

---

# Semantische Sprachverarbeitung

---

Carola Eschenbach

Universität Hamburg, MIN-Fakultät, Dept. Informatik  
AB Wissens- und Sprachverarbeitung (WSV)

Sommersemester 2008

---

---

# Semantische Sprachverarbeitung

---

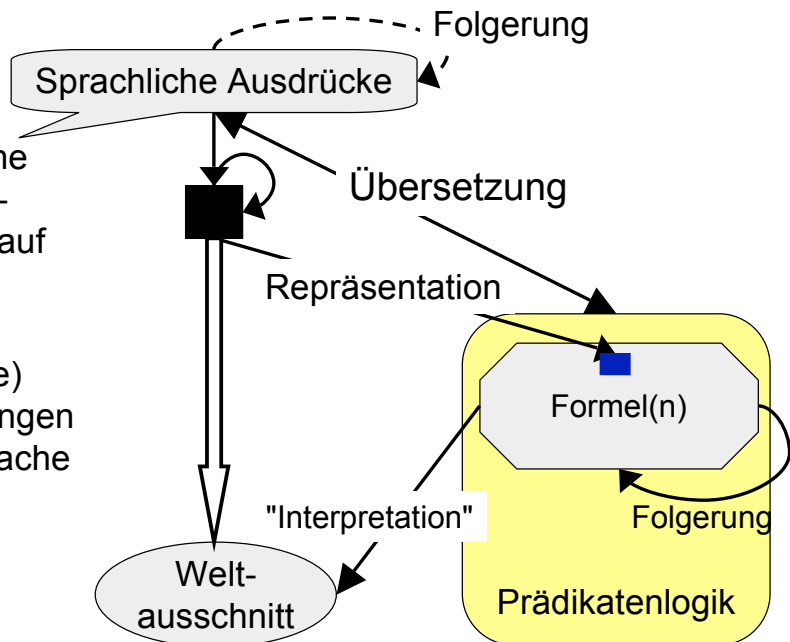
## Sitzung 4

- Techniken der Bedeutungserfassung
    - Logik : Semantik
  - Syntaxgesteuerter Bedeutungsaufbau
-

## 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



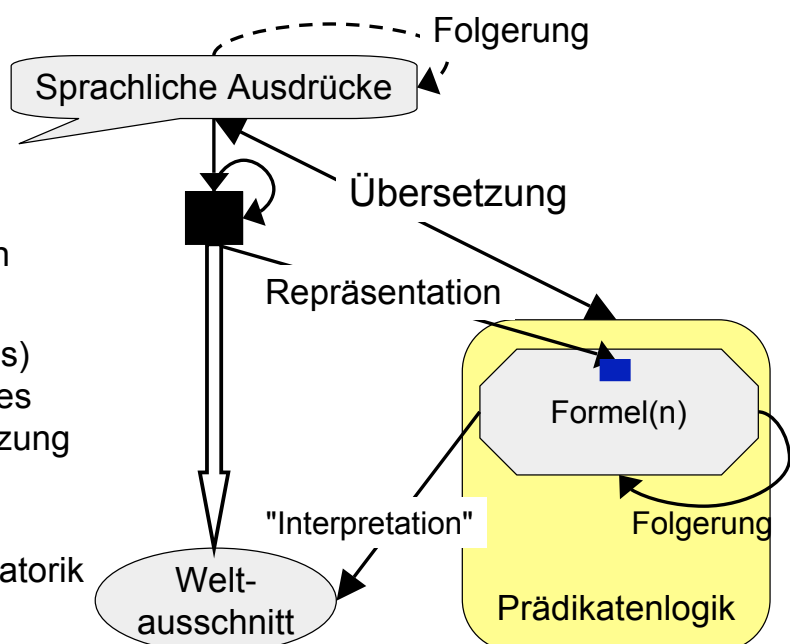
## Formale Linguistik

### Logik dient

- zur Repräsentation der Bedeutungen
- zur Kommunikation zwischen Forschern

### Ziel

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



# 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

# Prädikatenlogische Repräsentation: Inventar

---

## Nicht-logisches Inventar

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

## Logisches Inventar

- (feste, logische Symbole)
- Junktoren, einstellig:  $\neg$  (Negation)  
zweistellig:  $\wedge$  (Konjunktion),  $\vee$  (Disjunktion),  $\Rightarrow$  (Implikation)
- Quantoren:  $\forall$  (All-Quantor),  $\exists$  (Existenzquantor)
- Variablen: *x*, *y*, *z*, ...

## Hilfssymbole

- Klammern, Komma

## 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:  $\neg lacht(peter)$
- Kombination von zwei Formeln mit einem zweistelligen Junktor:  $(winkt(peter) \wedge \neg lacht(peter))$
- Kombination aus Quantor, Variable und Formel:  $\exists x [lacht(x)]$

## 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

# 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/FGI1-Vorlesung)
- z.B. Tableau-Beweiser ( $\rightarrow$  LOS/FGI3-Logik-Vorlesung)

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

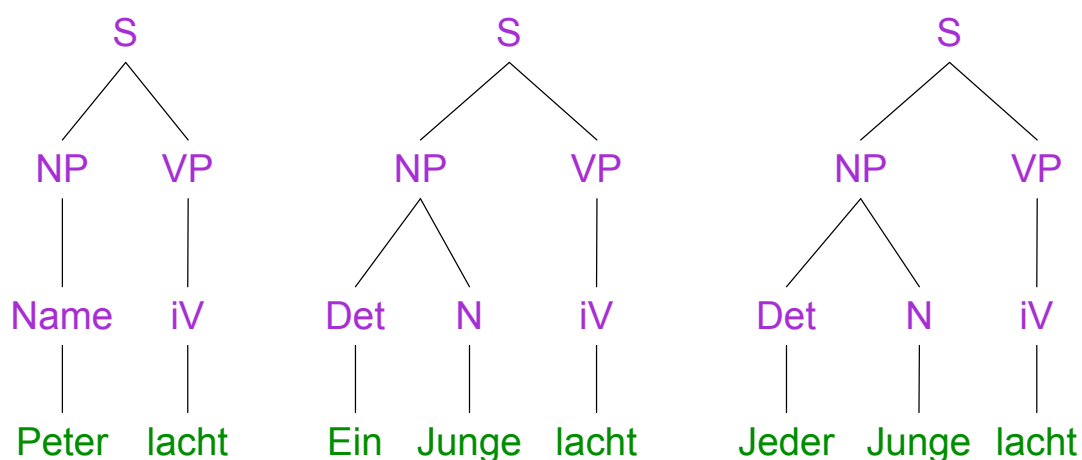
- für jedes Beweisverfahren gibt es eine Aufgabe, die es nicht (in endlicher Zeit) bearbeiten kann
- man muss damit rechnen, kein Ergebnis zu erhalten

### Bestimmte Fragmente der PL sind entscheidbar

## Systematische Variationen

NP-Variation	VP-Konjunktion
$[_{NP} \text{ Peter}] [_{VP} \text{ lacht}]$ . $\text{lacht}(\text{peter})$	$[_{NP} \text{ Peter}] [_{VP} \text{ lacht und winkt}]$ . $\text{lacht}(\text{peter}) \wedge \text{winkt}(\text{peter})$
$[_{NP} \text{ Ein Junge}] [_{VP} \text{ lacht}]$ . $\exists x [\text{junge}(x) \wedge \text{lacht}(x)]$	$[_{NP} \text{ Ein Junge}] [_{VP} \text{ lacht und winkt}]$ . $\exists x [\text{junge}(x) \wedge (\text{lacht}(x) \wedge \text{winkt}(x))]$
$[_{NP} \text{ Jeder Junge}] [_{VP} \text{ lacht}]$ . $\forall x [\text{junge}(x) \Rightarrow \text{lacht}(x)]$	$[_{NP} \text{ Jeder Junge}] [_{VP} \text{ lacht und winkt}]$ . $\forall x [\text{junge}(x) \Rightarrow (\text{lacht}(x) \wedge \text{winkt}(x))]$
$[_{NP} \text{ Kein Junge}] [_{VP} \text{ lacht}]$ . $\neg \exists x [\text{junge}(x) \wedge \text{lacht}(x)]$	$[_{NP} \text{ Kein Junge}] [_{VP} \text{ lacht und winkt}]$ . $\neg \exists x [\text{junge}(x) \wedge (\text{lacht}(x) \wedge \text{winkt}(x))]$
Satz-Konjunktion	
$[_{NP} \text{ Peter}] [_{VP} \text{ lacht}]$ und $[_{NP} \text{ Peter}] [_{VP} \text{ winkt}]$ . $\text{lacht}(\text{peter}) \wedge \text{winkt}(\text{peter})$	
$[_{NP} \text{ Ein Junge}] [_{VP} \text{ lacht}]$ und $[_{NP} \text{ ein Junge}] [_{VP} \text{ winkt}]$ . $\exists x [\text{junge}(x) \wedge \text{lacht}(x)] \wedge \exists y [\text{junge}(y) \wedge \text{winkt}(y)]$	
$[_{NP} \text{ Jeder Junge}] [_{VP} \text{ lacht}]$ und $[_{NP} \text{ jeder Junge}] [_{VP} \text{ winkt}]$ . $\forall x [\text{junge}(x) \Rightarrow \text{lacht}(x)] \wedge \forall y [\text{junge}(y) \Rightarrow \text{winkt}(y)]$	
$[_{NP} \text{ Kein Junge}] [_{VP} \text{ lacht}]$ und $[_{NP} \text{ kein Junge}] [_{VP} \text{ winkt}]$ . $\neg \exists x [\text{junge}(x) \wedge \text{lacht}(x)] \wedge \neg \exists y [\text{junge}(y) \wedge \text{winkt}(y)]$	

## Beispiel: Syntaktische Strukturen



## Eine einfache Grammatik mit Lexikon für F1

---

### Grammatik

s → np vp  
s → s conj s  
vp → tv np  
vp → iv  
vp → vp conj vp  
np → pn  
np → det n  
n → adj n

### Lexikon

n → haus  
n → junge  
adj → rotes  
det → ein  
det → jeder  
det → kein  
pn → peter  
pn → laura  
iv → lacht  
iv → winkt  
tv → sieht  
conj → und  
conj → oder

## Eine DCG Grammatik für F1

---

### Grammatik

s --> np, vp.  
s --> s, conj, s.  
vp --> tv, np.  
vp --> iv.  
vp --> vp, conj, vp.  
np --> pn.  
np --> det, n.  
n --> adj, n.

### Lexikon

n --> [haus].  
n --> [junge].  
adj --> [rotes].  
det --> [ein].  
det --> [jeder].  
det --> [kein].  
pn --> [peter].  
pn --> [laura].  
iv --> [lacht].  
iv --> [winkt].  
tv --> [sieht].  
conj --> [und].  
conj --> [oder].

## Erzeugung des Syntaxbaums (F1)

---

s(s/[NP,VP])	---> np(NP), vp(VP).	n(n/ (-haus))	---> lexem(haus).
s(s/[S1,C,S2])	---> s(S1), conj(C), s(S2).	n(n/ (-junge))	---> lexem(junge).
vp(vp/[TV,NP])	---> tv(TV), np(NP).	adj(adj/ (-rotes))	---> lexem(rotes).
vp(vp/[IV])	---> iv(IV).	det(det/ (-ein))	---> lexem(ein).
vp(vp/[VP1,C,VP2])	---> vp(VP1), conj(C), vp(VP2).	det(det/ (-jeder))	---> lexem(jeder).
np(np/[PN])	---> pn(PN).	det(det/ (-kein))	---> lexem(kein).
np(np/[Det, N])	---> det(Det), n(N).	pn(pn/ (-peter))	---> lexem(peter).
n(n/[Adj, N])	---> adj(Adj), n(N).	pn(pn/ (-laura))	---> lexem(laura).
		iv(iv/ (-lacht))	---> lexem(lacht).
		iv(iv/ (-winkt))	---> lexem(winkt).
		tv(tv/ (-sieht))	---> lexem(sieht).
		conj(conj/ (-und))	---> lexem(und).
		conj(conj/ (-oder))	---> lexem(oder).

Parserinformation  
Syntaxbaum  
Sprachlicher Ausdruck

## Das Lexikon

---

- Appendix zur Grammatik
- Sammlung von Ausnahmen
- ungeordnete Liste von Worten / Lexemen
- Eigenständiges regelhaftes Modul (→ Morphologie)
- Schnittstelle zwischen
  - phonologischer
  - orthographischer
  - syntaktischer und
  - semantischer Ebene

## Lexikalische Einträge (LE)

---

**Oberflächenform (Wort) / phonologische Form / orthographische Form**

**grammatische Merkmale**

- Wortart (Wortform)

**Bedeutungsbeitrag (semantische Form)**

### Beispiel

lexem(junge, n(nom, sg, mask), junge).

lexem(winkt, iv(fin(pres, ind, sg, 3)), winkt).

lexem(ein, det(nom, sg, mask), indef).

## Erzeugung des Syntaxbaums (F1)

---

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

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

vp(vp/[TV,NP]) ---> tv(TV), np(NP).

vp(vp/[IV]) ---> iv(IV).

vp(vp/[VP1,C,VP2]) ---> vp(VP1),  
conj(C), vp(VP2).

np(np/[PN]) ---> pn(PN).

np(np/[Det, N]) ---> det(Det), n(N).

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

n(n/ (-Lex)) ---> lexem(Lex, n).

adj(adj/ (-Lex)) ---> lexem(Lex, adj).

det(det/ (-Lex)) ---> lexem(Lex, det).

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

iv(iv/ (-Lex)) ---> lexem(Lex, iv).

tv(tv/ (-Lex)) ---> lexem(Lex, tv).

conj(conj/ (-Lex)) ---> lexem(Lex, conj).

lexem(haus, n). lexem(junge, n).

lexem(rotes, adj). lexem(ein, det).

lexem(jeder, det). lexem(kein, det).

lexem(peter, pn). lexem(laura, pn).

lexem(lacht, iv). lexem(winkt, iv).

lexem(sieht, tv).

lexem(und, conj). lexem(oder, conj).

Parserinformation

Syntaxbaum/Wortart

Sprachlicher Ausdruck

# Korrespondenz Syntax - Semantik

---

## Syntax

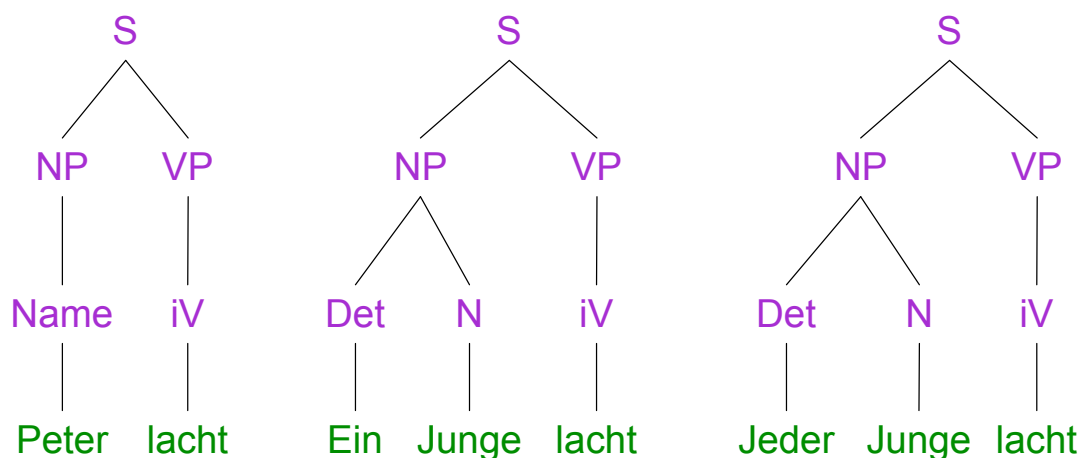
- Ausdrücke (Wörter, Konstituenten, Phrasen) gehören **syntaktischen Kategorien** an, die ihren grammatischen Positionsmöglichkeiten entsprechen.
  - Verb (intransitiv, transitiv, di-transitiv, ...), Verbalphrase
  - Nomen, Nominalphrase, Artikel, Determinator, Pronomen

## Semantik

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

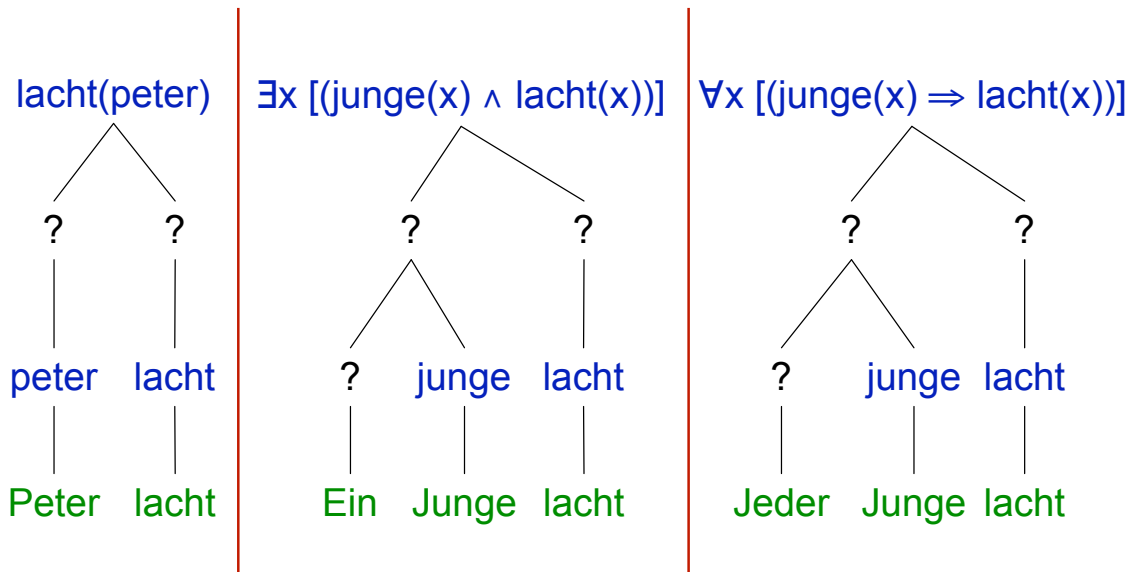
## Beispiel: Syntaktische Strukturen

---



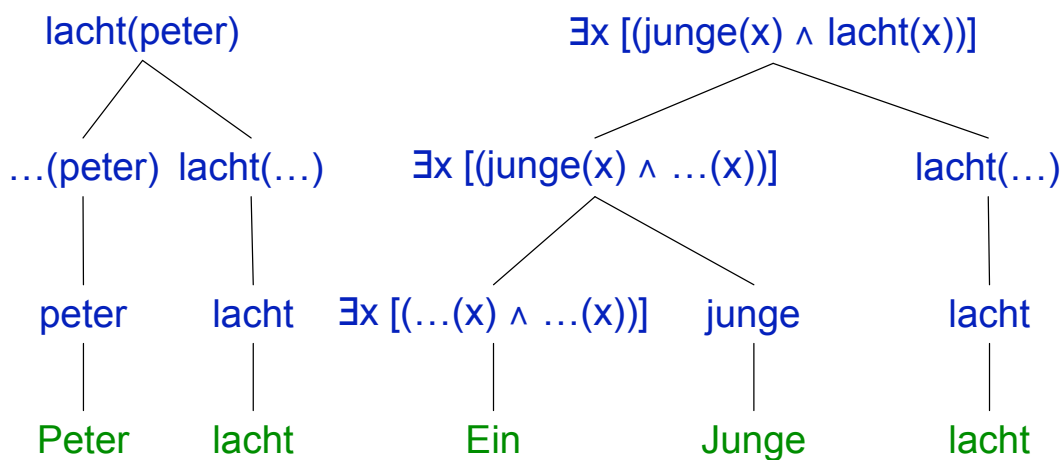
## Beispiel

---



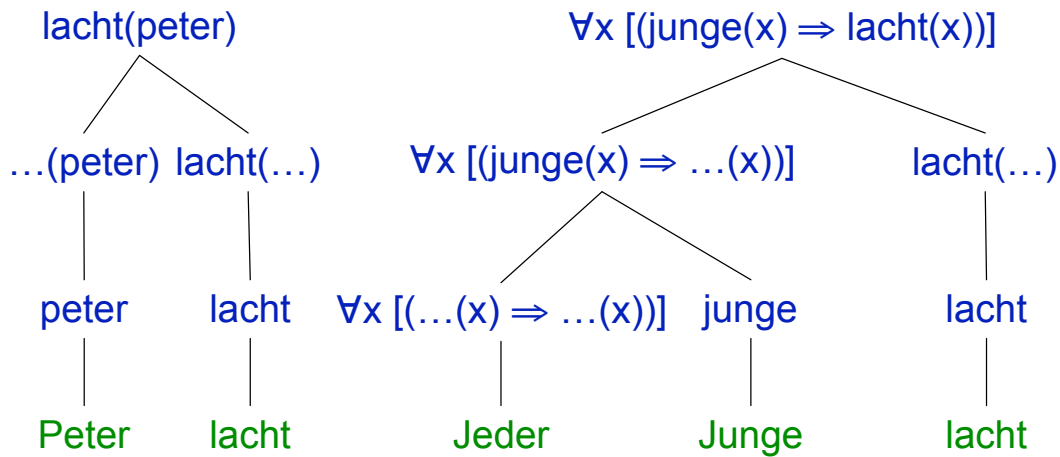
## Beispiel

---



## 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

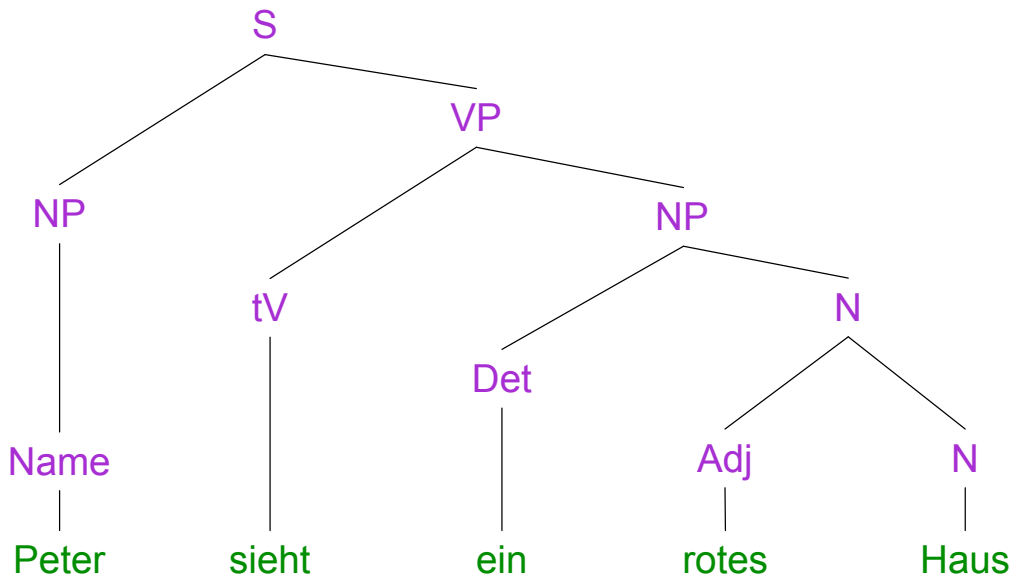
### der Typ des Ausdrucks

- determiniert, welche Lücken vom Kontext zu füllen sind

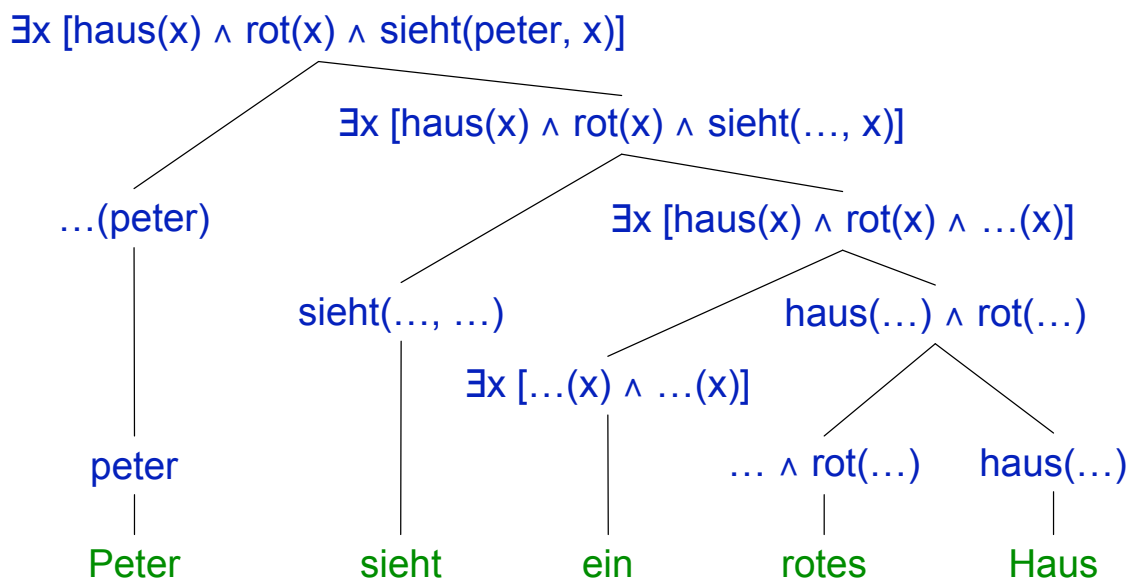
### Regeln

- determinieren, wie die Lücken gefüllt werden

## Peter sieht ein rotes Haus.



## Wahrheitsbedingungen: Peter sieht ein rotes Haus.



## Verknüpfung von Lexikon und Grammatik (F1)

---

$n(n \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } n, \text{Sem)}$ .

$adj(adj \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } adj, \text{Sem)}$ .

$det(det \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } det, \text{Sem)}$ .

$pn(pn \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } pn, \text{Sem)}$ .

$iv(iv \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } iv, \text{Sem)}$ .

$tv(tv \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } tv, \text{Sem)}$ .

$conj(conj/ \text{ (-Lexem)}, \text{Sem}) \text{ ---> lexem(Lexem, } conj, \text{Sem)}$ .

## 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

## 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).

## Satz-Konjunktionen (F1)

### verknüpfen zwei Sätze zu einem Satz

#### Lexikon

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

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

#### Regeln

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

#### Verknüpfung von Lexikon und Grammatik

conj(conj/ (-Lexem), Sem) ---> lexem(Lexem, conj, Sem).

Parser-Information

Syntaxbaum

Semantik

Wortform

inhaltlicher Beitrag

interne Lücke

Lücken-Info nach außen geben

Lücken-Info nutzen und füllen

Resultat nach außen geben

## Satz- und VP-Konjunktionen (F1)

---

**verknüpfen zwei Sätze zu einem Satz**

### Lexikon

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

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

### Regeln

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

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

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

## Nomen und Adjektive (F1)

---

**Nomen drücken Eigenschaften aus**  
**Adjektive modifizieren Eigenschaften**

### Lexikon

**lexem**(haus, n, [X]/(haus(X))).

**lexem**(junge, n, [X]/(junge(X))).

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

### Regel

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

## Nominalphrasen (F1)

---

### Determinatoren verknüpfen Nomen und VPs

#### Lexikon

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

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

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

lexem(peter, pn, peter).

lexem(laura, pn, 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)

---

#### Lexikon

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

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

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

#### 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 (F1)

---

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).

## Verknüpfung von Lexikon und Grammatik (F1)

---

n(n (-Lexem), Sem)	----> lexem(Lexem, n, Sem).
adj(adj (-Lexem), Sem)	----> lexem(Lexem, adj, Sem).
det(det (-Lexem), Sem)	----> lexem(Lexem, det, Sem).
pn(pn (-Lexem), Sem)	----> lexem(Lexem, pn, Sem).
iv(iv (-Lexem), Sem)	----> lexem(Lexem, iv, Sem).
tv(tv (-Lexem), Sem)	----> lexem(Lexem, tv, Sem).
conj(conj/ (-Lexem), Sem)	----> lexem(Lexem, conj, Sem).

## Lexikon mit Syntax und Semantik (F1)

---

`lexem(haus, n, [X]/(haus(X)))`.  
`lexem(junge, n, [X]/(junge(X)))`.  
`lexem(rotes, adj, [X, F]/(rot(X) & F))`.  
`lexem(ein, det, [X, F, G]/exists(X, F & G))` .  
`lexem(jeder, det, [X, F, G]/forall(X, F -> G))`.  
`lexem(kein, det, [X, F, G]/(~ exists(X, F & G)))`.  
`lexem(peter, pn, peter)`.  
`lexem(laura, pn, laura)`.  
`lexem(lacht, iv, [X]/lacht(X))`.  
`lexem(winkt, iv, [X]/winkt(X))`.  
`lexem(sieht, tv, [X, Y]/sieht(X, Y))`.  
`lexem(und, conj, [F, G]/(F & G))`.  
`lexem(oder, conj, [F, G]/(F v G))`.

## Ausprobieren ?

---

### Paket PL

#### main.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 ?