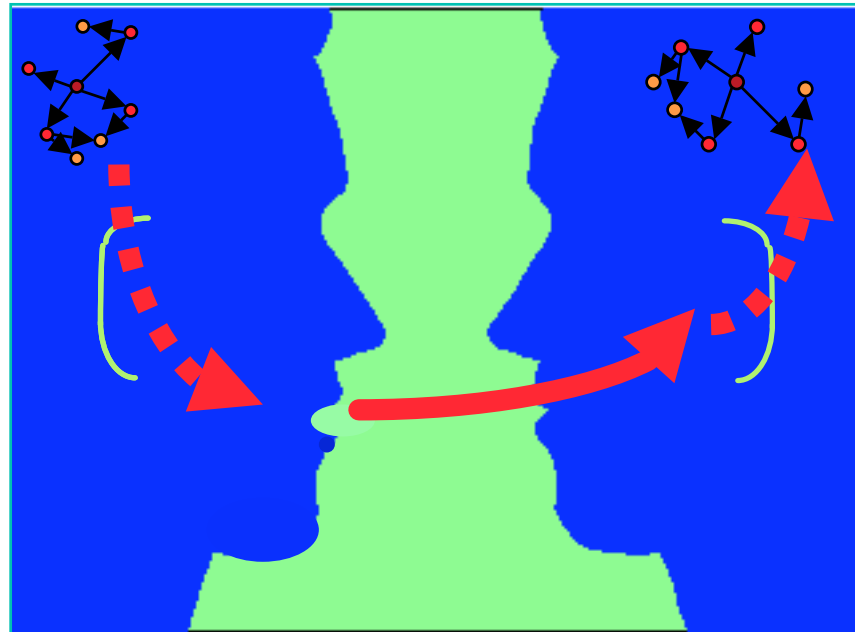


## Die Rolle der Grammatik, Ersetzungsregelgrammatiken, und Parsing

Hans Uszkoreit

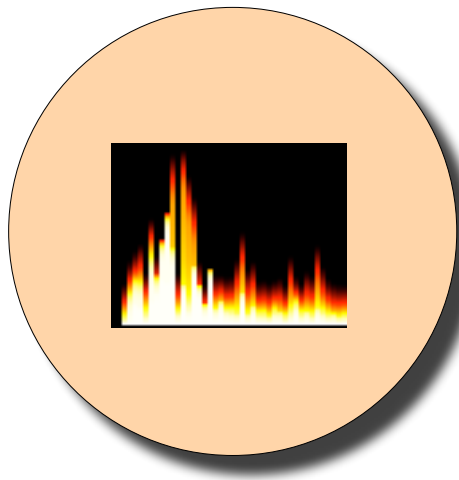
# WHAT HAPPENS IN BETWEEN?

---

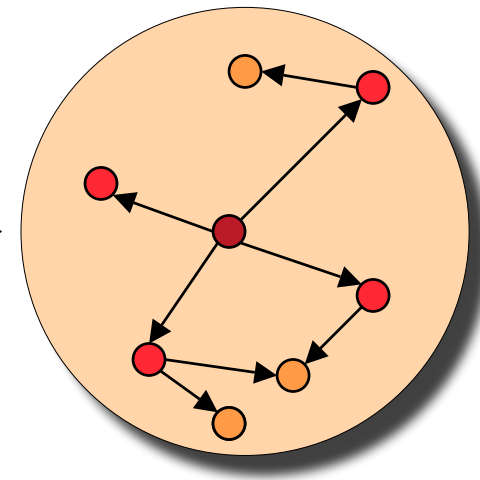


# WHAT HAPPENS IN BETWEEN?

---



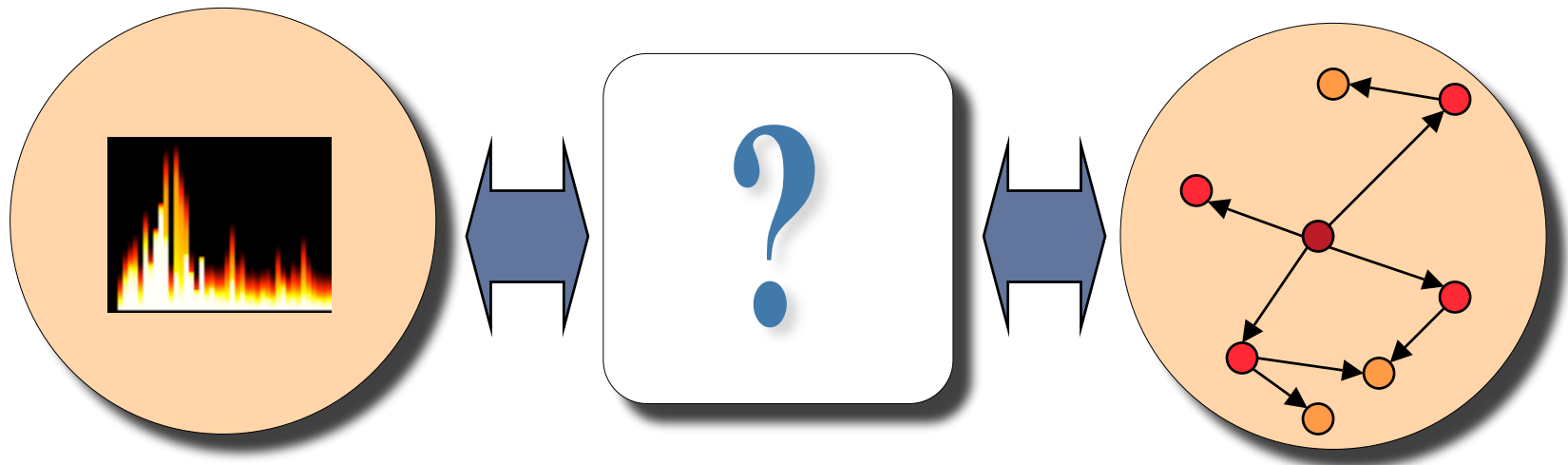
sound waves



activation of concepts

# WHAT HAPPENS IN BETWEEN?

---



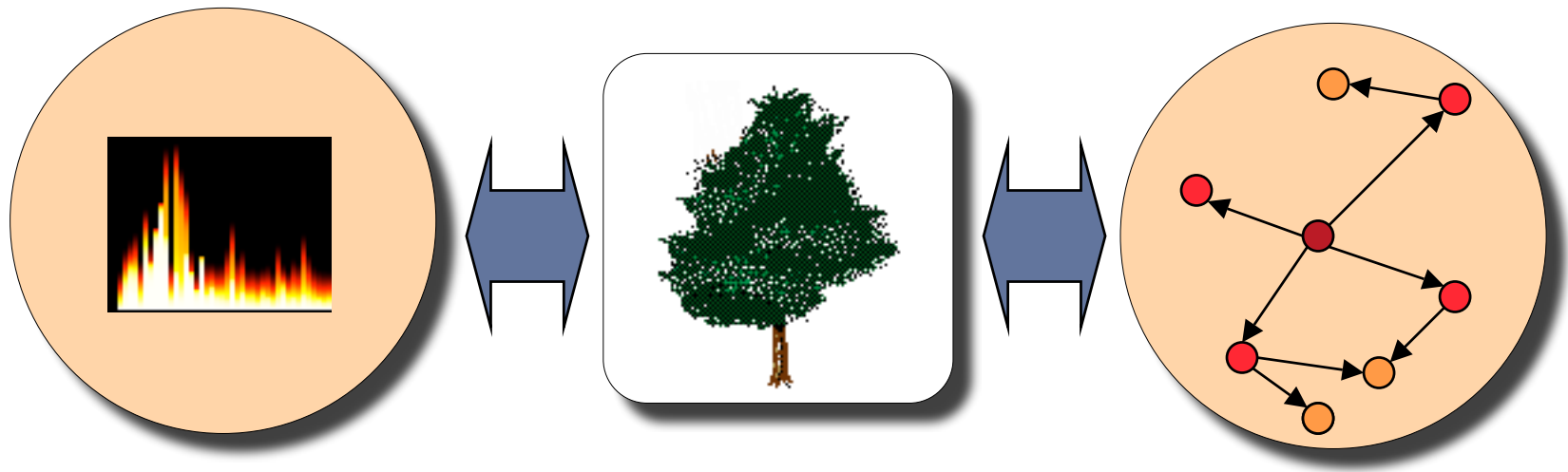
sound waves

Grammar

activation of concepts

# WHAT HAPPENS IN BETWEEN?

---



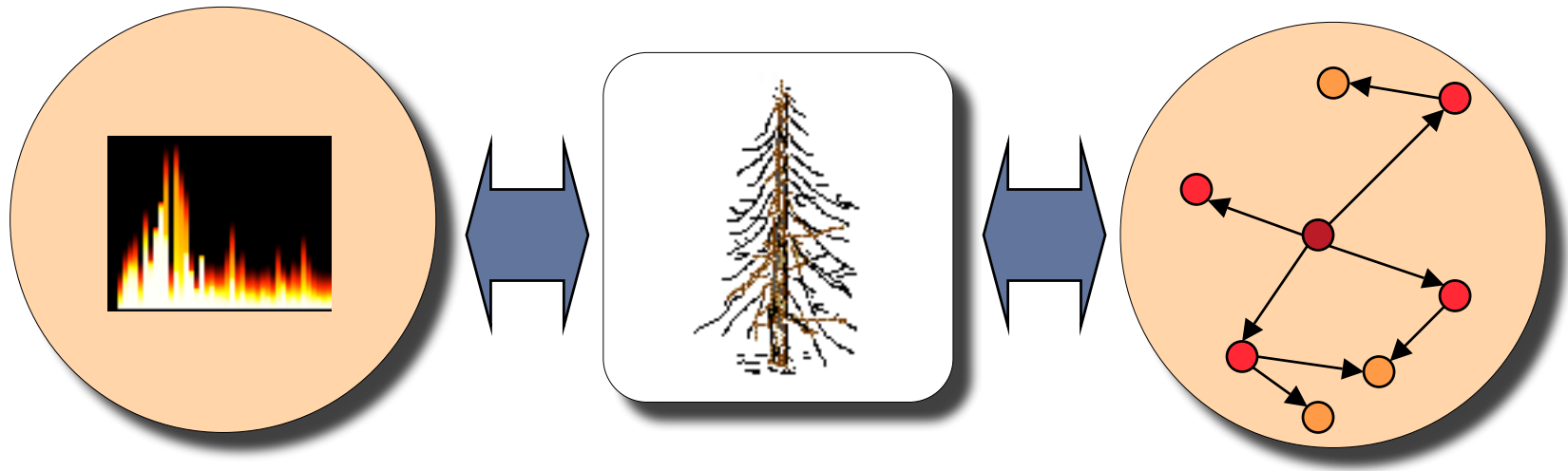
sound waves

Grammar

activation of concepts

# WHAT HAPPENS IN BETWEEN?

---

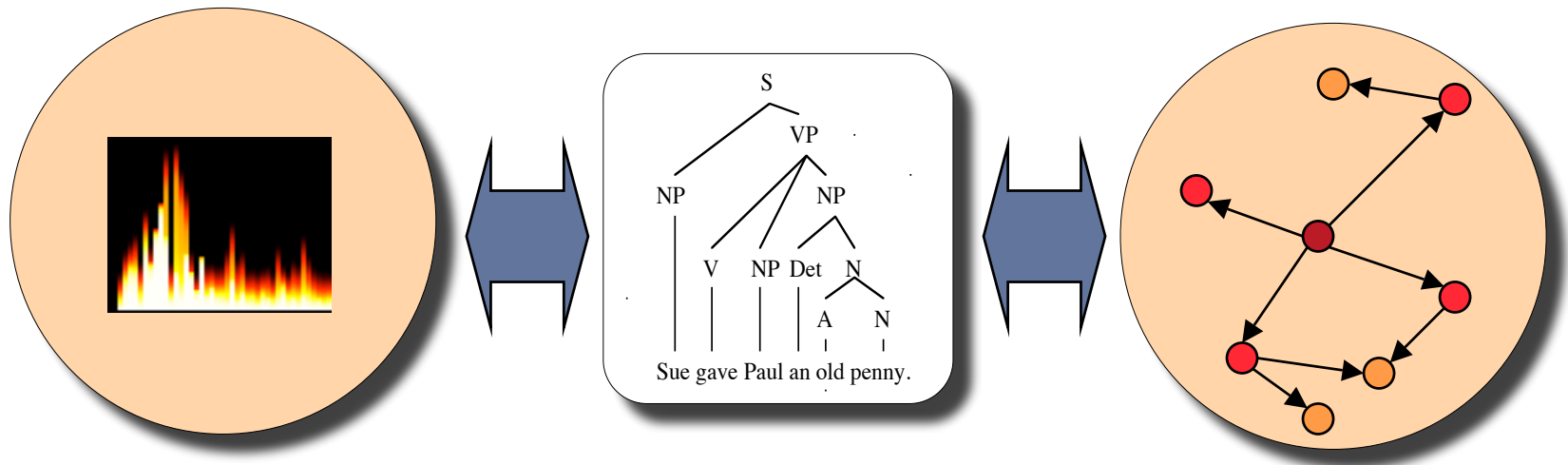


sound waves

Grammar

activation of concepts

# WHAT HAPPENS IN BETWEEN?

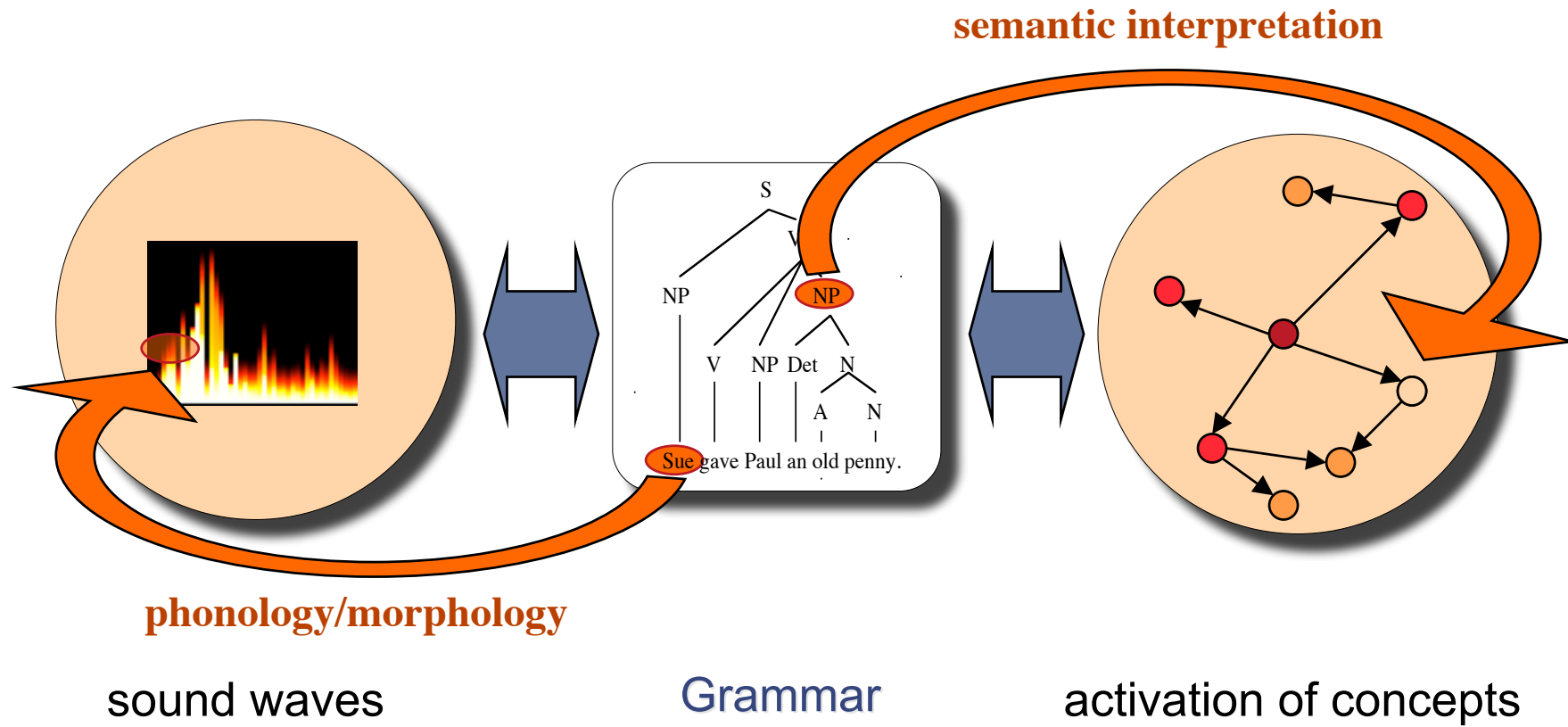


sound waves

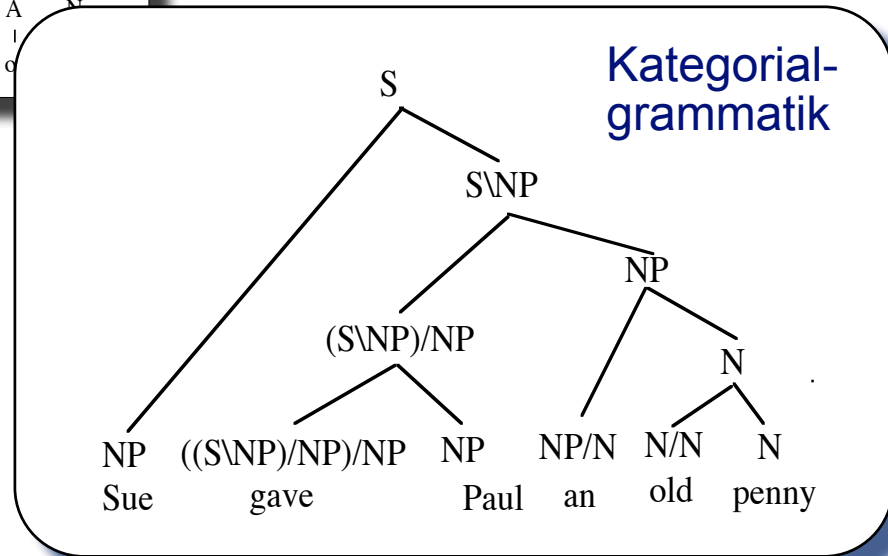
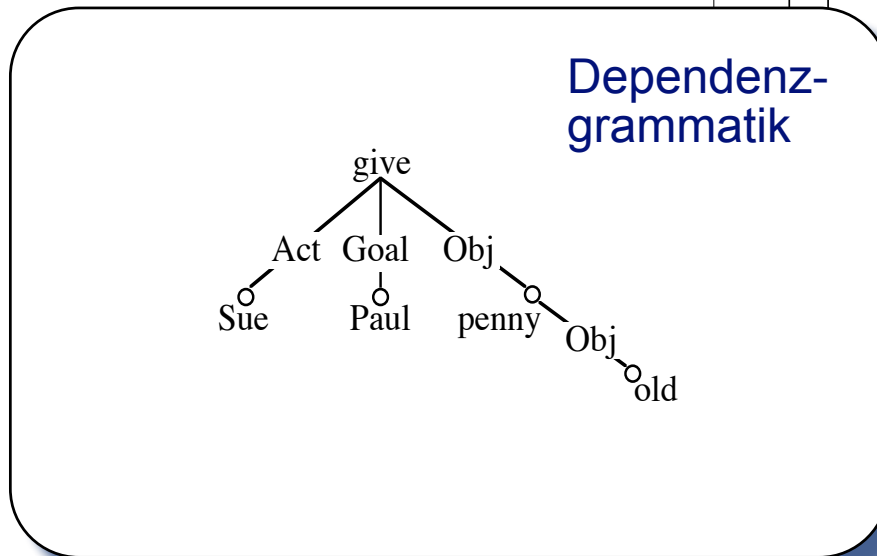
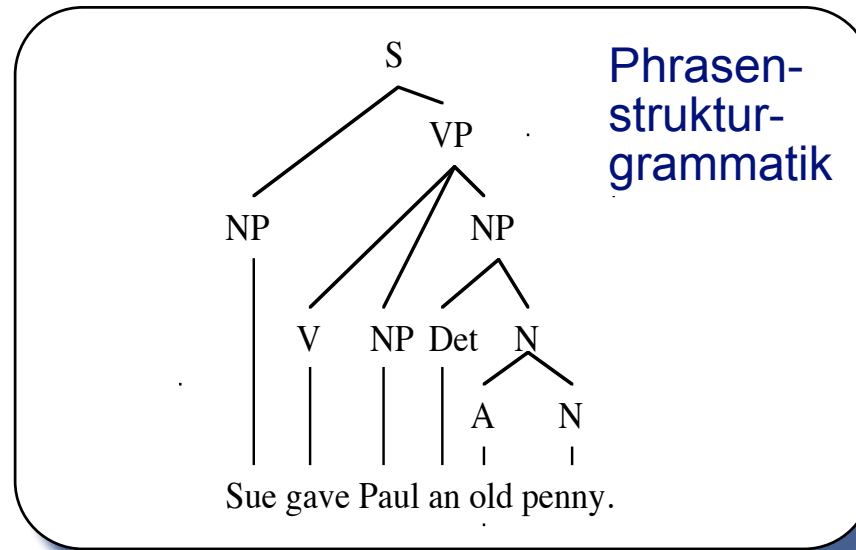
Grammar

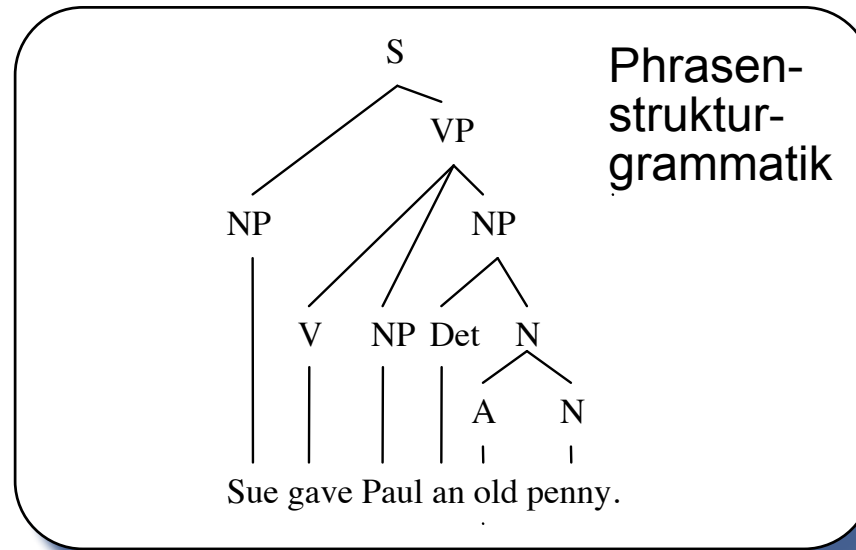
activation of concepts

# WHAT HAPPENS IN BETWEEN?

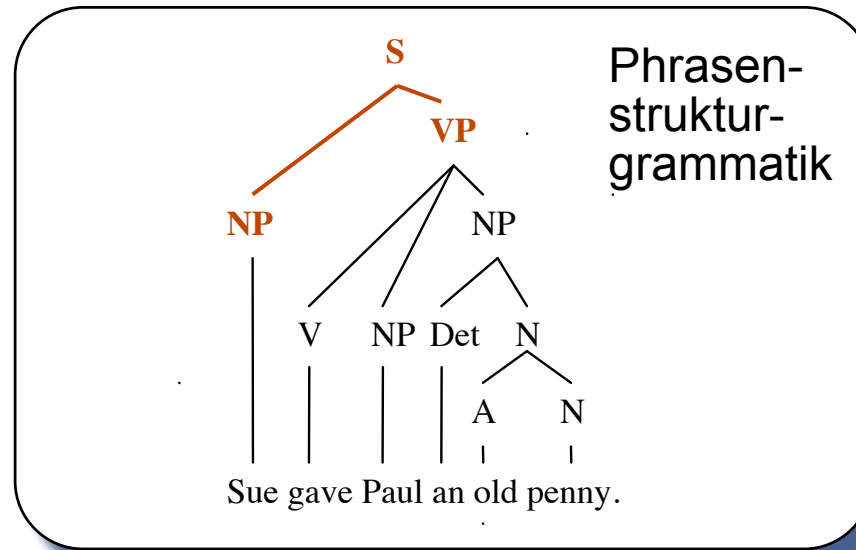


# THREE TRADITIONS

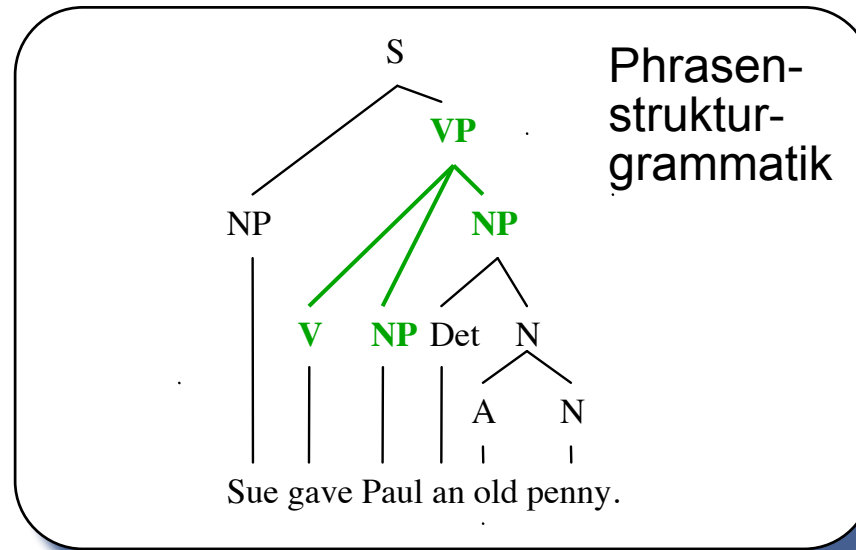




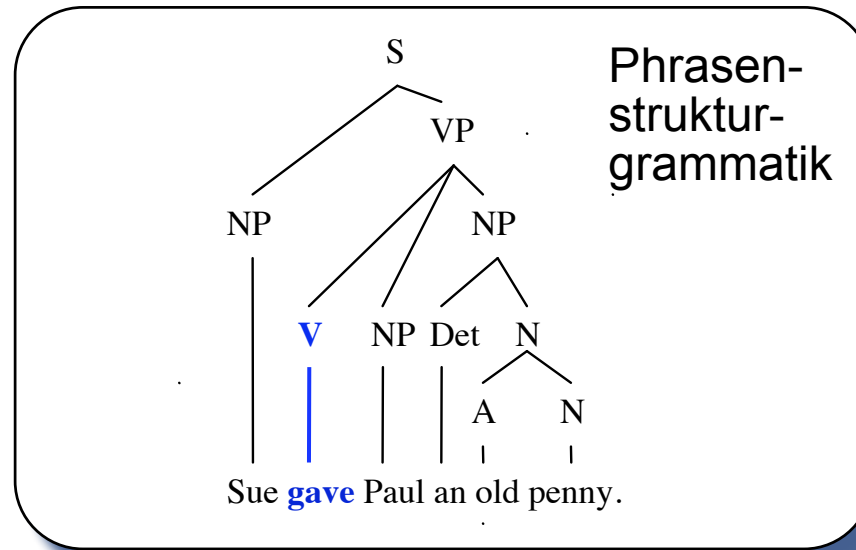
$S \rightarrow NP VP$



$S \rightarrow NP VP$



$S \rightarrow NP VP$   
 $VP \rightarrow V NP NP$



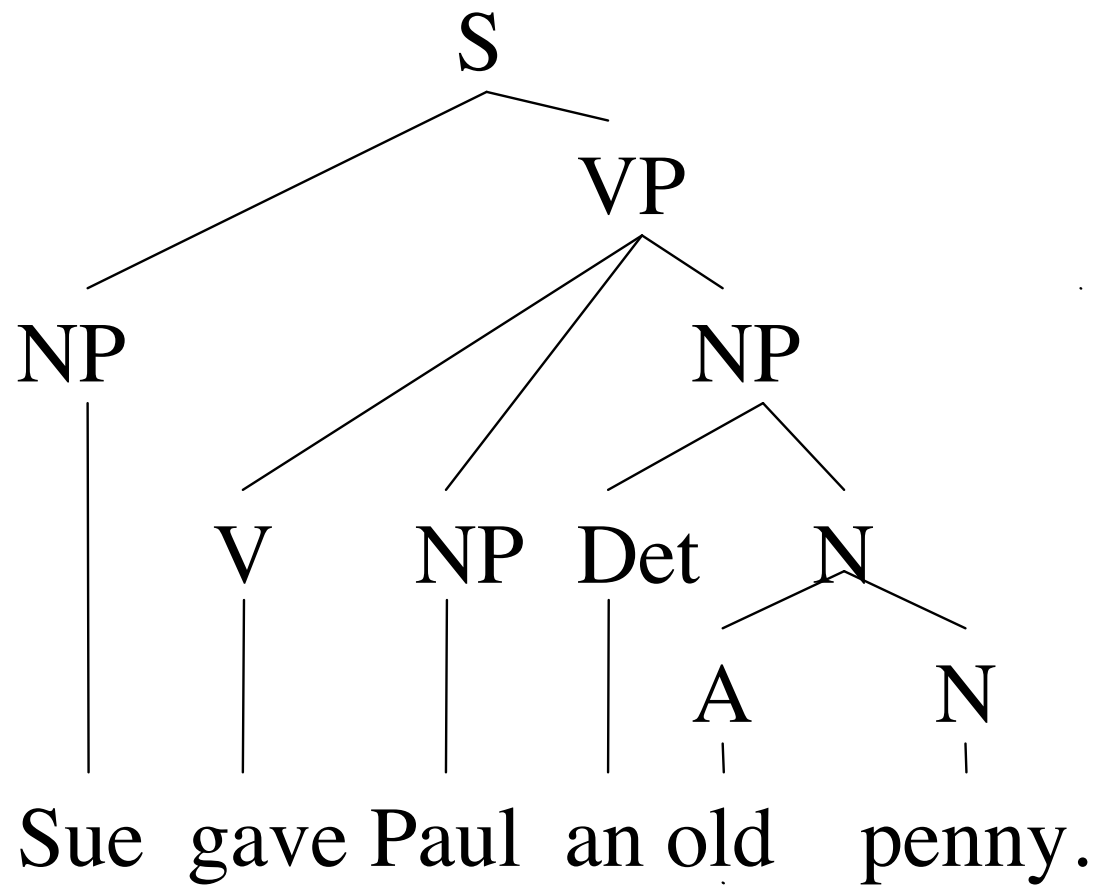
$S \rightarrow NP VP$   
 $VP \rightarrow V NP NP$

$V \rightarrow \text{gave}$

# Symbolkonventionen

---

	Einzelsymbole	Ketten
nichtterminale	A, B, C, ...	... , X, Y, Z
terminale	a, b, c, ...	... , x, y, z
unspezifiziert	$\alpha, \beta, \gamma, \dots$	... , $\phi, \chi, \psi, \omega$
Startsymbol	S	
die leere Kette	e	
Zahlen	..., i, j, k, l, m, n, ...	



# Formale Grammatik

---

Eine Sprache  $L$  über einem Alphabet (Vokabular)  $\Sigma$  ist eine Teilmenge von  $\Sigma^*$ .

Eine formale Grammatik  $G_L$  für eine Sprache  $L$  ist ein  
Quadrupel  $(V_N, V_T, \{S\}, P)$ .

$V_N$  - nichtterminales Vokabular, Hilfsalphabet

$V_T$  - terminales Vokabular

$$(V_T \cap V_N = \emptyset, \quad L \subseteq V_T^*, \quad V = V_T \cup V_N)$$

$\{S\}$  - Einermenge mit dem Startsymbol, Axiomenmenge

$P$  - Menge der Produktionen, Regelmenge

Menge von Regeln der Form  $\omega_1 \phi \omega_2 \rightarrow \omega_1 \psi \omega_2$

meist geschrieben als  $\phi \rightarrow \psi$

# Ableitung

---

## Relation "folgt":

Wenn  $G = (V_N, V_T, \{S\}, P)$ , dann folgt  $\psi$  aus  $\phi$  nach  $G$  genau dann, wenn es Ketten  $\phi_1, \phi_2, \chi, \omega$  gibt, so daß  $\phi = \phi_1 \chi \phi_2$  und  $\psi = \phi_1 \omega \phi_2$  und  $\chi \rightarrow \omega \in P$ .

Notation:  $\phi \underset{G}{\Rightarrow} \psi$

## Ableitung:

Eine Folge von Ketten  $\phi_1, \phi_1, \dots, \phi_n$  ist eine Ableitung nach  $G$  g.d.w.  $\phi_i \underset{G}{\Rightarrow} \phi_{i+1}$  für alle  $i, 1 \leq i \leq n$ .

Wenn es eine Ableitung nach  $G$  von  $\phi$  nach  $\psi$  gibt, können wir das

so schreiben:  $\phi \overset{*}{\underset{G}{\Rightarrow}} \psi$

Die Relation ableitbar ist transitiv und wird als reflexiv definiert.

## Die erzeugte Sprache

---

Die Sprache L: Eine Kette  $\omega$  ist nach  $G_L$  in der Sprache L g.d.w. die folgenden drei Bedingungen erfüllt sind:

1.  $\omega \in V_T^*$

2.  $S \xRightarrow[G]{*} \omega$

3. Es gibt kein  $\chi$ , so daß  $\omega \xRightarrow[G]{*} \chi$  und  $\omega \neq \chi$ .

Man sagt auch  $G_L$  erzeugt die Sprache L. Die von G erzeugte Sprache L wird auch als  $L(G)$  geschrieben.

Schwache Äquivalenz: Zwei Grammatiken  $G_1$  und  $G_2$  sind schwach äquivalent, wenn sie dieselbe Sprache erzeugen.

# Typen von Grammatiken

---

## Typ 0 (unbeschränkte Ersetzungsregelsysteme):

Jede Grammatik, die die Definition einer formalen Grammatik erfüllt, ist vom Typ 0.

## Typ 1 (kontextsensitive Grammatiken):

Jede Produktion hat die Form  $\phi A \psi \rightarrow \phi \omega \psi$ , wobei  $A \in V_T$ ,  $\omega \neq \varepsilon$ .

## Typ 2 (kontextfreie Grammatiken):

Jede Produktion hat die Form  $A \rightarrow \omega$ , wobei  $\omega \neq \varepsilon$ .

## Typ 3 (reguläre Grammatiken):

Jede Produktion hat die Form  $A \rightarrow x B$  oder  $A \rightarrow x$ , wobei  $x \neq \varepsilon$ .

# Grammars and Automata

---

## Formal grammars, formal languages and their corresponding automata

<b>Chomsky hierarchy</b>	<b>Grammars</b>	<b>Languages</b>	<b>Minimal Automaton</b>
Type-0	Unrestricted	Recursively enumerable	Turing machine
n/a	(no common name)	Recursive	Decider
Type-1	Context-sensitive	Context-sensitive	Linear-bounded Automaton
Type-2	Context-free	Context-free	Pushdown Automaton
Type-3	Regular	Regular	Finite-State Automaton

## Eine kurze Chronologie der Kontroverse

---

- 57-70: Chomsky, Postal u.a. behaupten, daß KFGs are unzureichend für natürliche Sprachen
- 63: Gilbert Harrman verteidigt Phrasenstrukturgrammatik
- 71&73: Peters und Ritchie zeigen, daß TG alle Sprachen vom Typ 0 erzeugt
- 79: Gazdar schlägt eine Zwei-Stufen PSG (GPSG) vor, die adäquat für das Englische ist
- 82: Gazdar & Pullum zeigen, daß alle bisherigen "Beweise" für die Nichtkontextfreiheit fehlerhaft sind
- 82: Uszkoreit und Peters beweisen, daß die damalige GPSG alle Sprachen vom Typ 0 erzeugt, dieser Umstand wird von GKPS durch den finite-closure constraint beseitigt
- 85: Shieber beweist, daß Zürcher Deutsch nicht KF ist, Culy zeigt das gleiche für Bambara
- 85: ähnliche Behauptungen von Higginbotham and Langendoen & Postal für das Englische wurden durch Pullum erfolgreich widerlegt

---

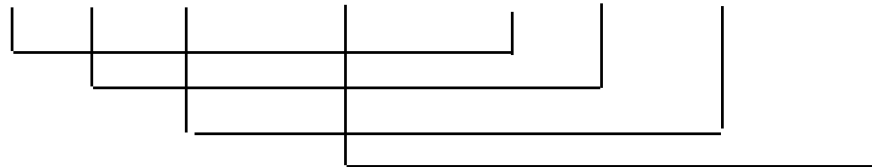
Jan säit das mer em Hans es huus hälfed aastriiche



---

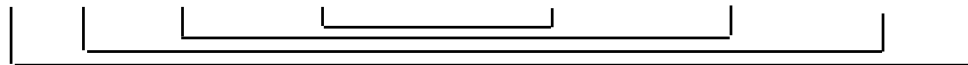
...dat Jan Piet Marie de kinderen zag helpen laten zwemmen

daß Jan Piet Marie den Kindern sah helfen lassen schwimmen



daß Jan Piet Marie helfen sah, die Kinder schwimmen zu lassen

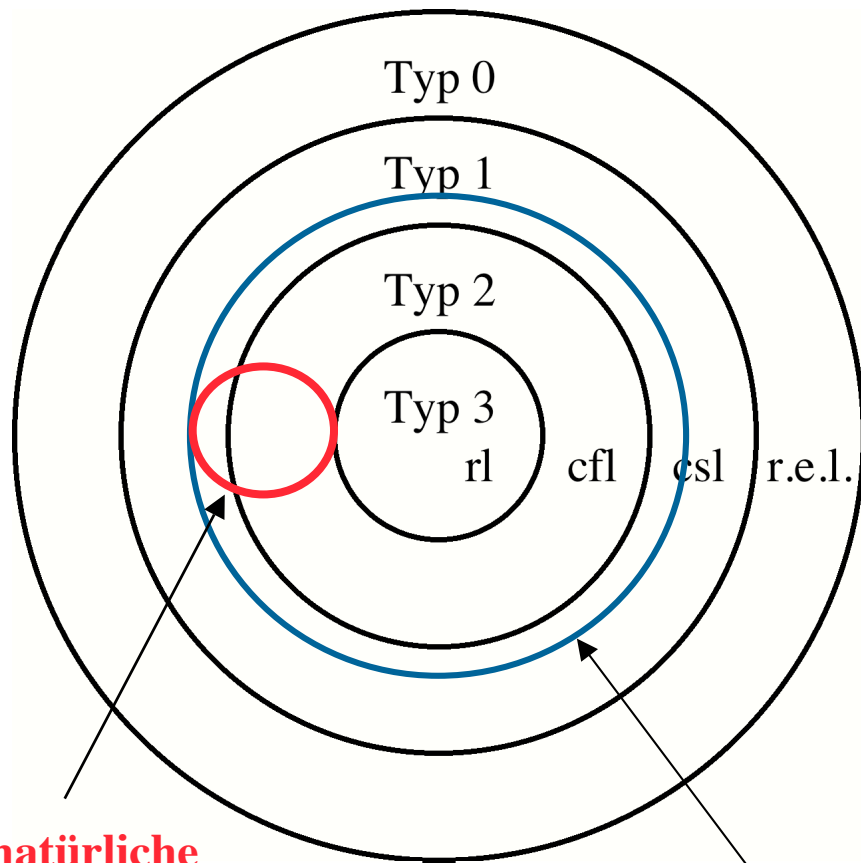
daß Jan Piet Marie die Kinder schwimmen zu lassen helfen sah



Jan säit das mer em Hans es huus hälfed aastriche



# Computational Complexity of Human Language



**natürliche  
Sprachen**

**mildly context-sensitive languages**

- Typ 0: recursively enumerable sets
- Typ 1: contextsensitive languages
- Typ 2: context-free languages
- Typ 3: regular languages

# Ableitung

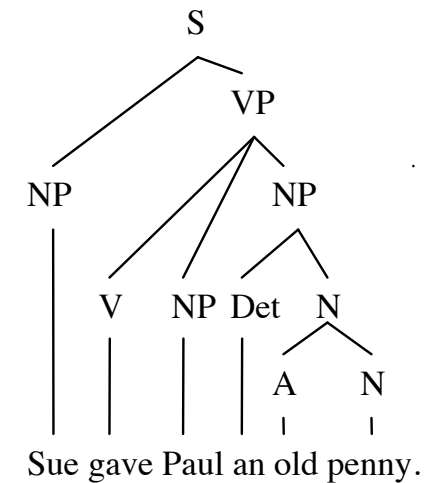
---

S  
NP VP  
DET ADJ N VP  
DET ADJ N V NP  
DET ADJ N V DET ADJ N  
.  
.  
.  
ein kleines Mädchen sucht ein kleines Mädchen

## der Begriff des syntaktischen Strukturbaums

kodierte Information

1. Die hierarchische Organisation der Teile eines Satzes in Konstituenten.
2. Der Zugehörigkeit jeder Konstituente in eine syntaktische (linguistische) Klasse (Kategorie).
3. Die lineare Abfolge der Konstituenten.



## Bäume II

---

**Relationen:** unmittelbare Dominanz - Dominanz  
unmittelbare Präzedenz - Präzedenz

**Konstituentenstrukturbaum: Quintupel** (N, Q, D, P, L)

N - endliche Menge von Knoten

Q - endliche Menge von Etiketten

D - schwache Teilordnung in  $N \times N$ , die Dominanzrelation  
(reflexiv, transitiv und antisymmetrisch)

P - starke Teilordnung in  $N \times N$ , die Präzedenzrelation  
(irreflexiv, transitiv und asymmetrisch)

L - Funktion von N in Q, die Etikettierfunktion

**Bedingungen:**

Wurzelbedingung

Exklusivitätsbedingung

Kreuzungsfreiheit

## Wurzelbedingung

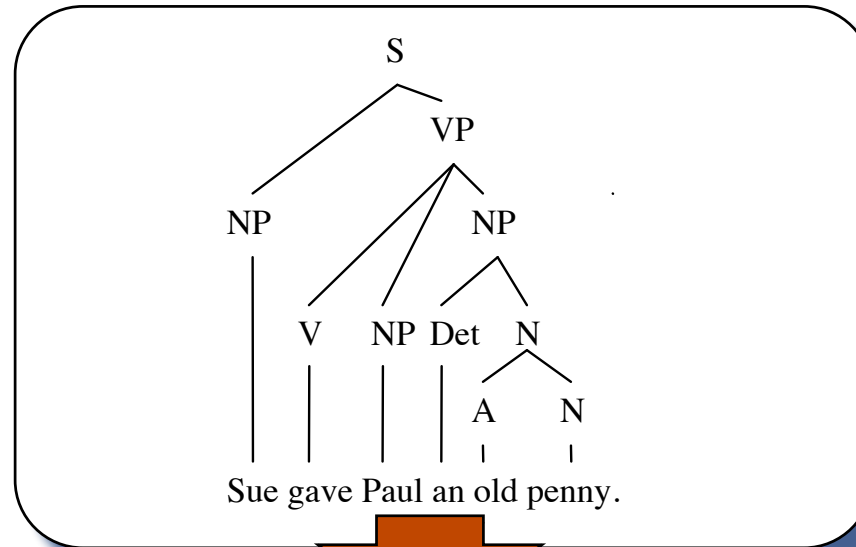
Es gibt genau einen Knoten, der alle Knoten des Baumes dominiert.

## Exklusivitätsbedingung

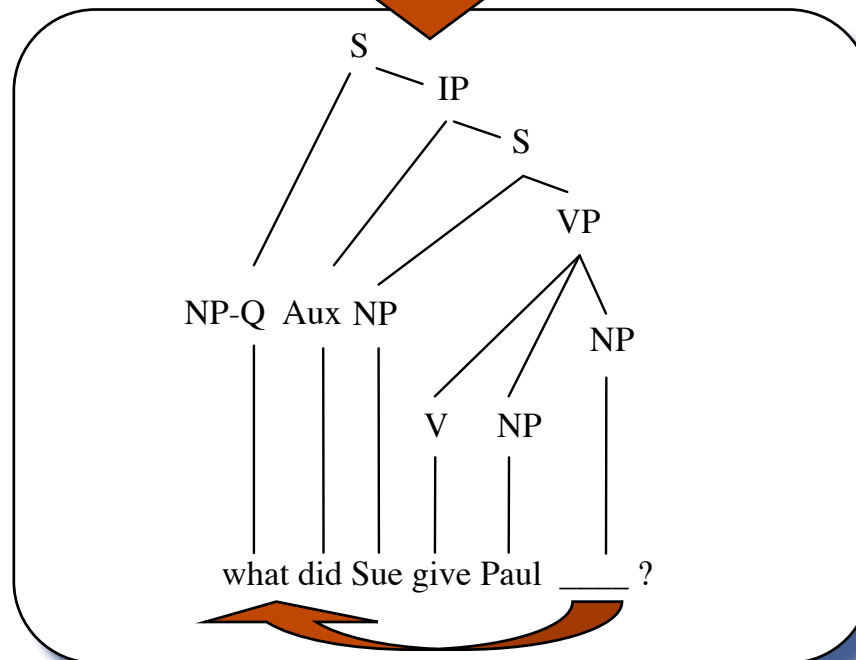
Für jegliche zwei Knoten  $x$  und  $y$  gilt, entweder  $D(x,y)$  oder  $D(y,x)$   
oder  $P(x,y)$  oder  $P(y,x)$

## Kreuzungsfreiheit

Wenn  $P(x,y)$  dann gilt für alle  $x'$ , die von  $x$  dominiert werden  $D(x,x')$  und für alle  $y'$ , die von  $y$  dominiert werden  $D(y, y')$  dass auch  $x' y'$  vorausgeht  
[ $P(x', y')$ ]



Transformations-  
grammatik



# Grammatische Verarbeitung

---

- Je nach Verarbeitungsrichtung sprechen wir von

**grammatischer Analyse,**

wenn wir sprachliche Eingabe mithilfe einer Grammatik analysieren,  
oder von

**grammatischer Generierung,**

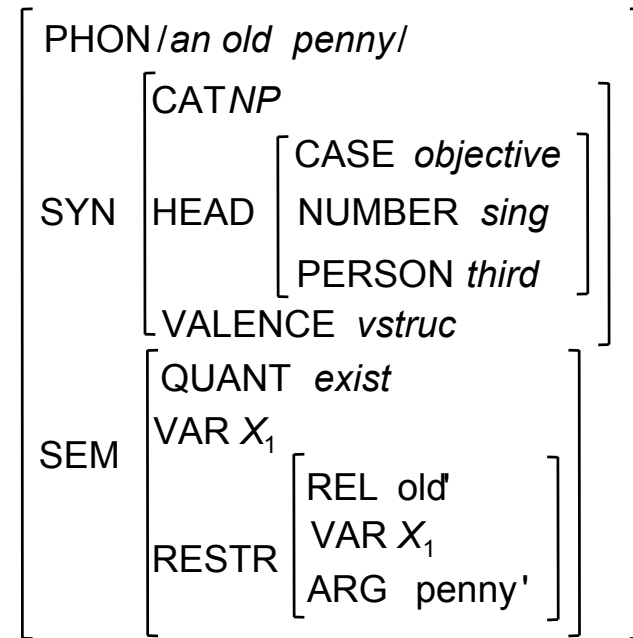
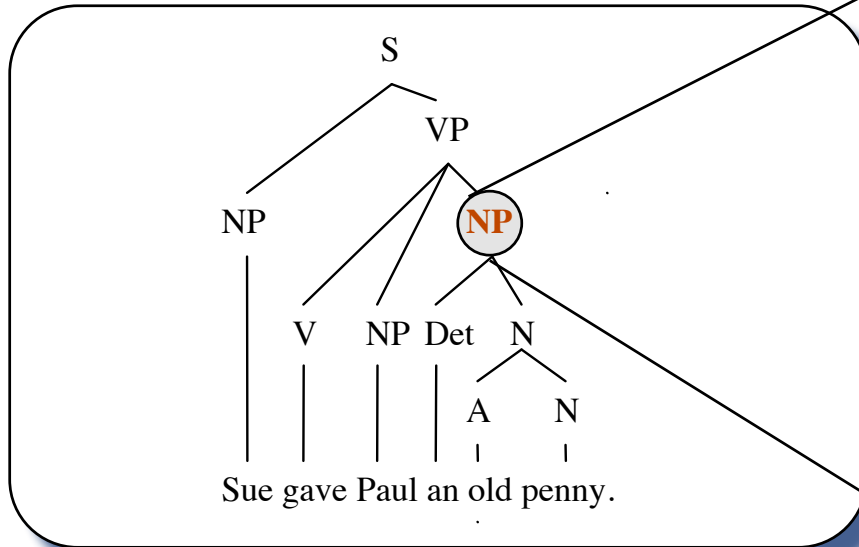
wenn wir mithilfe einer Grammatik aus eine Bedeutungsrepräsentation  
eine sprachliche Ausgabe erzeugen.

## Warum syntaktische Analyse?

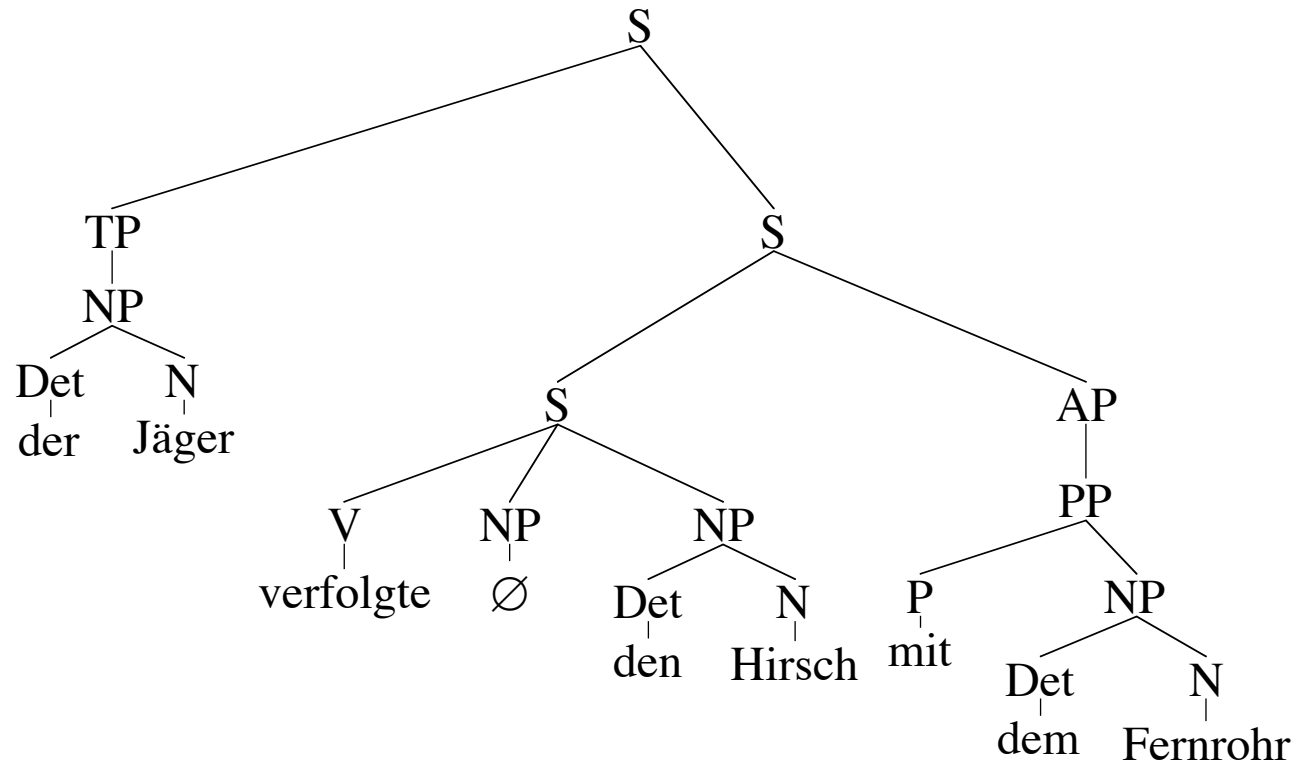
---

- Einen Hund hat dieser Mann gebissen.
- Ein Hund hat diesen Mann gebissen.
- This man has bitten a dog.
- A dog this man has bitten.
- A dog has bitten this man.
- Peter versprach Paul, die Akten zu bearbeiten.
- Peter überredete Paul, die Akten zu bearbeiten

## Unifikationsgrammatik



# Strukturbaum



# Bottom-Up Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

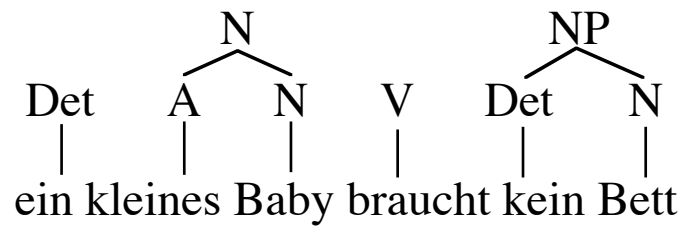
$A \rightarrow \text{kleines}$   
 $A \rightarrow \text{schönes}$   
 $A \rightarrow \text{warmes}$   
 $Det \rightarrow \text{ein}$   
 $Det \rightarrow \text{kein}$   
 $Det \rightarrow \text{sein}$   
 $N \rightarrow \text{Baby}$   
 $N \rightarrow \text{Bett}$   
 $N \rightarrow \text{Kissen}$   
 $P \rightarrow \text{ohne}$   
 $V \rightarrow \text{braucht}$   
 $V \rightarrow \text{hat}$

Det    A    N    V    Det    N  
|    |    |    |    |    |  
ein kleines Baby braucht kein Bett

# Bottom-Up Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

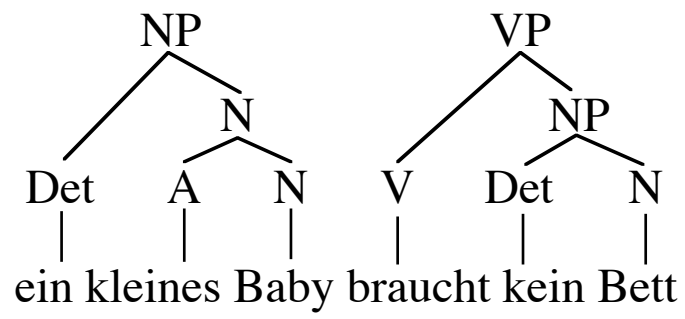
$A \rightarrow$  kleines  
 $A \rightarrow$  schönes  
 $A \rightarrow$  warmes  
 $Det \rightarrow$  ein  
 $Det \rightarrow$  kein  
 $Det \rightarrow$  sein  
 $N \rightarrow$  Baby  
 $N \rightarrow$  Bett  
 $N \rightarrow$  Kissen  
 $P \rightarrow$  ohne  
 $V \rightarrow$  braucht  
 $V \rightarrow$  hat



# Bottom-Up Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

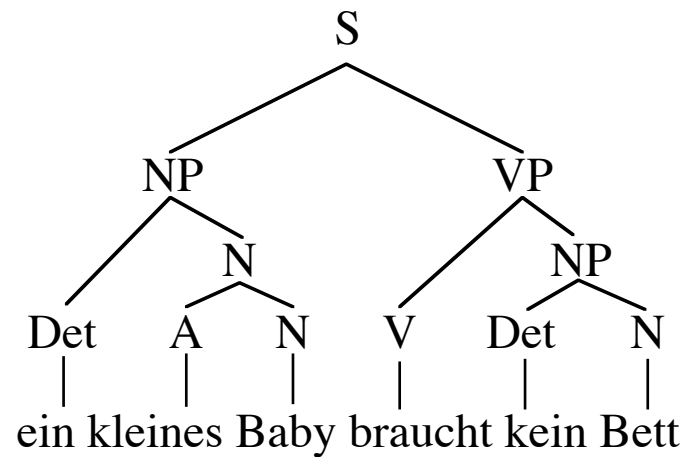
$A \rightarrow$  kleines  
 $A \rightarrow$  schönes  
 $A \rightarrow$  warmes  
 $Det \rightarrow$  ein  
 $Det \rightarrow$  kein  
 $Det \rightarrow$  sein  
 $N \rightarrow$  Baby  
 $N \rightarrow$  Bett  
 $N \rightarrow$  Kissen  
 $P \rightarrow$  ohne  
 $V \rightarrow$  braucht  
 $V \rightarrow$  hat



# Bottom-Up Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

$A \rightarrow$  kleines  
 $A \rightarrow$  schönes  
 $A \rightarrow$  warmes  
 $Det \rightarrow$  ein  
 $Det \rightarrow$  kein  
 $Det \rightarrow$  sein  
 $N \rightarrow$  Baby  
 $N \rightarrow$  Bett  
 $N \rightarrow$  Kissen  
 $P \rightarrow$  ohne  
 $V \rightarrow$  braucht  
 $V \rightarrow$  hat



# Top-Down Parsing

---

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

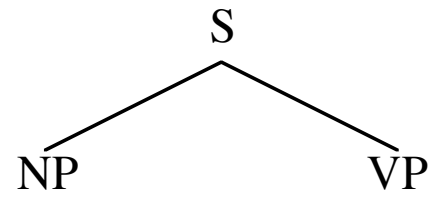
$A \rightarrow \text{kleines}$   
 $A \rightarrow \text{schönes}$   
 $A \rightarrow \text{warmes}$   
 $Det \rightarrow \text{ein}$   
 $Det \rightarrow \text{kein}$   
 $Det \rightarrow \text{sein}$   
 $N \rightarrow \text{Baby}$   
 $N \rightarrow \text{Bett}$   
 $N \rightarrow \text{Kissen}$   
 $P \rightarrow \text{ohne}$   
 $V \rightarrow \text{braucht}$   
 $V \rightarrow \text{hat}$

ein kleines Baby braucht kein Bett

# Top-Down Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

$A \rightarrow$  kleines  
 $A \rightarrow$  schönes  
 $A \rightarrow$  warmes  
 $Det \rightarrow$  ein  
 $Det \rightarrow$  kein  
 $Det \rightarrow$  sein  
 $N \rightarrow$  Baby  
 $N \rightarrow$  Bett  
 $N \rightarrow$  Kissen  
 $P \rightarrow$  ohne  
 $V \rightarrow$  braucht  
 $V \rightarrow$  hat

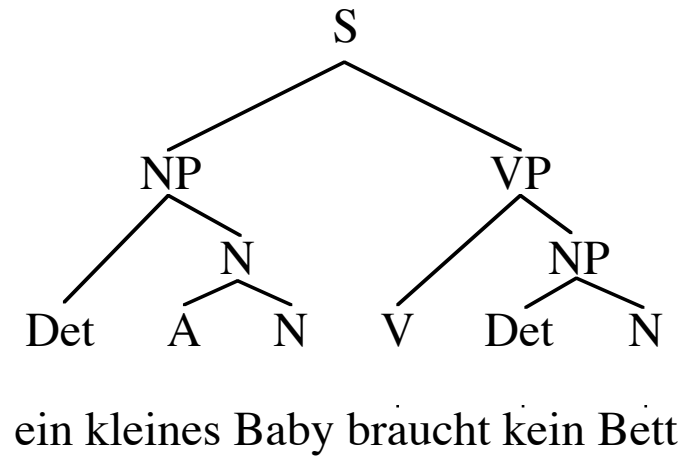


ein kleines Baby braucht kein Bett

# Top-Down Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

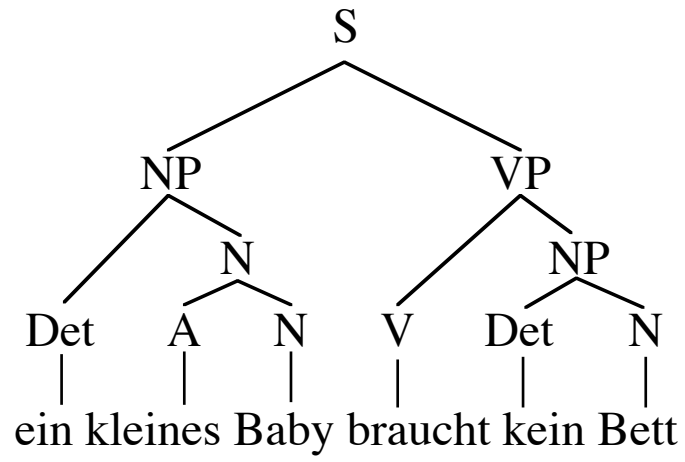
$A \rightarrow$  kleines  
 $A \rightarrow$  schönes  
 $A \rightarrow$  warmes  
 $Det \rightarrow$  ein  
 $Det \rightarrow$  kein  
 $Det \rightarrow$  sein  
 $N \rightarrow$  Baby  
 $N \rightarrow$  Bett  
 $N \rightarrow$  Kissen  
 $P \rightarrow$  ohne  
 $V \rightarrow$  braucht  
 $V \rightarrow$  hat



# Top-Down Parsing

$S \rightarrow NP VP$   
 $NP \rightarrow Det N$   
 $N \rightarrow A N$   
 $N \rightarrow N PP$   
 $VP \rightarrow V NP$

$A \rightarrow$  kleines  
 $A \rightarrow$  schönes  
 $A \rightarrow$  warmes  
 $Det \rightarrow$  ein  
 $Det \rightarrow$  kein  
 $Det \rightarrow$  sein  
 $N \rightarrow$  Baby  
 $N \rightarrow$  Bett  
 $N \rightarrow$  Kissen  
 $P \rightarrow$  ohne  
 $V \rightarrow$  braucht  
 $V \rightarrow$  hat



## Beispiel

---

S → NP VP	Det → der	Det → den	Det → das	
NP → Det N	N → Junge	N → Ball	N → Mädchen	
N → N PP	PN → Berlin	PN → Peter	PN → Gerda	PN → Bonn
NP → PN	P → aus	P → für	P → ohne	P → nach
VP → V	V → gab	V → sah	V → schlief	V → stahl
VP → V NP				
VP → V NP NP				
PP → P NP				

## Beispiel

---

S → NP VP

NP → Det N

N → N PP

NP → PN

VP → V

VP → V NP

Det → der

N → Junge

PN → Berlin

P → aus

V → gab

Det → den

N → Ball

PN → Peter

P → für

V → sah

Det → das

N → Mädchen

PN → Gerda

P → ohne

V → schlief

PN → Bonn

P → nach

V → stahl

Der Junge aus Berlin gab Peter den Ball.

## Beispiel: Lokale Ambiguität

S → NP VP

NP → Det N

N → N PP

NP → PN

VP → V

VP → V NP

VP → V NP NP

PP → P NP

Det → der

N → Junge

PN → Berlin

P → aus

V → gab

Det → den

N → Ball

PN → Peter

P → für

V → sah

Det → das

N → Mädchen

PN → Gerda

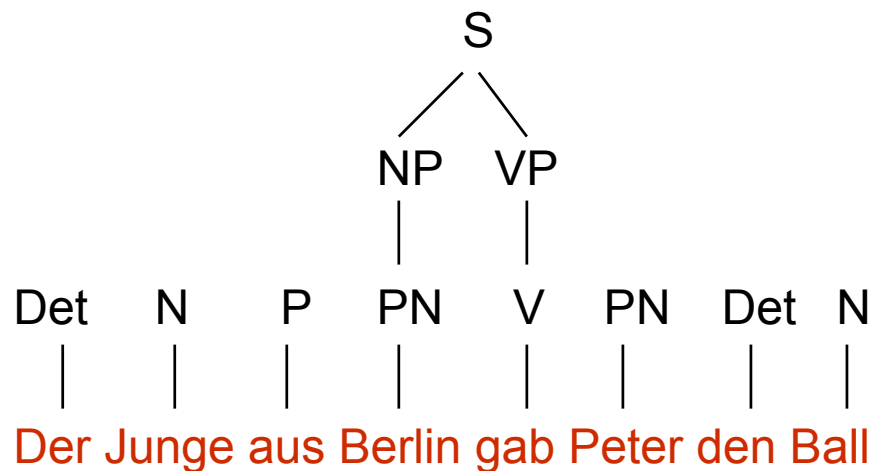
P → ohne

V → schlief

PN → Bonn

P → nach

V → stahl



# Earley Algorithmus

---

- 1970 für das Parsen mit kontextfreien Grammatiken von Jay Earley entwickelt.
- Der Algorithmus verknüpft top-down Vorhersage mit bottom-up Überprüfung.
- Als ein Chart-Parser verwendet der Algorithmus eine Tabelle mit erkannten wohlgeformten Teilketten (well-formed substring table).
- Jede wohlgeformte Zeichenkette wird nur einmal gemerkt.
- Der Algorithmus besteht aus den Teilen: (Initialisierung), Vorhersage, Vervollständigung und Einlesen.

## **Parsing: Chart Parsing, Earley-Algorithm, top down/bottom up, left-right**

### **Permanent data structure:**

grammar (context free phrase structure grammar)

**Input:** a string of words  $w_1 w_2 \dots w_n$

### **Working structures:**

Input string - input with position markers  $\boxed{0} w_1 \boxed{1} w_2 \boxed{2} \dots w_n \boxed{n}$

Position – point in the input string

Chart - set of items (edges) Every item is a triple  $\langle h, i, A \rightarrow \alpha.\beta \rangle$ , where  $h$  and  $i$  are positions of the input string, and  $A \rightarrow \alpha.\beta$  a dotted rule.  $\alpha$  and  $\beta$  are members of  $(V_T \cup V_N)^*$ .

$A \rightarrow \alpha \beta$  is a rule of the grammar.  $\alpha$  is the part of the right-hand-side of the rule that has already been applied to the segment of the input string from  $h$  to  $i$ .

Initially the chart contains for each rule of the form  $S \rightarrow \alpha$  exactly one item  $\langle 0, 0, S \rightarrow .\alpha \rangle$ . (initialization)

## Algorithm:

For each position  $i$  from 0 to  $n$  repeat the following steps until no more items can be produced:

Scanner (consumption of input)

If  $w_i = a$  add for each item  $\langle h, i-1, A \rightarrow \alpha.a\beta \rangle$   
a new item  $\langle h, i, A \rightarrow \alpha.a.\beta \rangle$ .

Completer (completion of constituents)

For each pair of items of the form  $\langle h, i, A \rightarrow \alpha. \rangle$  and  
 $\langle k, h, B \rightarrow \beta.A\gamma \rangle$  add a new entry  $\langle k, i, B \rightarrow \beta A.\gamma \rangle$  to the  
chart if it is not already present in the chart.

Predictor (top down prediction of constituents)

For each item  $\langle h, i, A \rightarrow \alpha.B\beta \rangle$  add for each rule  $B \rightarrow \gamma$   
a new item  $\langle i, i, B \rightarrow .\gamma \rangle$  to the chart if it is not already  
present in the chart.

If the chart contains at least one item  $\langle 0, n, S \rightarrow \alpha. \rangle$  return  
**success**, else return **failure**.



<i>i</i>	word <sub>i</sub>	new items	step
3	baby	<ul style="list-style-type: none"> <li>• &lt;2,3, N → baby.&gt;</li> <li>• &lt;1,3, N → A N.&gt;</li> <li>• &lt;0,3, NP → Det N.&gt;</li> <li>• &lt;0,3, S → NP.VP&gt;</li> <li>• &lt;3,3, VP → .V NP&gt;</li> <li>• &lt;3,3, VP → .V NP PP&gt;</li> <li>• &lt;3,3, V → .needs</li> <li>• &lt;3,3, V → .has</li> <li>• &lt;3,3, V → .sees</li> </ul>	Scanner Completer Completer Completer Predictor Predictor Predictor Predictor Predictor
4	needs	<ul style="list-style-type: none"> <li>• &lt;3,4, V → needs.&gt;</li> <li>• &lt;3,4, VP → V.NP&gt;</li> <li>• &lt;3,4, VP → V.NP PP&gt;</li> <li>• &lt;4,4, NP → .Det N&gt;</li> <li>• &lt;4,4, Det → .the&gt;</li> <li>• &lt;4,4, Det → .a&gt;</li> <li>• &lt;4,4, Det → .his&gt;</li> </ul>	Scanner Completer Completer Predictor Predictor Predictor Predictor
5	a	<ul style="list-style-type: none"> <li>• &lt;4,5, Det → a.&gt;</li> <li>• &lt;4,5, NP → Det.N&gt;</li> <li>• &lt;5,5, N → .A N&gt;</li> <li>• &lt;5,5, A → .little&gt;</li> <li>• &lt;5,5, A → .nice&gt;</li> <li>• &lt;5,5, A → .warm&gt;</li> <li>• &lt;5,5, N → .baby&gt;</li> <li>• &lt;5,5, N → .bed&gt;</li> <li>• &lt;5,5, N → .pillow&gt;</li> </ul>	Scanner Completer Predictor Predictor Predictor Predictor Predictor Predictor Predictor
6	bed	<ul style="list-style-type: none"> <li>• &lt;5,6, N → bed.&gt;</li> <li>• &lt;4,6, NP → Det N.&gt;</li> <li>• &lt;3,6, VP → V NP.&gt;</li> <li>• &lt;3,6, VP → V NP.PP&gt;</li> <li>• &lt;0,6, S → NP VP.&gt;</li> <li>• &lt;6,6, PP → .P NP&gt;</li> <li>• &lt;6,6, P → .without&gt;</li> <li>• &lt;6,6, P → .for&gt;</li> </ul>	Scanner Completer Completer Completer Completer Predictor Predictor Predictor

# Initialisierung

---

Anfangs enthält die Chart für jede Regel  $S \rightarrow \alpha$  genau einen Eintrag  
 $\langle 0, 0, S \rightarrow . \alpha \rangle$ .

*Position = 0*

Für  $S \rightarrow NP VP$  enthält die Chart den folgenden ersten Eintrag:  
 $\langle 0, 0, S \rightarrow .NP VP \rangle$

S  


① Der ② Junge ③ aus ④ Berlin ⑤ gab ⑥ Peter ⑦ den ⑧ Ball ⑨

# Predictor

---

## Vorhersage (Predictor)

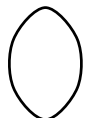
Für jeden Eintrag  $\langle h, i, A \rightarrow \alpha.B \beta \rangle$  füge für jede Regel  $B \rightarrow \gamma$  den Eintrag  $\langle i, i, B \rightarrow .\gamma \rangle$  in die Chart ein, wenn er nicht schon in der Chart steht.

*Position = 0*

Für  $\langle 0, 0, S \rightarrow .NP VP \rangle$  und  $NP \rightarrow Det N$  füge ein:  $\langle 0, 0, NP \rightarrow .Det N \rangle$

S  


① Der ① Junge ② aus ③ Berlin ④ gab ⑤ Peter ⑥ den ⑦ Ball ⑧



NP

# Predictor

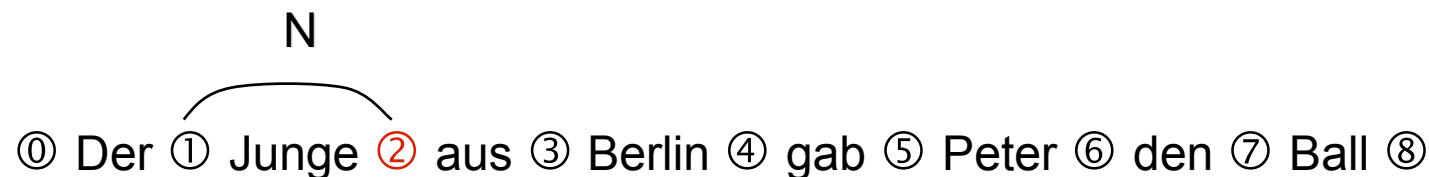
---

## Vorhersage (Predictor)

Für jeden Eintrag  $\langle h, i, A \rightarrow \alpha.B \beta \rangle$  füge für jede Regel  $B \rightarrow \gamma$  den Eintrag  $\langle i, i, B \rightarrow .\gamma \rangle$  in die Chart ein, wenn er nicht schon in der Chart steht.

*Position = 2*

Für  $\langle 1, 2, N \rightarrow N.PP \rangle$  und  $PP \rightarrow P NP$  füge ein:  $\langle 2, 2, PP \rightarrow .P NP \rangle$



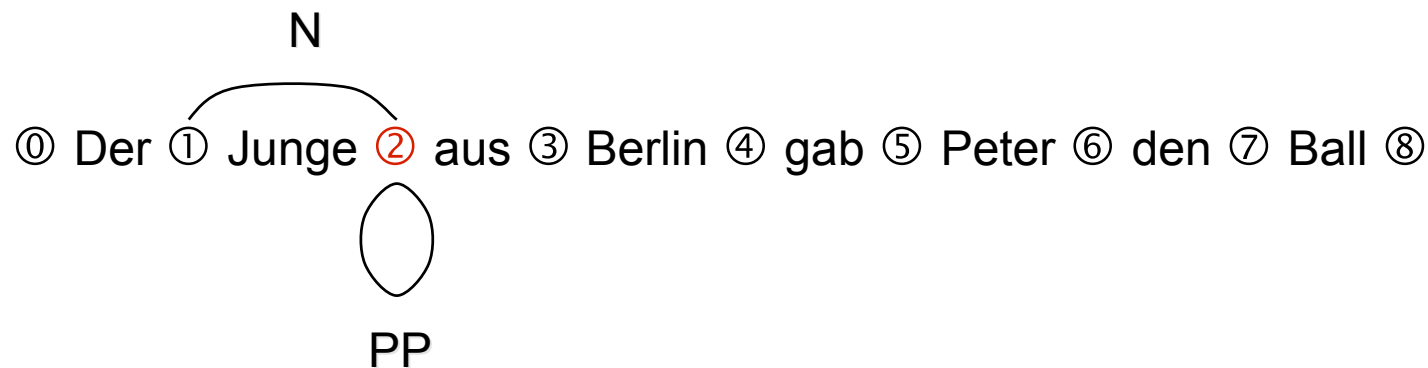
# Predictor

## Vorhersage (Predictor)

Für jeden Eintrag  $\langle h, i, A \rightarrow \alpha.B \beta \rangle$  füge für jede Regel  $B \rightarrow \gamma$  den Eintrag  $\langle i, i, B \rightarrow .\gamma \rangle$  in die Chart ein, wenn er nicht schon in der Chart steht.

*Position = 2*

**Für**  $\langle 1, 2, N \rightarrow N.PP \rangle$  **und**  $PP \rightarrow P NP$  **füge ein:**  $\langle 2, 2, PP \rightarrow .P NP \rangle$



# Predictor

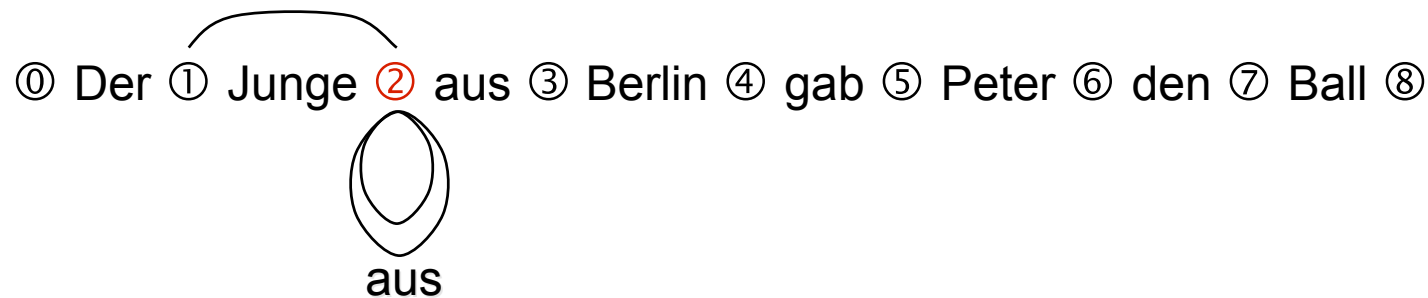
## Vorhersage (Predictor)

Für jeden Eintrag  $\langle h, i, A \rightarrow \alpha.B \beta \rangle$  füge für jede Regel  $B \rightarrow \gamma$  den Eintrag  $\langle i, i, B \rightarrow .\gamma \rangle$  in die Chart ein, wenn er nicht schon in der Chart steht.

*Position = 2*

Für  $\langle 1, 2, N \rightarrow N.PP \rangle$  und  $PP \rightarrow P NP$  füge ein:  $\langle 2, 2, PP \rightarrow .P NP \rangle$

Für  $\langle 2, 2, PP \rightarrow .P NP \rangle$  und  $P \rightarrow \text{aus}$  füge ein:  $\langle 2, 2, P \rightarrow .\text{aus} \rangle$



# Scanner

---

## Eingabeverarbeitung (Scanner)

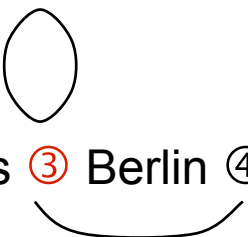
Wenn  $w_i = a$  füge für jeden Eintrag  $\langle h, i - 1, A \rightarrow \alpha.a \beta \rangle$   
einen Eintrag  $\langle h, i, A \rightarrow \alpha a.\beta \rangle$ .

---

*Position = 3*

Für  $w_i = \text{Berlin}$  und  $\langle 3, 3, \text{PN} \rightarrow \text{.Berlin} \rangle$  füge ein:  $\langle 3, 4, \text{PN} \rightarrow \text{Berlin.} \rangle$

① Der ① Junge ② aus ③ Berlin ④ gab ⑤ Peter ⑥ den ⑦ Ball ⑧



# Completer

---

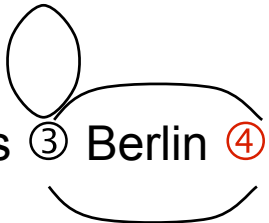
## Vervollständigung (Completer)

Für jedes Paar von Einträgen der Form  $\langle h, i, A \rightarrow \alpha. \rangle$  und  $\langle k, h, B \rightarrow \beta .A \gamma \rangle$  füge den Eintrag  $\langle k, i, B \rightarrow \beta A .\gamma \rangle$  in die Chart ein, wenn er nicht schon in der Chart steht.

*Position = 4*

**Für**  $\langle 3, 4, PN \rightarrow \text{Berlin.} \rangle$  **und**  $\langle 3, 3, NP \rightarrow \text{.PN} \rangle$  **füge ein:**  $\langle 3, 4, NP \rightarrow \text{PN.} \rangle$

① Der ① Junge ② aus ③ Berlin ④ gab ⑤ Peter ⑥ den ⑦ Ball ⑧



# Completer

---

## Vervollständigung (Completer)

Für jedes Paar von Einträgen der Form  $\langle h, i, A \rightarrow \alpha. \rangle$  und  $\langle k, h, B \rightarrow \beta .A \gamma \rangle$  füge den Eintrag  $\langle k, i, B \rightarrow \beta A .\gamma \rangle$  in die Chart ein, wenn er nicht schon in der Chart steht.

*Position = 4*

Für  $\langle 3, 4, \text{PN} \rightarrow \text{Berlin}. \rangle$  und  $\langle 3, 3, \text{NP} \rightarrow .\text{PN} \rangle$  füge ein:  $\langle 3, 4, \text{NP} \rightarrow \text{PN}. \rangle$

Für  $\langle 3, 4, \text{NP} \rightarrow \text{PN}. \rangle$  und  $\langle 2, 3, \text{PP} \rightarrow \text{P} .\text{NP} \rangle$  füge ein:  $\langle 2, 4, \text{PP} \rightarrow \text{P NP}. \rangle$

① Der ① Junge ② aus ③ Berlin ④ gab ⑤ Peter ⑥ den ⑦ Ball ⑧

## Übung

---

- Fügen Sie der Grammatik die Regel  $VP \rightarrow VP PP$  hinzu.
- Parsen Sie mithilfe des Earley Algorithmus den folgenden Satz:

Der Junge sah Peter aus Berlin.

- Füllen Sie dabei die Chart vollständig aus.

## Weitere Übungsaufgaben bzw. -fragen

---

- Was ist die Definition eines Konstituentenstrukturbaumes?
- Was ist der Unterschied zwischen kontextsensitiven und kontextfreien Ersetzungsregelgrammatiken?
- Für welche der folgenden künstlichen Sprachen kann ich einen endlichen Erkennungsautomaten (keinen Transduktor) schreiben und für welche lediglich eine kontextfreie Ersetzungsregelgrammatik? Für welche kann ich weder einen Automaten noch eine KFG schreiben?

$L_1$  ist die Menge aller Wörter  $x = a^n b^n c^n$ ,  $x \in \{a, b, c\}^*$

$L_2$  ist die Menge aller Wörter  $x = ww^R$  und  $x = w\alpha w^R$  wo  $w^R$  das Spiegelbild von  $w$  ist  $\alpha \in \{a, b, \dots, z\}$  und  $w \in \{a, b, \dots, z\}^*$  (Palyndromsprache) z.B. anna, babbab, tat, retsinakanister

$L_3$  ist die Menge aller Wörter  $x = a^{2n} b^n a^{2n}$

$L_4$  ist die Menge aller Wörter  $x = a^{2n} b^m a^{2n}$

$L_5$  ist die Menge aller Wörter  $x = (a b^n)^m$ ,  $x \in \{a, b\}^*$

Schreiben Sie bitte die jeweiligen Automaten bzw. Grammatiken

