Einführung in die Computerlinguistik
Chart-Parsing

Dozentin: Wiebke Petersen

21.12.2009

Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{rcl|rcl|rcl} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

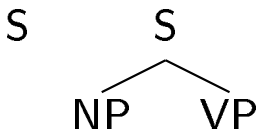
Mary calls Peter

S

Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} \text{S} & \rightarrow & \text{NP VP} & \text{VP} & \rightarrow & \text{V} & \text{VP} & \rightarrow & \text{V NP} \\ \text{V} & \rightarrow & \text{calls} & & & & & & \\ \text{NP} & \rightarrow & \text{Peter} & \text{NP} & \rightarrow & \text{Mary} & & & \end{array} \right\}$$

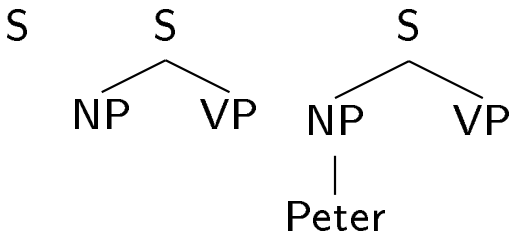
Mary calls Peter



Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} \text{S} & \rightarrow & \text{NP VP} & \text{VP} & \rightarrow & \text{V} & \text{VP} & \rightarrow & \text{V NP} \\ \text{V} & \rightarrow & \text{calls} & & & & & & \\ \text{NP} & \rightarrow & \text{Peter} & \text{NP} & \rightarrow & \text{Mary} & & & \end{array} \right\}$$

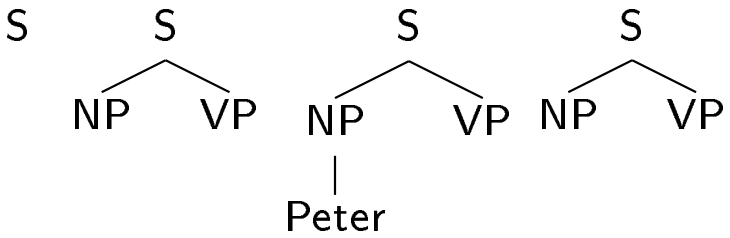
Mary calls Peter



Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

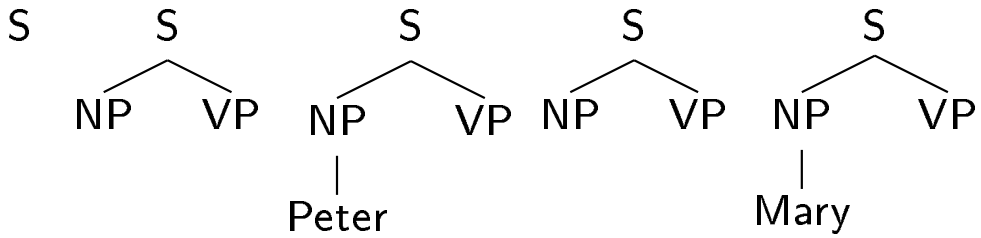
Mary calls Peter



Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

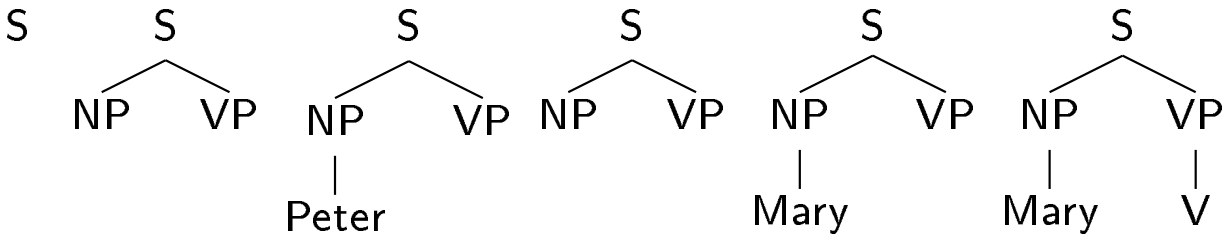
Mary calls Peter



Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

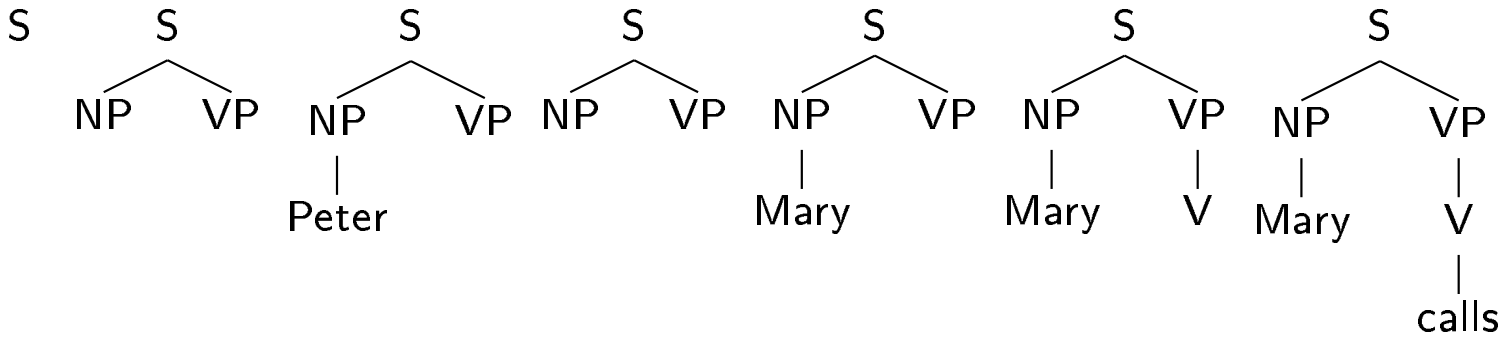
Mary calls Peter



Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

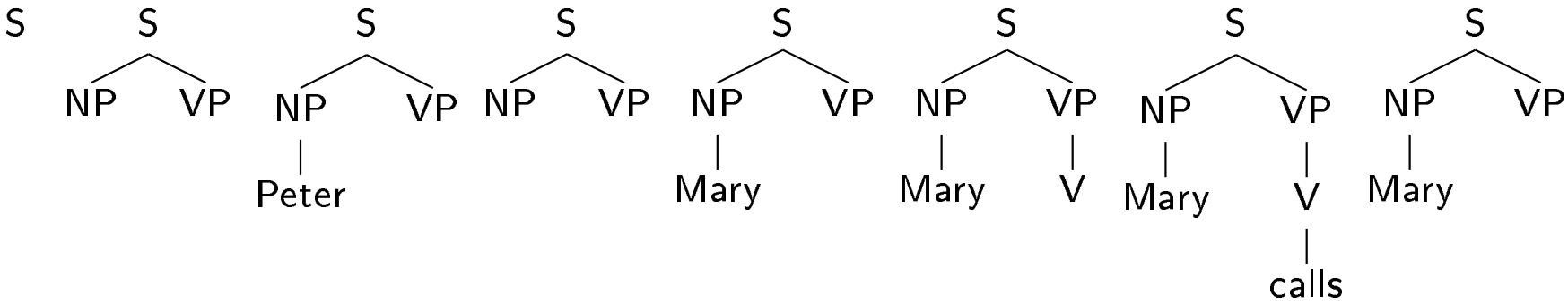
Mary calls Peter



Naives Parsen ⇒ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

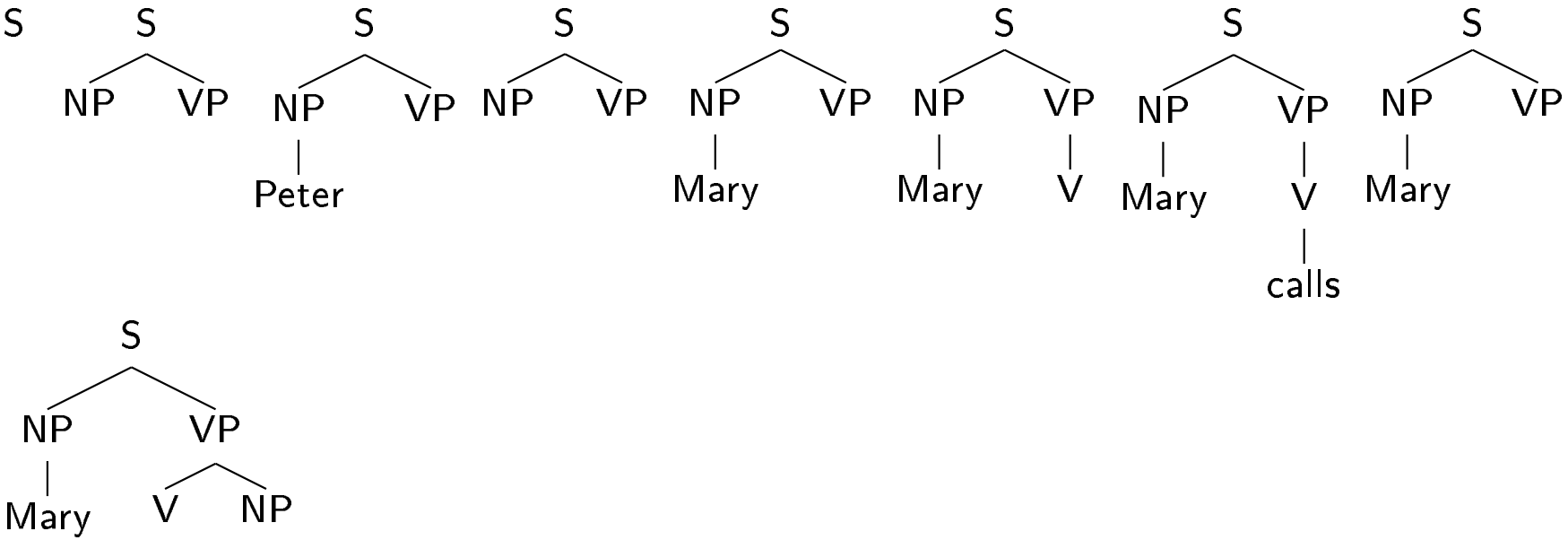
Mary calls Peter



Naives Parsen $\Rightarrow$ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

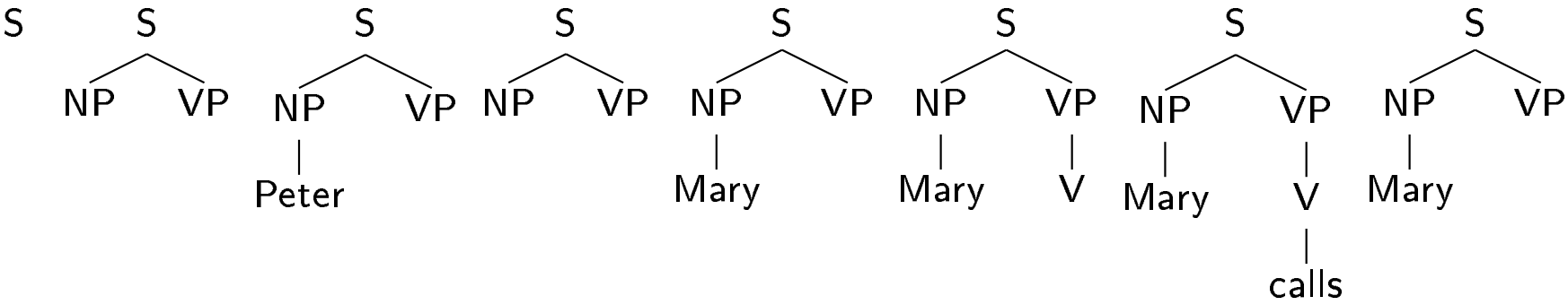
Mary calls Peter

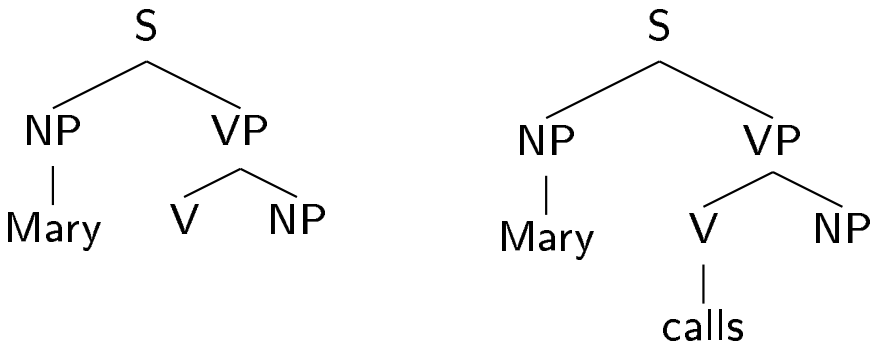


Naives Parsen ⇒ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} S & \rightarrow & NP\ VP & VP & \rightarrow & V & VP & \rightarrow & V\ NP \\ V & \rightarrow & calls & & & & & & \\ NP & \rightarrow & Peter & NP & \rightarrow & Mary & & & \end{array} \right\}$$

Mary calls Peter

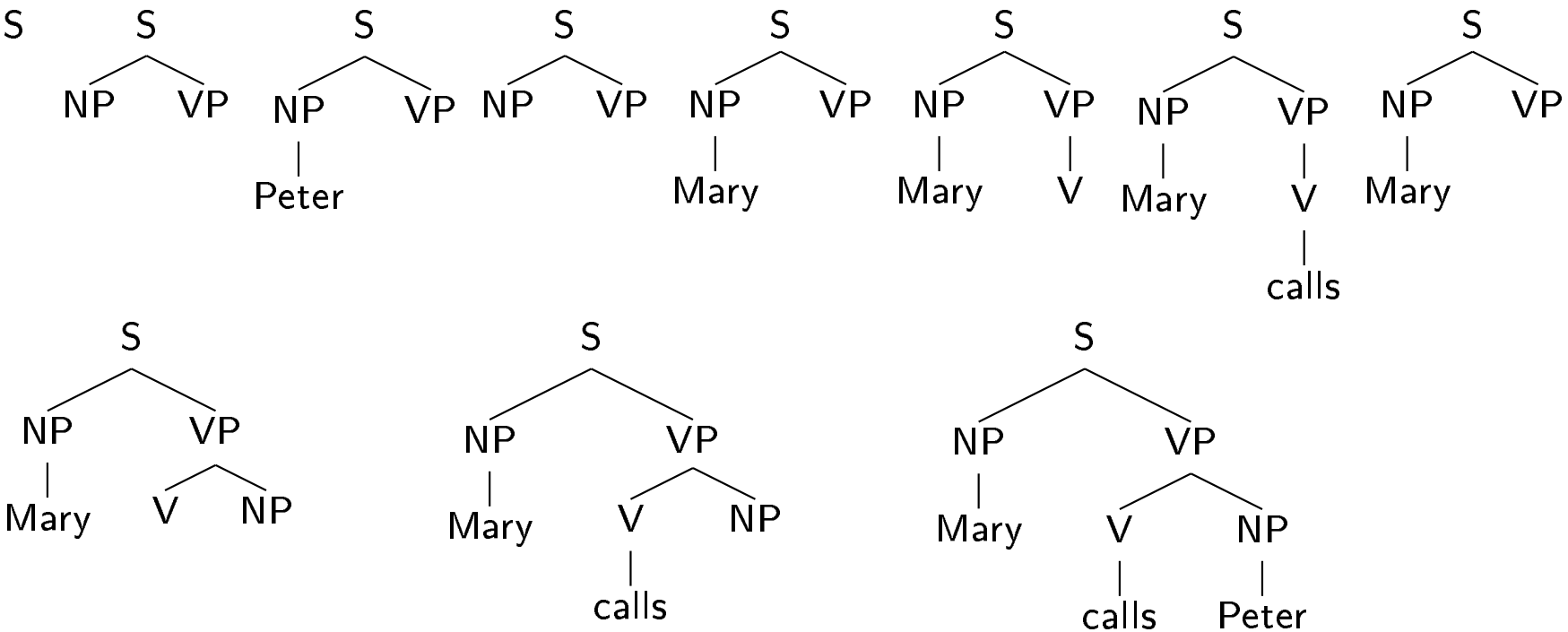




Naives Parsen ⇒ redundante Berechnungen

$$P = \left\{ \begin{array}{llllllllll} \text{S} & \rightarrow & \text{NP VP} & \text{VP} & \rightarrow & \text{V} & \text{VP} & \rightarrow & \text{V NP} \\ \text{V} & \rightarrow & \text{calls} & & & & & & \\ \text{NP} & \rightarrow & \text{Peter} & \text{NP} & \rightarrow & \text{Mary} & & & \end{array} \right\}$$

Mary calls Peter



Dynamische Programmierung

Ziel: Parsing-Methode,

- die einmal berechnete Teilbäume speichert und nicht erneut berechnet
- kein Problem mit Linksrekursion hat

$\Rightarrow$ Chart-Parser speichern Zwischenergebnisse in einer Chart

Dynamische Programmierung

Ziel: Parsing-Methode,

- die einmal berechnete Teilbäume speichert und nicht erneut berechnet
- kein Problem mit Linksrekursion hat

$\Rightarrow$ Chart-Parser speichern Zwischenergebnisse in einer Chart

Chart-Parser: CYK (Cocke-Younger-Kasami)

- setzt voraus, dass die Grammatik in Chomsky-Normalform ist
- wird auch CKY-Parser genannt

Chomsky-Normalform

Definition 1

Eine Grammatik ist in Chomsky-Normalform (CNF), wenn alle Regeln die Gestalt

1. $A \rightarrow a$
2. $A \rightarrow BC$

   mit $A, B, C \in T$ und $a \in \Sigma$ haben (und gegebenenfalls $S \rightarrow \epsilon$, dann aber ohne S in den rechten Regelseiten).

Chomsky-Normalform

Definition 1

Eine Grammatik ist in Chomsky-Normalform (CNF), wenn alle Regeln die Gestalt

1. $A \rightarrow a$
2. $A \rightarrow BC$

   mit $A, B, C \in T$ und $a \in \Sigma$ haben (und gegebenenfalls $S \rightarrow \epsilon$, dann aber ohne S in den rechten Regelseiten).

Theorem 2

Zu jeder kontextfreien Sprache gibt es eine Grammatik in Chomsky-Normalform, die sie generiert.

Was sind die charakteristischen Eigenschaften einer Grammatik in CNF?

Jede kontextfreie Sprache kann durch eine Grammatik in Chomsky-Normalform generiert werden

3 Umwandlungsschritte

1. Eliminierung der Regeln der Form $A \rightarrow B$.
   Ersetze diese Regel durch alle Regeln $A \rightarrow \beta$, für die es eine Regel $B \rightarrow \beta$ gibt.
2. Anpassen der Grammatik, so dass Terminale nur in Regeln der Form $A \rightarrow a$ vorkommen.
   Einfügen eines Dummy-Nichtterminalen für jeden Terminalen, der in einer falschen Regel steht.
3. Eliminierung der Regeln der Form $A \rightarrow B_1 B_2 \ldots B_n$ mit $n \geq 3$.
   Wiederholtes Einfügen eines Dummy-Nichtterminalen für jedes Cluster $B_2 \ldots B_n$.

CYK-Parser

- Demo von Roy Mennicke
  `http://www.informatik.uni-leipzig.de/alg/lehre/ss08/AUTO-SPRACHEN/Java-Applets/CYK-Algorithmus.html`
  Wandelt Grammatiken in CNF um, beschreibt detailliert die Umwandlung und den CYK-Algorithmus.
- Demo von Martin Lazarov
  `http://homepages.uni-tuebingen.de/student/martin.lazarov/demos/cky.html`
  Erlaubt die Eingabe von Grammatiken mit komplexen Terminal- und Nichtterminalsymbolen (z.B., “Maria”, “NP”)
- Exorciser
  Zeigt die implizit erzeugten Derivationsbäume an.