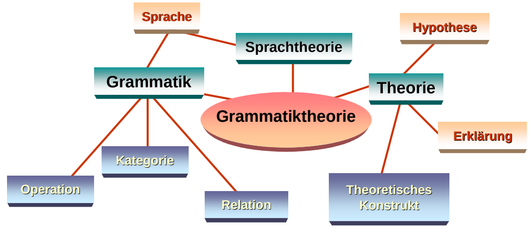
Grammatik als Deduktionssystem

Unifikations-Grammatiken (UG)

Theorie, Grammatik, Grammatiktheorie



Unifikations-Grammatiken (UG)

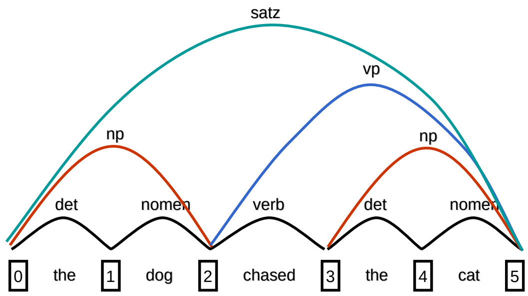
Grundbegriffe: Theorie und Wissenschaftssprache 1

Bei der Theoriebildung müssen grundsätzlich drei Aspekte unterschieden werden:

- Der Objektbereich, der durch eine Theorie erklärt oder durch ein Modell modelliert werden soll. Dieser ist immer schon im Sinne eines Formalobjektes zu verstehen.
- Die Theorie selbst, die den Objektbereich beschreibt und erklärt. Der gleiche Objektbereich kann durch unterschiedliche Theorien erklärt werden.
- Die Sprache, in der eine Theorie ausgedrückt wird.
  - Die gleiche Theorie kann gegebenenfalls durch unterschiedliche Sprachen ausgedrückt werden.
  - Ausdrücke aus verschiedenen Sprachen sind dann ineinander übersetzbar.

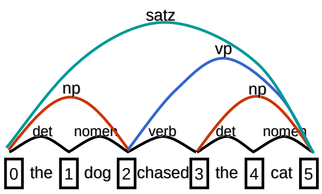
Unifikations-Grammatiken (UG)

Phrasenstrukturgrammatik (PSG) als Deduktionssystem



Unifikations-Grammatiken (UG)

PSG als Deduktionssystem 1



Zwischen **0** und **1** liegt ein **det** - oder

Was zwischen **0** und **1** liegt , ist ein **det**

det(0, 1)

det(0, 1)

nomen(1, 2)

verb(2, 3)

det(3, 4)

nomen(4, 5)

np(0, 2)

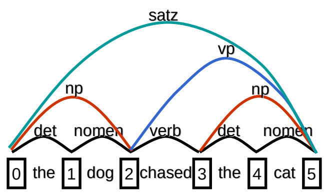
np(3, 5)

vp(2, 5)

satz(0, 5)

Unifikations-Grammatiken (UG)

PSG als Deduktionssystem 2



Zwischen $\mathbf{p_0}$ und **p** liegt ein **np**, falls es ein $\mathbf{p_1}$ gibt derart, dass zwischen $\mathbf{p_0}$ und $\mathbf{p_1}$ ein **det** liegt und zwischen $\mathbf{p_1}$ und **p** ein **nomen**.

np(p_0, p) ← det(p_0, p_1), noun(p_1, p)

det(0, 1)

nomen(1, 2)

verb(2, 3)

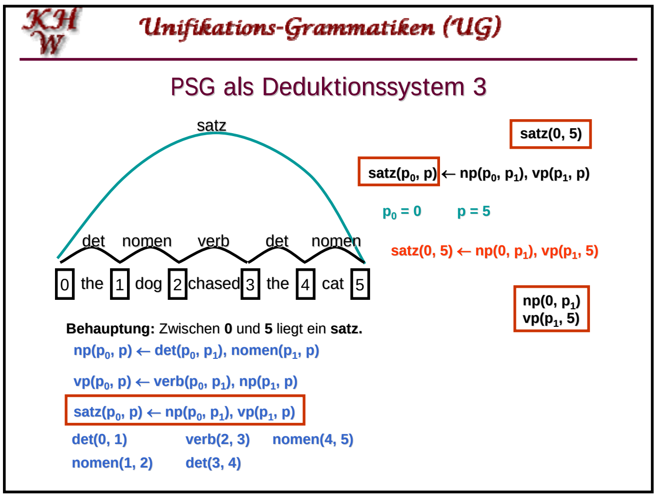
det(3, 4)

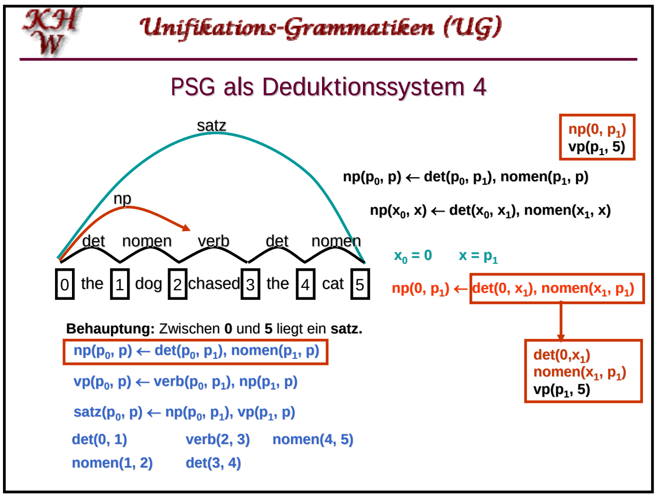
nomen(4, 5)

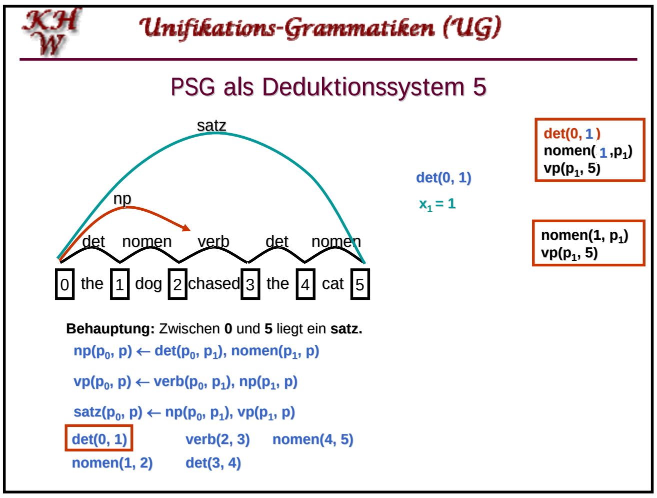
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)

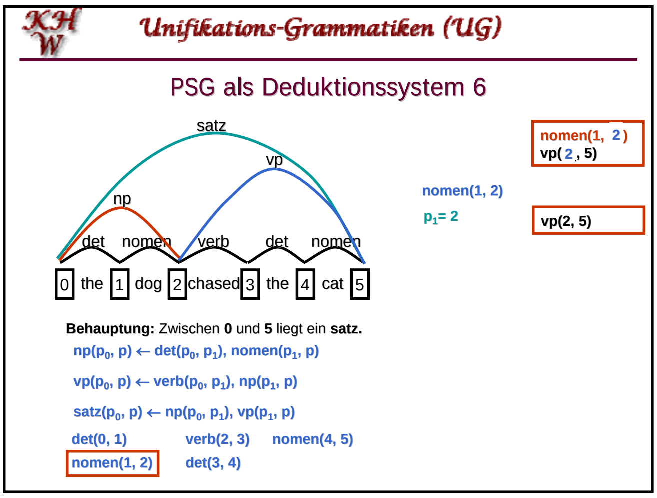
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)

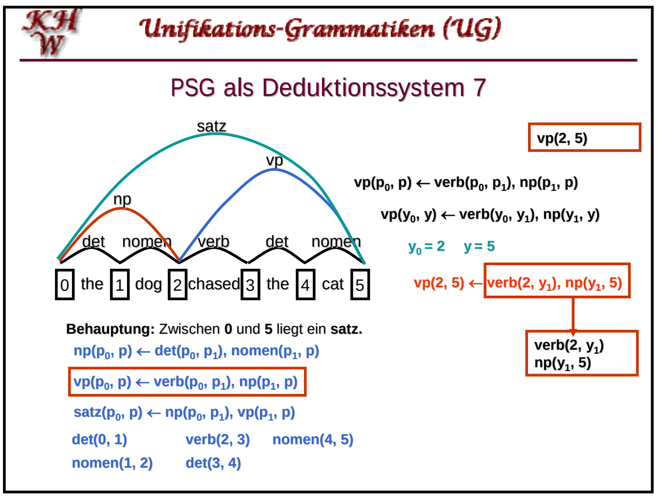
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)

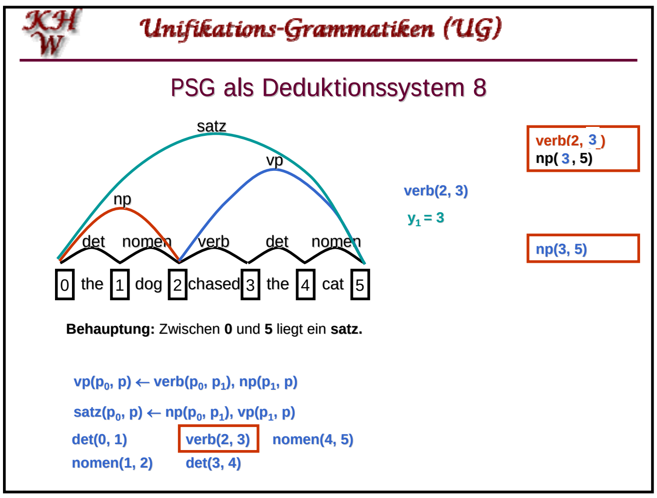

Unifikations-Grammatiken (UG)
PSG als Deduktionssystem 3
satz
det nomen verb det nomen
0 the 1 dog 2 chased 3 the 4 cat 5
satz(0, 5)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
p_0 = 0 p = 5
satz(0, 5) ← np(0, p_1), vp(p_1, 5)
np(0, p_1)
vp(p_1, 5)
Behauptung: Zwischen 0 und 5 liegt ein satz.
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
det(0, 1) verb(2, 3) nomen(4, 5)
nomen(1, 2) det(3, 4)


Unifikations-Grammatiken (UG)
PSG als Deduktionssystem 4
satz
np
det nomen verb det nomen
0 the 1 dog 2 chased 3 the 4 cat 5
np(0, p_1)
vp(p_1, 5)
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)
np(x_0, x) ← det(x_0, x_1), nomen(x_1, x)
x_0 = 0 x = p_1
np(0, p_1) ← det(0, x_1), nomen(x_1, p_1)
det(0,x_1)
nomen(x_1, p_1)
vp(p_1, 5)
Behauptung: Zwischen 0 und 5 liegt ein satz.
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
det(0, 1) verb(2, 3) nomen(4, 5)
nomen(1, 2) det(3, 4)


Unifikations-Grammatiken (UG)
PSG als Deduktionssystem 5
satz
np
det nomen verb det nomen
0 the 1 dog 2 chased 3 the 4 cat 5
det(0, 1)
nomen(1 ,p_1)
vp(p_1, 5)
det(0, 1)
x_1 = 1
nomen(1, p_1)
vp(p_1, 5)
Behauptung: Zwischen 0 und 5 liegt ein satz.
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
det(0, 1) verb(2, 3) nomen(4, 5)
nomen(1, 2) det(3, 4)

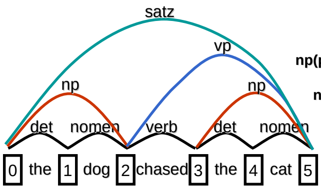

Unifikations-Grammatiken (UG)
PSG als Deduktionssystem 6
satz
vp
np
det nomen verb det nomen
0 the 1 dog 2 chased 3 the 4 cat 5
nomen(1, 2)
vp(2 , 5)
nomen(1, 2)
p_1= 2
vp(2, 5)
Behauptung: Zwischen 0 und 5 liegt ein satz.
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
det(0, 1) verb(2, 3) nomen(4, 5)
nomen(1, 2) det(3, 4)


Unifikations-Grammatiken (UG)
PSG als Deduktionssystem 7
satz
vp
np
det nomen verb det nomen
0 the 1 dog 2 chased 3 the 4 cat 5
vp(2, 5)
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
vp(y_0, y) ← verb(y_0, y_1), np(y_1, y)
y_0 = 2 y = 5
vp(2, 5) ← verb(2, y_1), np(y_1, 5)
verb(2, y_1)
np(y_1, 5)
Behauptung: Zwischen 0 und 5 liegt ein satz.
np(p_0, p) ← det(p_0, p_1), nomen(p_1, p)
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
det(0, 1) verb(2, 3) nomen(4, 5)
nomen(1, 2) det(3, 4)


Unifikations-Grammatiken (UG)
PSG als Deduktionssystem 8
satz
vp
np
det nomen verb det nomen
0 the 1 dog 2 chased 3 the 4 cat 5
verb(2, 3)
np(3 , 5)
verb(2, 3)
y_1 = 3
np(3, 5)
Behauptung: Zwischen 0 und 5 liegt ein satz.
vp(p_0, p) ← verb(p_0, p_1), np(p_1, p)
satz(p_0, p) ← np(p_0, p_1), vp(p_1, p)
det(0, 1) verb(2, 3) nomen(4, 5)
nomen(1, 2) det(3, 4)

Unifikations-Grammatiken (UG)

PSG als Deduktionssystem 9



np(3, 5)

$np(p_0, p) \leftarrow det(p_0, p_1), nomen(p_1, p)$

$np(z_0, z) \leftarrow det(z_0, z_1), nomen(z_1, z)$

$z_0 = 3$ $z = 5$

$np(3, 5) \leftarrow det(3, z_1), nomen(z_1, 5)$

det(3, z_1)
nomen(z_1, 5)

Behauptung: Zwischen **0** und **5** liegt ein **satz.**

$np(p_0, p) \leftarrow det(p_0, p_1), nomen(p_1, p)$

$vp(p_0, p) \leftarrow verb(p_0, p_1), np(p_1, p)$

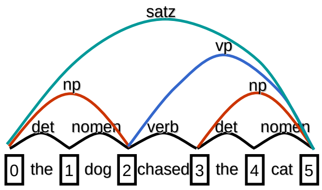
$satz(p_0, p) \leftarrow np(p_0, p_1), vp(p_1, p)$

det(0, 1) verb(2, 3) nomen(4, 5)

nomen(1, 2) det(3, 4)

Unifikations-Grammatiken (UG)

PSG als Deduktionssystem 10



det(3, 4)
nomen(4, 5)

det(3, 4)

$z_1 = 4$

nomen(4, 5)

Behauptung: Zwischen **0** und **5** liegt ein **satz.**

$np(p_0, p) \leftarrow det(p_0, p_1), nomen(p_1, p)$

$vp(p_0, p) \leftarrow verb(p_0, p_1), np(p_1, p)$

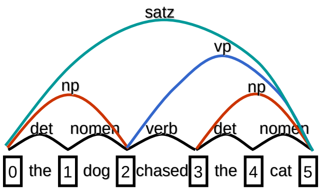
$satz(p_0, p) \leftarrow np(p_0, p_1), vp(p_1, p)$

det(0, 1) verb(2, 3) nomen(4, 5)

nomen(1, 2) det(3, 4)

Unifikations-Grammatiken (UG)

PSG als Deduktionssystem 11



nomen(4, 5)

nomen(4, 5)

Behauptung: Zwischen **0** und **5** liegt ein **satz.**

$np(p_0, p) \leftarrow det(p_0, p_1), nomen(p_1, p)$

$vp(p_0, p) \leftarrow verb(p_0, p_1), np(p_1, p)$

$satz(p_0, p) \leftarrow np(p_0, p_1), vp(p_1, p)$

det(0, 1) verb(2, 3) nomen(4, 5)

nomen(1, 2) det(3, 4)

Unifikations-Grammatiken (UG)

Termunifikation

- Wenn es bei einer Ableitung darum geht, zu beweisen, dass beispielsweise np(0,2) gilt, muss gezeigt werden, dass es eine Regel (ein Axiom) gibt, die auf diesen Fall anwendbar ist. Eine solche Regel ist z.B. $np(p_0, p) \leftarrow det(p_0, p_1), nomen(p_1, p)$. Damit die Regel jedoch angewandt werden kann, muss eine weitere Bedingung erfüllt sein.
- Die syntaktischen Grundbausteine eines Logikprogramms (eine PSG kann als ein Logikprogramm interpretiert werden) heißen Terme. Der Ausdruck np(0,2) z.B. ist ein Term.
- Die Bestandteile, aus denen die Regel zusammengesetzt ist, also $np(p_0, p)$, $det(p_0, p_1)$, und $nomen(p_1, p)$ sind ebenfalls Terme.
- Die Regel als Ganzes ist ebenfalls ein Term, wenn auch in einer speziellen Form.

Unifikations-Grammatiken (UG)

Termunifikation: Term

- Es können einfache und komplexe (zusammengesetzte) Terme unterschieden werden.
- Ein einfacher Term ist entweder eine Konstante, oder eine Variable. Die Ziffern 0 und 1 in np(0,2) sind Konstante. Die Buchstaben p_0 und p in $np(p_0, p)$ sind Variable. Variable sind Platzhalter für andere Terme.
- Komplexe Terme heißen Strukturen. Sie bestehen aus einem Funktor und einer beliebigen Anzahl von Argumenten, die ihrerseits wieder Terme sind. Man spricht daher von Funktor-Argumentstrukturen. Der komplexe Term $np(p_0, p)$ ist eine solche Struktur.

Unifikations-Grammatiken (UG)

Termunifikation: komplexe Terme

satz(nominalphrase , verbalphrase)

↑ Funktor ↑ Argument ↑ Argument

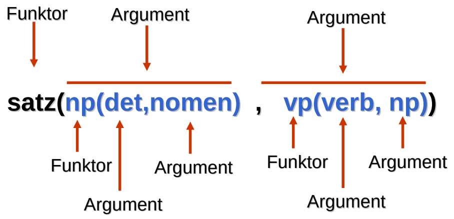
Die Argumente einer Struktur sind selbst Terme im allgemeinen Sinn, d.h. es können auch Strukturen sein, z.B.:

satz(np(det,nomen) , vp(verb, np))

↑ Funktor ↑ Argument ↑ Argument

Unifikations-Grammatiken (UG)

Termunifikation: komplexe Strukturen



Unifikations-Grammatiken (UG)

Termunifikation: Regeln

Regeln wie np(p_0, p) ← det(p_0, p_1), nomen(p_1, p) weisen eine besondere Struktur auf, eine Operand – Operator – Operand-Struktur. Terme wie np(p_0, p) sind Operanden, ← (=falls) und , (= und) sind Operatoren.

Der Term, der links des Operators ← steht, wird Kopf genannt. Was rechts davon steht, heißt Rumpf.

Damit eine Regel wie np(p_0, p) ← det(p_0, p_1), nomen(p_1, p) auf einen Term wie np(0, 2) angewandt werden kann, ist Voraussetzung, dass dieser Term und der Regelkopf in Übereinstimmung gebracht werden kann. Dies geschieht durch geeignete Substitution von der Variablen durch andere Terme, z.B. Konstante.

Unifikations-Grammatiken (UG)

Termunifikation: Substitution

Im konkreten Fall müsste die Variable p_0 durch die Konstante 0 und die Variable p durch die Konstante 2 ersetzt werden, und zwar überall wo diese Variablen im Ausdruck vorkommen.

Damit geht die Regel np(p_0, p) ← det(p_0, p_1), nomen(p_1, p) über in np(0, 2) ← det(0, p_1), nomen(p_1, 2).

Für die Ersetzung einer Variablen v_i durch einen Term t_i schreibt man v_i/t_i, z.B. p_0/0.

Eine Substitution ist eine Menge von solchen Ersetzungen $\{v_1/t_1, v_2/t_2 \ldots v_i/t_i\}$ im konkreten Fall also $\{p_0/0, p/2\}$.
Durch die Anwendung einer Substitution auf einen Term entsteht ein neuer Term, in dem alle nach der Substitution möglichen Ersetzungen vorgenommen worden sind.

np(p_0, p) $\{p_0/0, p/2\}$ = np(0,2).

Unifikations-Grammatiken (UG)

Termunifikation: Unifikation

Unter Term-Unifikation versteht man grob gesagt eine Operation durch die zwei Terme durch eine Substitution gleich gemacht (unifiziert) werden.

Die Terme np(p_0, p) und np(0, p_1) beispielsweise werden durch die Substitution $\{p_0/0, p/p_1\}$ gleich gemacht, denn es gilt:

np(p_0, p) $\{p_0/0, p/p_1\}$ = np(0, p_1) $\{p_0/0, p/p_1\}$ = np(0, p_1).

Damit komplexe Terme (Strukturen) unifiziert werden können, ist Voraussetzung, dass die Funktoren identisch sind, die Zahl der Argumente gleich ist und die Argumente paarweise unifizierbar sind.