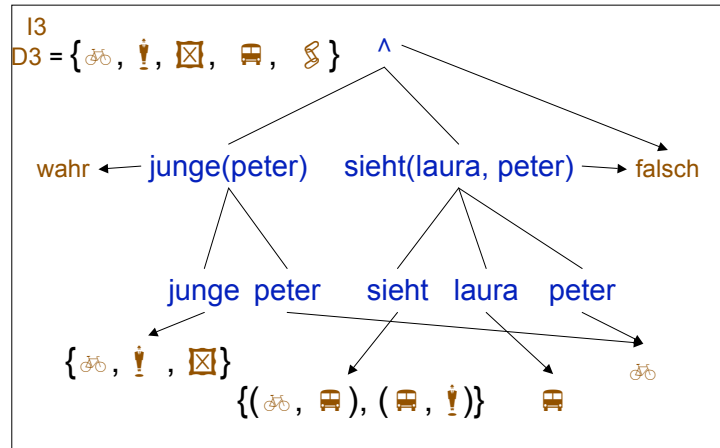


### Interpretation von Junktoren

Sei  $I$  eine Interpretation

- $I(\neg F) = \text{wahr}$ , wenn  $I(F) = \text{falsch}$ , sonst falsch
- $I((F \wedge G)) = \text{wahr}$ , wenn  $I(F) = I(G) = \text{wahr}$ , sonst falsch
- $I((F \vee G)) = \text{wahr}$ , wenn  $I(F) = \text{wahr}$  oder  $I(G) = \text{wahr}$ , sonst falsch
  - $I((F \vee G)) = \text{falsch}$ , wenn  $I(F) = I(G) = \text{falsch}$ , sonst wahr
- $I((F \Rightarrow G)) = \text{falsch}$ , wenn  $I(F) = \text{wahr}$  und  $I(G) = \text{falsch}$ , sonst wahr
  - $I((F \Rightarrow G)) = \text{wahr}$ , wenn  $I(F) = \text{falsch}$  oder  $I(G) = \text{wahr}$ , sonst falsch

### Interpretation 3: Formeln mit Junktoren



### Interpretation von Quantoren

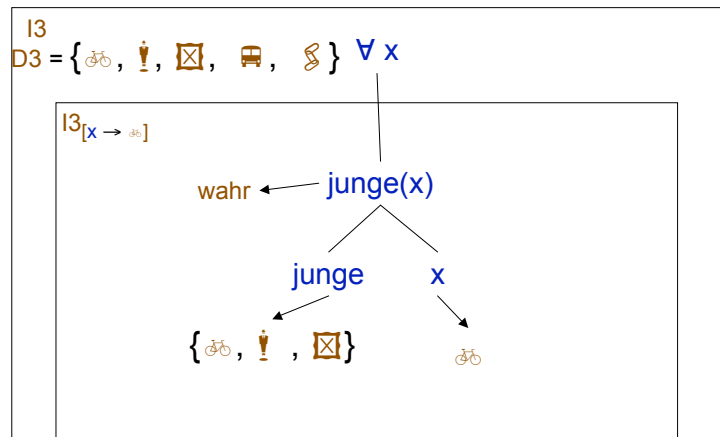
Sei  $D$  eine Menge und  $I$  eine Interpretation in  $D$

- bzgl. jeder Variable ( $x$ ) und jedes  $d \in D$  lässt sich die Interpretation  $I$  ergänzen / variieren
- $I_{[x \rightarrow d]}(x) = d$ , für alle anderen Symbole  $\pi$ :  $I_{[x \rightarrow d]}(\pi) = I(\pi)$

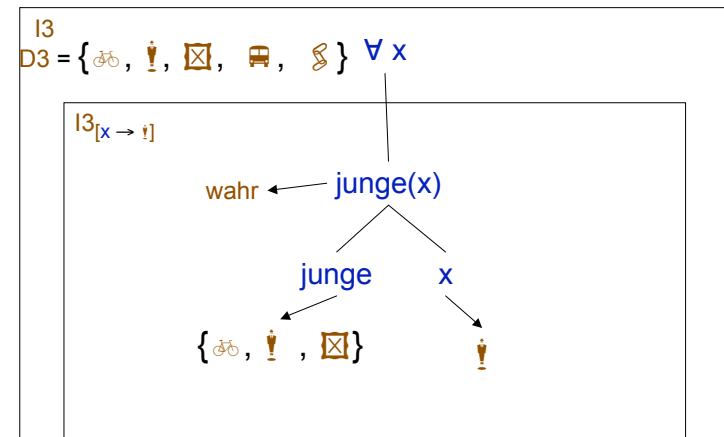
Quantoren-Interpretation erfordert Variation

- $I(\exists x [F]) = \text{wahr}$ , wenn es ein  $d \in D$  gibt, so dass  $I_{[x \rightarrow d]}(F) = \text{wahr}$ , sonst falsch
- $I(\forall x [F]) = \text{falsch}$ , wenn es ein  $d \in D$  gibt, so dass  $I_{[x \rightarrow d]}(F) = \text{falsch}$ , sonst wahr
- $I(\forall x [F]) = \text{wahr}$ , wenn für jede Wahl  $d \in D$   $I_{[x \rightarrow d]}(F) = \text{wahr}$ , sonst falsch

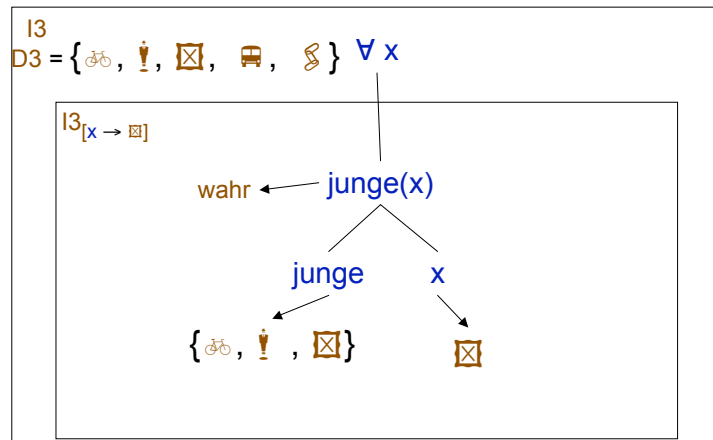
### Interpretation 3: Quantoren und Variationen (1)



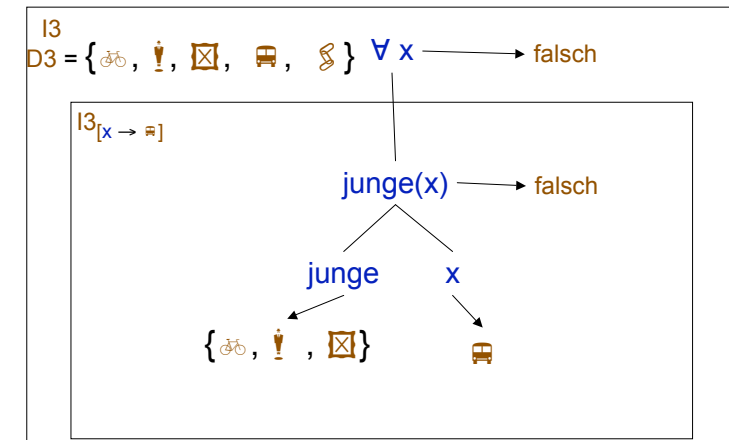
### Interpretation 3: Quantoren und Variationen (2)



### Interpretation 3: Quantoren und Variationen (3)



### Interpretation 3 : Quantoren und Variationen (4)



### Diskussion

#### Aufgabe

Welchen Wahrheitswert erhalten die Formeln

$(\text{sieht}(\text{peter}, \text{laura}) \vee \text{lacht}(\text{laura}))$

$\forall x [\text{junge}(x) \Rightarrow \text{lacht}(x)]$

unter den Interpretationen

$D_1 = \{a_1, a_2, a_3, a_4, a_5\}$

$I_1$ :  $\text{peter} \rightarrow a_1, \text{laura} \rightarrow a_3, \text{junge} \rightarrow \{a_1, a_2, a_4\},$   
 $\text{lacht} \rightarrow \{a_1, a_3\}, \text{sieht} \rightarrow \{(a_1, a_3), (a_1, a_5)\}$

$D_2 = \{a_1, a_2, a_3\}$

$I_2$ :  $\text{peter} \rightarrow a_1, \text{laura} \rightarrow a_2, \text{junge} \rightarrow \{a_1, a_3\},$   
 $\text{lacht} \rightarrow \{a_1, a_3\}, \text{sieht} \rightarrow \{(a_1, a_2), (a_3, a_1), (a_1, a_1)\}$

was ist mit

$(\text{sieht}(\text{peter}, x) \vee \text{lacht}(x))$

#### Darstellung in PROLOG

Prädikatenlogische Formeln

- Prüfung auf Wohlgeformtheit

Interpretationen

Modell-Prüfer (Model Checker)

- Berechnung des Wahrheitswertes einer wohlgeformten Formel

## Prädikatenlogik in PROLOG: Beispiel

```
example_F(e1, lacht(peter)).
example_F(e2, sieht(peter, laura) v lacht(laura)).
example_F(e3, sieht(peter, A) v lacht(A)).
example_F(e4, forall(X, junge(X) -> lacht(X))).
example_F(e5, exists(X, junge(X) & winkt(X))).
example_F(e6, ~ forall(X, junge(X) -> lacht(X))).
example_F(e7, ~ exists(X, junge(X) & winkt(X))).
example_F(e8, ~ exists(X, junge(X) & ~ lacht(X))).
example_F(e9, forall(X, sieht(X, A) v sieht(A, X))).
```

## Prädikatenlogik in PROLOG

- Formeln und Terme werden durch PROLOG-Terme repräsentiert
- Variablen
  - PROLOG-Variablen

### Kein Symbol-Zeichensatz in PROLOG

- Junktoren
  - &, v, ->, ~ anstelle von  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ ,  $\neg$
  - definiert als (Infix-)Operatoren
- Quantoren
  - exists(X, F), forall(Y, F) anstelle von  $\exists x [F]$ ,  $\forall y [F]$
- freie Symbole: Deklaration von Konstanten und Relationen
  - constant(laura).      relation(junge,1).

## Prädikatenlogik in PROLOG: Vokabular

```
% constant ?Atom
constant(peter).
constant(laura).

%relation ?Atom ?Stelligkeit
relation(junge,1).
relation(haus,1).
relation(winkt,1).
relation(lacht,1).
relation(rot,1).

relation(sieht,2).
```

## Wohlgeformtheitstest (Variante 3)

**wft/1: ist das Argument ein wohlgeformter Term ?**

```
wft(A) :- var(A).
wft(A) :- \+var(A), constant(A).
```

**wfts/1: ist das Argument eine Liste von wohlgeformten Termen ?**

```
wfts([]).
wfts([A | RestList]) :- wft(A), wfts(RestList).
```

**wff/1: ist das Argument eine wohlgeformte Formel ? z.B.**

```
wff(Formula1 v Formula2):- wff(Formula1), wff(Formula2).
wff(exists(X, Formula)) :- wff(Formula), var(X).
wff(Formula):-
    Formula =.. [Pred | ArgList],
    relation(Pred, Arity),
    length(ArgList, Arity), wfts(ArgList).
```

## Interpretationen in PROLOG

### Domäne

- Mengen werden durch Listen dargestellt  
`domain(i1, [a1, a2, a3, a4, a5]).`

### Interpretation

- Für die Darstellung von Abbildungen wird auch `->` verwendet  
`interpretation(i1,  
 [peter -> a1, laura -> a3,  
 junge -> [a1, a2, a4], haus -> [a5],  
 winkt -> [a1, a2], lacht -> [a1, a3],  
 rot -> [a5],  
 sieht -> [(a1, a3), (a1, a5)]).`

### Variablenbelegungen

`[X -> a1, Y -> a4]`

## Einbindung des Modell-Prüfers (Variante 3)

```
evaluate_F(Formula, Model):-  
  wff(Formula), % evaluiere nur wohlgeformte Formeln  
  domain(Model, Domain), % die Domäne  
  interpretation(Model, Interpretation), % die Interpretation  
  freieVariablenBelegung_F(Formula, Domain, []-VarBelegung),  
  % generiere eine Variablenbelegung fuer die freien Variablen  
  ... % Textausgabe  
  ((satisfy(Formula, Model, VarBelegung), % Puffe, ob erfuehlt  
    writeln('erfuehlt.')); % und gib das Ergebnis bekannt  
  (dissatisfy(Formula, Model, VarBelegung),%oder nicht erfuehlt  
    writeln('nicht erfuehlt.'))) . % gib das Ergebnis bekannt
```

## Modell-Prüfer 3 (Ausschnitt, vereinfacht)

```
satisfy(Formula1 & Formula2, Model, VarBelegung):-  
  satisfy(Formula1, Model, VarBelegung),  
  satisfy(Formula2, Model, VarBelegung).  
satisfy(~ Formula, Model, VarBelegung):-  
  dissatisfy(Formula, Model, VarBelegung).  
satisfy(exists(X, Formula), Model, VarBelegung) :-  
  domain(Model, Domain), member(D, Domain),  
  satisfy(Formula, Model, [X -> D |VarBelegung]).  
satisfy(Formula,Model, VarBelegung):-  
  Formula =..[Pred | ArgList],  
  evaluateArgs(ArgList, Args, Model, VarBelegung),  
  interpretation(Model, Interpretation),  
  member(Pred -> Value, Interpretation),  
  member(Args, Value).
```



## Ausprobieren?

**exampleModels.pl:** Beispiel Operatoren und Modelle  
**modelChecker1.pl:** ohne Variablen und Quantoren  
**modelChecker2.pl:** gebundene Variablen und Quantoren, keine freien Variablen  
**modelChecker3.pl:** sogar freie Variablen