Parsingschemata

Vorlesung "Computerlinguistische Techniken"

Alexander Koller

10. November 2014

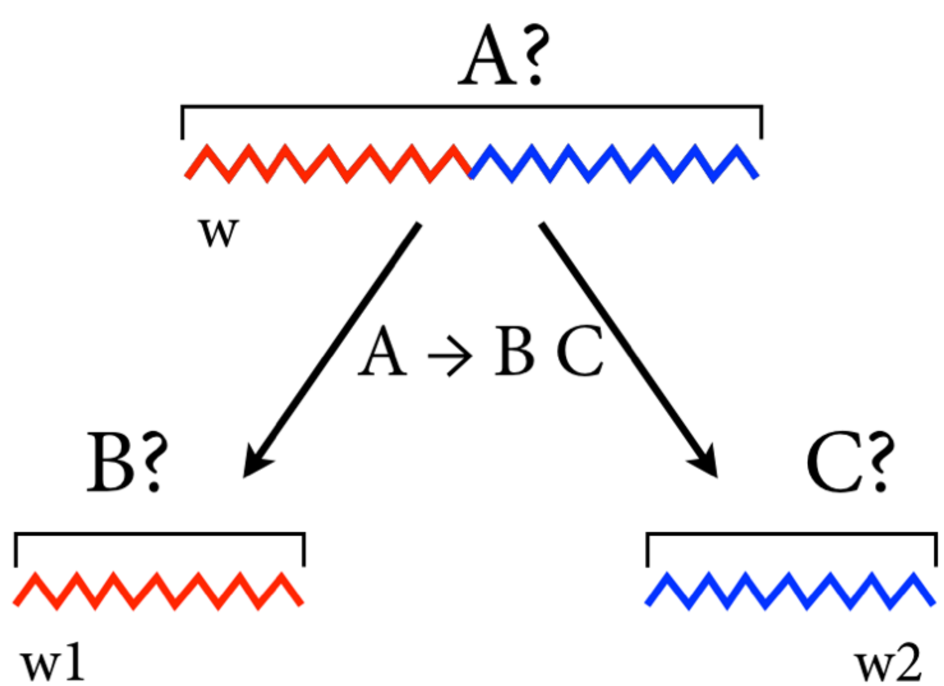
Was bisher geschah

- Wir haben in der Vorlesung bisher vier Parsingalgorithmen gesehen:
  - Recursive Descent
  - Shift-Reduce
  - CKY
  - Earley
- Es gibt in der Literatur noch viel mehr. Da ist es leicht, den Überblick zu verlieren.

Recursive-Descent-Parsing

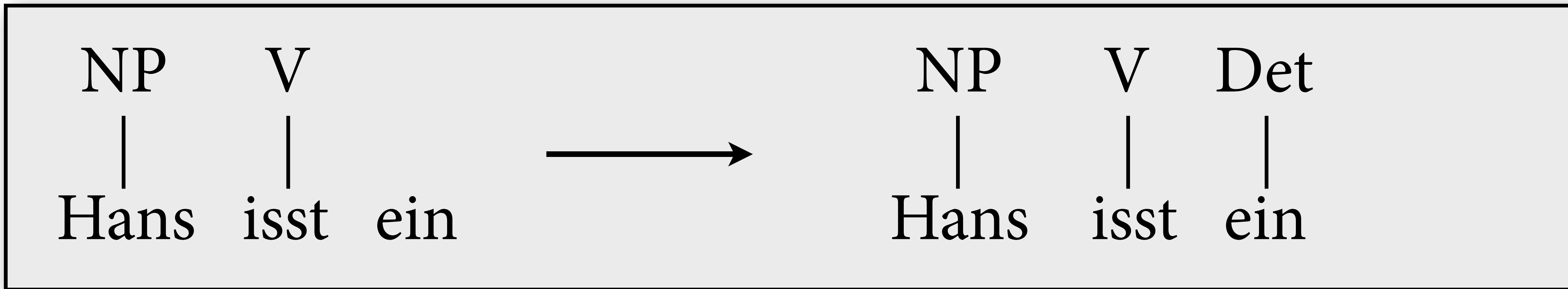
- Rekursiver Algorithmus, der für A und w die Frage "$A \Rightarrow^* w$?" entscheidet.

- Grundidee:

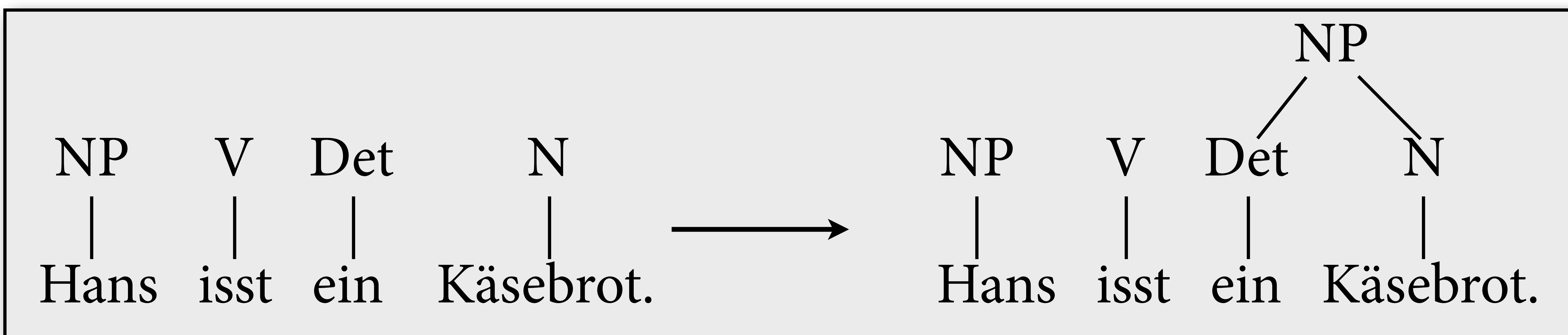


Shift-Reduce-Parsing

Shift: nächstes Terminalsymbol lesen

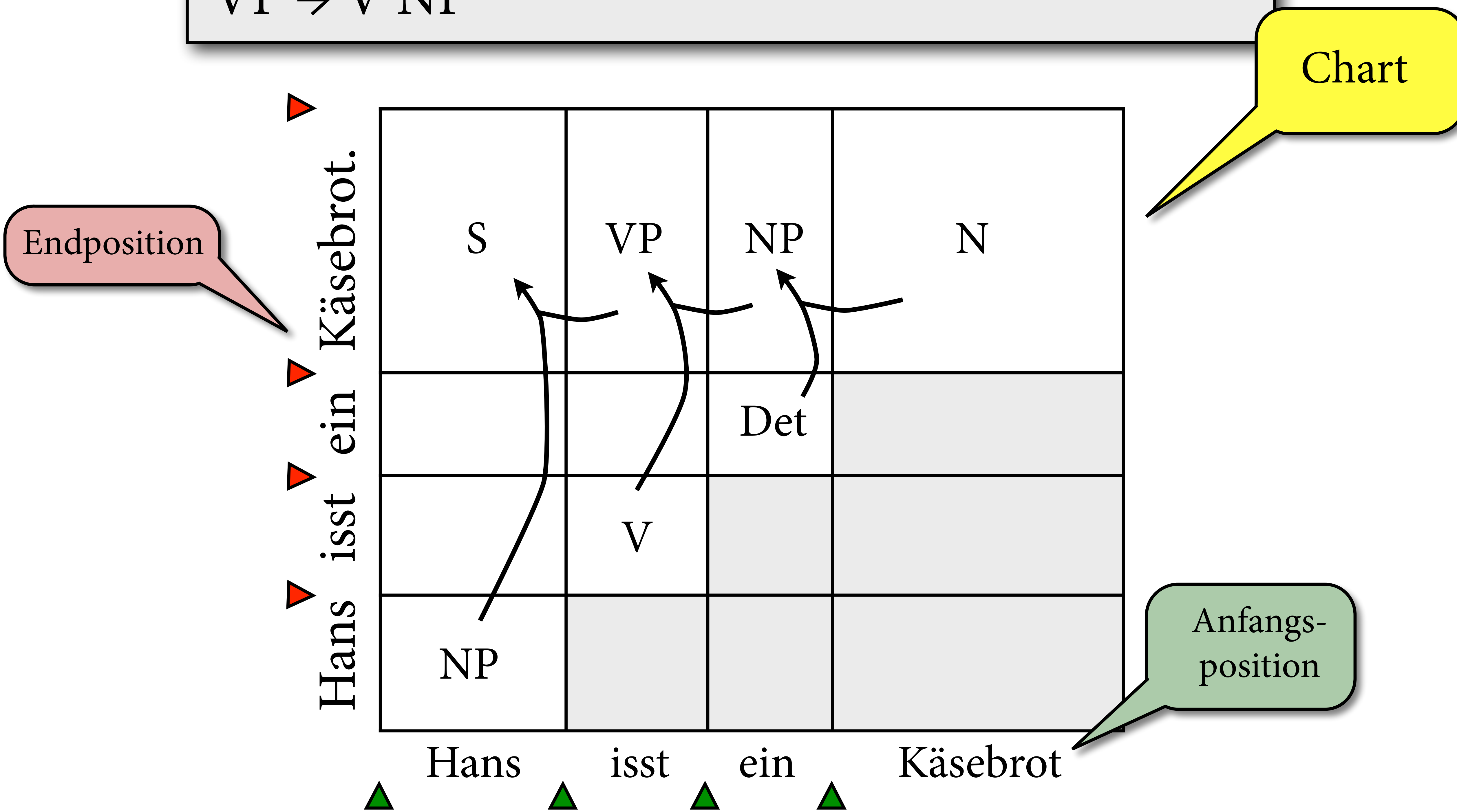


Reduce: Regel auf letzte offene Symbole anwenden



Der CKY-Parser

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		



Der Earley-Parser

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

Predict
Scan
Complete

Punkt-position

K.brot	1, S → NP VP •, 5 **1, S' → S •, 5**	2, VP → V NP •, 5	3, NP → Det N •, 5	4, N → kb •, 5
ein			3, Det → ein •, 4 3, NP → Det • N, 4	4, N → • kb, 4
isst		2, V → isst •, 3 2, VP → V • NP, 3	3, NP → • Det N, 3 3, Det → • ein, 3	
Hans	1, NP → Hans •, 2 1, S → NP • VP, 2	2, VP → • V NP, 2 2, V → • isst, 2		
	1, S' → • S, 1 1, S → • NP VP, 1 1, NP → • Hans, 1			
	Hans	isst	ein	Käsebrot

Anfangs-position

Übersicht

- Beweisregeln und Parsingschemata
- Parsingschemata für CKY, Earley, Shift-Reduce
- Vereinfachte Kochrezepte für
  Korrektheit, Vollständigkeit, Laufzeit
  auf der Grundlage von Parsingschemata.

Beweisregeln

- Eine mögliche Welt in Aussagenlogik formalisiert:

Wenn Hans kein Geld hat, kann er nicht einkaufen gehen.	$\neg Geld \rightarrow \neg Eink$
Wenn Hans nicht einkaufen geht, hat er kein Essen.	$\neg Eink \rightarrow \neg Essen$
Wenn Hans kein Essen hat, dann hat er morgen Hunger.	$\neg Essen \rightarrow Hunger$
Hans hat kein Geld.	$\neg Geld$

- Wie beweist man automatisch, dass Hans morgen Hunger hat?

Beweisregeln

- Eine Beweisregel erlaubt uns, aus Prämissen eine Konklusion abzuleiten.

- Beispiel: Modus Ponens

$$\frac{P \to Q \qquad P}{Q}$$

- P, Q sind Platzhalter für beliebige Formeln.

Beispiel

- Beweis durch schematische Anwendung von Modus Ponens:

$$\frac{P \rightarrow Q \qquad P}{Q}$$

1.	$\neg$Geld	(Axiom)
2.	$\neg$Geld $\rightarrow$ $\neg$Eink	(Axiom)
3.	$\neg$Eink $\rightarrow$ $\neg$Essen	(Axiom)
4.	$\neg$Essen $\rightarrow$ Hunger	(Axiom)
5.	$\neg$Eink	(MP aus 1, 2)
6.	$\neg$Essen	(MP aus 3, 5)
7.	Hunger	(MP aus 4, 6)

Beispiel

(alternative Darstellung als Beweisbaum)

Axiome

$$\dfrac{\neg\text{Essen} \to \text{Hunger} \qquad \dfrac{\neg\text{Eink} \to \neg\text{Essen} \qquad \dfrac{\neg\text{Geld} \to \neg\text{Eink} \qquad \neg\text{Geld}}{\neg\text{Eink}}}{\neg\text{Essen}}}{\text{Hunger}}$$

Theorem

$$\boxed{\dfrac{P \to Q \qquad P}{Q}}$$

Wahrheit und Beweisbarkeit

- Eine Formel F *folgt* aus den Formeln $P_1, \ldots, P_n$, wenn jedes Modell, das $P_1, \ldots, P_n$ wahr macht, auch F wahr macht. $P_1, \ldots, P_n \models F$
  - Folgerung ist ein *semantischer* Begriff: basiert auf der Wahrheit von Formeln in Modellen/Variablenbelegungen.

- Eine Formel F ist aus $P_1, \ldots, P_n$ *beweisbar,* wenn sie Theorem eines Beweises ist mit den Axiomen $P_1, \ldots, P_n$. $P_1, \ldots, P_n \vdash F$
  - Beweisbarkeit ist ein *syntaktischer* Begriff: basiert auf der Manipulation von Formeln mit Beweisregeln.

Korrektheit und Vollständigkeit

- Ein System von Beweisregeln heißt *Kalkül*.
- Kalkül heißt *korrekt*, wenn jedes beweisbare Theorem auch tatsächlich aus den Axiomen folgt.
- Kalkül heißt *vollständig*, wenn jede Formeln, die logisch aus den Axiomen folgt, auch aus ihnen bewiesen werden kann.

$$\boxed{\models \;\Leftrightarrow\; \vdash}$$

Parsingschemata

- Idee (Shieber, Pereira, Schabes 94):
  - Parser als System von Beweisregeln hinschreiben.
  - Parser ≈ Kalkül
  - Grammatikalität ≈ Wahrheit
  - Ableitbarkeit ≈ Beweisbarkeit
- Vorteile:
  - Unterschiede zwischen Algorithmen übersichtlich.
  - Laufzeit ist aus Schema direkt ablesbar.
  - Einheitliche Rezepte für Korrektheitsbeweise.

Parsing-Items

- In einer Parse-Chart sammeln wir *Items:*
  - CKY: [A, i, k]
  - Earley: [i, A → α • β, k]

- Items machen *Aussagen* über den String:
  - [A, i, k] bedeutet: $A \Rightarrow^* w_i \ldots w_{k-1}$
  - [i, A → α • β, k] bedeutet (unter anderem):
    $A \Rightarrow \alpha\, \beta \Rightarrow^* w_i \ldots w_{k-1}\, \beta$

- Aufgabe des Parsers: alle wahren Items berechnen.

Parsingschema

- Ein Parsingschema besteht aus:
  - einer Spezifikation der möglichen *Items*
  - *Beweisregeln*, die aussagen, wie man Items aus anderen Items ableiten darf
  - für jede Eingabe: eine endliche Menge von *Start-Items*, mit denen die Berechnung anfängt (≈ Axiome)
  - für jede Eingabe: eine endliche Menge von *Ziel-Items* (≈ Theoreme)

- Parser behauptet $S \Rightarrow^* w$ gdw. man für w aus den Start-Items ein Ziel-Item ableiten kann.

Der CKY-Parser als Schema

- Items des CKY-Parser: [A, i, k]
  - Bedeutung: $A \Rightarrow^* w_i \ldots w_{k-1}$
- Start-Items: alle Items [A, i, i+1], für die $A \rightarrow w_i$ Produktionsregel ist
- Ziel-Item: [S, 1, n+1]
  - denn das bedeutet gerade $S \Rightarrow^* w_1 \ldots w_n$

Der CKY-Parser als Schema

$$\frac{[i, B, j] \quad [j, C, k] \quad A \rightarrow B\ C \text{ ist Regel}}{[i, A, k]}$$

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

[1, NP, 2]   [2, V, 3]   [3, Det, 4]   [4, N, 5]

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

Start-Items

[1, NP, 2] [2, V, 3] [3, Det, 4] [4, N, 5]

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

Start-Items

[1, NP, 2] [2, V, 3] [3, Det, 4] [4, N, 5]

$$\frac{[3, \mathrm{Det}, 4] \quad [4, \mathrm{N}, 5]}{[3, \mathrm{NP}, 5]}$$

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

Start-Items

$$[1, NP, 2] \qquad \dfrac{[2, V, 3] \qquad \dfrac{[3, Det, 4] \qquad [4, N, 5]}{[3, NP, 5]}}{[2, VP, 5]}$$

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

Start-Items

$$\dfrac{[1, NP, 2] \qquad \dfrac{[2, V, 3] \qquad \dfrac{[3, Det, 4] \qquad [4, N, 5]}{[3, NP, 5]}}{[2, VP, 5]}}{[1, S, 5]}$$

CKY-Schema: Berechnung

S → NP VP	V → isst	Det → ein
NP → Det N	NP → Hans	N → Käsebrot
VP → V NP		

Start-Items

$$\dfrac{[1, NP, 2] \qquad \dfrac{[2, V, 3] \qquad \dfrac{[3, Det, 4] \qquad [4, N, 5]}{[3, NP, 5]}}{[2, VP, 5]}}{[1, S, 5]}$$

Ziel-Item

Earley-Parser als Schema

- Items des Earley-Parsers: $[i, A \rightarrow \alpha \bullet \beta, k]$
- Ein Item macht zwei Aussagen:
  - $A \Rightarrow \alpha\, \beta \Rightarrow^* w_i \ldots w_{k-1}\, \beta$ (bottom-up)
  - $S \Rightarrow^* w_1 \ldots w_{i-1}\, A\, \gamma$ für irgendein γ (top-down)
- Start-Items: $[1, S \rightarrow \bullet\, \gamma, 1]$ für alle Regeln $S \rightarrow \gamma$
- Ziel-Items: alle $[1, S \rightarrow \gamma\, \bullet, n+1]$

NB: Das ist alternative Formulierung der Earley-Startitems.

Earley-Parser als Schema

$$\frac{[i, A \rightarrow \alpha \bullet B\,\beta, k] \qquad B \rightarrow \gamma \text{ ist Regel}}{[k, B \rightarrow \bullet\, \gamma, k]} \text{ Predict}$$

$$\frac{[j, A \rightarrow \alpha \bullet wi\ \beta, i]}{[j, A \rightarrow \alpha\ wi \bullet \beta, i+1]} \text{ Scan}$$

$$\frac{[i, A \rightarrow \alpha \bullet B\,\beta, j] \qquad [j, B \rightarrow \gamma\, \bullet, k]}{[i, A \rightarrow \alpha\, B \bullet \beta, k]} \text{ Complete}$$

Schema für Shift-Reduce

- Items sind Paare [i, s] aus Eingabeposition und Stack.
  - [i, s] bedeutet: $s\ w_i \ldots w_n \Rightarrow^* w$

- Start-Item ist $[1, \varepsilon]$

- Ziel-Item ist $[n+1, S]$

Schema für Shift-Reduce

$$\frac{[i, s]}{[i{+}1, s \cdot w_i]} \; \text{Shift}$$

$$\frac{[i, s \cdot \gamma] \qquad A \rightarrow \gamma \text{ ist Regel}}{[i, s \cdot A]} \; \text{Reduce}$$

Recursive Descent

- Recursive Descent passt nicht so gut ins Schema.

- Kann man auch als Schema schreiben, wird aber recht unnatürlich:

  - Item [A •, i, k] bedeutet: $A \Rightarrow^* w_i \ldots w_{k-1}$

  - Item [• A, i, k] hat keine Bedeutung im engeren Sinn (nur dass es nützlich wäre, [A •, i, k] abzuleiten)

- Betrachten wir deshalb heute nicht.

Korrektheit und Vollständigkeit

- Erkenner behauptet "$w \in L(G)$" gdw. ein Ziel-Item ableitbar ist. Hat er recht?

- Wir müssen für jeden Algorithmus zeigen:
  - *Korrektheit:* Erkenner leitet Ziel-Items nur ab, wenn $w \in L(G)$.
  - *Vollständigkeit:* Wenn $w \in L(G)$, dann leitet Erkenner auch ein Ziel-Item ab.

Rezept für Korrektheit

- Mit Schemata kann man besonders einfach Korrektheit von Parsing-Algorithmen beweisen.

- Induktion über # Anwendung von Beweisregeln:
  - Anfang: zeige, dass alle Start-Items wahr sind
  - Schritt: zeige, dass jede Regel aus wahren Items nur wahre Items ableiten kann
  - Zeige, dass aus der Aussage jedes Ziel-Items $S \Rightarrow^* w$ folgt (normalerweise trivial).

Korrektheit von CKY

- Aussage von [A, i, k]: $A \Rightarrow^* w_i \ldots w_{k-1}$

- Zeige, dass alle Start-Items wahr sind:
  - [A, i, i+1] ist Start-Item, wenn $A \rightarrow w_i$ Regel, also sind sie alle wahr.

- Zeige, dass aus Ziel-Items $S \Rightarrow^* w$ folgt:
  - einziges Ziel-Item ist [S, 1, n+1];
    seine Aussage ist $S \Rightarrow^* w_1 \ldots w_n = w$.

Korrektheit von CKY: Regeln

- Zeige, dass alle Regeln Wahrheit von Items erhalten:
  - es gibt nur eine Regel (siehe unten)
  - angenommen, Prämissen sind wahr,
    d.h. $B \Rightarrow^* w_i \dots w_{j-1}$ und $C \Rightarrow^* w_j \dots w_{k-1}$
  - dann gilt $A \Rightarrow B\ C \Rightarrow^* w_i \dots w_{k-1}$,
    d.h. [A, i, k] ist wahr.

$$\frac{[i, B, j] \quad [j, C, k] \quad A \rightarrow B\ C \text{ ist Regel}}{[i, A, k]}$$

Korrektheit von Earley

- Item $[i, A \to \alpha \bullet \beta, k]$ macht zwei Aussagen:
  - $A \Rightarrow \alpha\, \beta \Rightarrow^* w_i \dots w_{k-1}\, \beta$ (bottom-up)
  - top-down-Aussage für Korrektheit nicht nötig
- Zeige, dass alle Start-Items wahr sind:
  - Start-Items sind von Form $[1, S \to \bullet\, \gamma, 1]$
  - $S \Rightarrow \gamma \Rightarrow^* \gamma$ (in null Schritten)
- Zeige, dass aus Aussage von Ziel-Items $w \in L(G)$ folgt:
  - Ziel-Items sind von der Form $[1, S \to \gamma\, \bullet, n+1]$
  - Aussage ist $S \Rightarrow \gamma \Rightarrow^* w_1 \dots w_n$

Korrektheit von Earley: Regeln

$$\frac{[i, A \rightarrow \alpha \bullet B\ \beta, k] \qquad B \rightarrow \gamma \text{ ist Regel}}{[k, B \rightarrow \bullet\ \gamma, k]}\ \text{Predict}$$

$$\frac{[j, A \rightarrow \alpha \bullet wi\ \beta, i]}{[j, A \rightarrow \alpha\ wi \bullet \beta, i+1]}\ \text{Scan}$$

$$\frac{[i, A \rightarrow \alpha \bullet B\ \beta, j] \qquad [j, B \rightarrow \gamma\ \bullet, k]}{[i, A \rightarrow \alpha\ B \bullet \beta, k]}\ \text{Complete}$$

Vollständigkeit

- Entspricht Induktion über “Größe” von Items (muss man definieren).

- Zeige:
  - wenn $S \Rightarrow^* w$, dann ist ein Ziel-Item wahr
  - alle wahren Items sind ableitbar

- Kochrezept:
  - alle kleinsten wahren Items sind ableitbar
  - wenn alle wahren Items von Größe $< n$ ableitbar sind, dann auch alle wahren Items von Größe n
  - alle wahren Items haben endliche Größe

CKY: Vollständigkeit

- Größe von [A, i, k] ist k-i (d.h. Breite).
- Alle kleinsten wahren Items (also die mit Breite 1) sind Start-Items.
- Alle wahren Items haben höchstens Größe n.

CKY: Vollständigkeit

- Angenommen:
  - [A, i, k] ist ein wahres Item (von Breite k-i)
  - Wir haben alle wahren Items von Größe 1 bis k-i-1 schon berechnet.

- Weil [A,i,k] wahr ist, gibt es B, C mit
  $A \Rightarrow B\ C \Rightarrow^* w_i \ldots w_{k-1}$.
  - $B \Rightarrow^* w_i \ldots w_{j-1}$ und $C \Rightarrow^* w_j \ldots w_{k-1}$ für irgendein $i < j < k$
  - [B, i, j] und [C, j, k] sind wahre Items mit Größe < k-i
  - Also ist [A, i, k] ableitbar.

Earley: Vollständigkeit

- Passt nicht richtig ins Rezept, weil man gleichzeitig top-down und bottom-up argumentieren muss.

- Idee:
  - Induktionsanfang: alle wahren Items der Form $[A \to \bullet\ \gamma]$ werden abgeleitet (aus Predict)
  - Induktionsschritt: wenn alle kleineren wahren Items abgeleitet werden, dann auch alle wahren$[A \to \alpha \bullet \beta]$ (aus Scan oder Complete, je nach letztem Zeichen in α).

Implementierung von Schemata

- Grundidee:
  - *Chart* enthält alle bisher abgeleiteten Items
  - *Agenda* ist Queue mit allen Items, die wir noch nicht als Prämisse einer Regel verwendet haben.

- Algorithmus:
  - initialisiere Chart und Agenda mit Start-Items
  - so lange Agenda nicht leer, nimm ein Item x von Agenda, wende alle Regeln auf x und Items in Chart an, füge bisher ungesehene Items zu Chart und Agenda hinzu.
  - teste, ob Chart ein Ziel-Item enthält

Laufzeit

- Optimale Implementierung eines Schemas schaut jede Instanz jeder Regel höchstens einmal an.
- Also Laufzeit = Anzahl der Regelinstanzen.
- Laufzeit direkt aus Regeln ablesbar:
    - $O(n)$ Möglichkeiten für jede Variable über Stringposition
- Vorsicht: Optimales Implementieren von Schemata nicht immer trivial.

Laufzeit von CKY

$$\frac{[i, B, j] \qquad [j, C, k] \qquad A \rightarrow B\ C \text{ ist Regel}}{[i, A, k]}$$

Laufzeit von CKY

$$\frac{[i, B, j] \qquad [j, C, k] \qquad A \to B\ C \text{ ist Regel}}{[i, A, k]}$$

O(n)

Laufzeit von CKY

$$\frac{[i, B, j] \qquad [j, C, k] \qquad A \to B\ C \text{ ist Regel}}{[i, A, k]}$$

$O(n) \qquad \times\, O(n)$

Laufzeit von CKY

$$\frac{[i, B, j] \qquad [j, C, k] \qquad A \to B\ C \text{ ist Regel}}{[i, A, k]}$$

$$O(n) \qquad \times\, O(n) \qquad \times\, O(n)$$

Laufzeit von CKY

$$\frac{[i, B, j] \qquad [j, C, k]}{[i, A, k]} \qquad A \to B\ C \text{ ist Regel}$$

$$O(n) \quad \times\, O(n) \quad \times\, O(n) \quad = O(n^3)$$

Laufzeit von CKY

$$\frac{[i, B, j] \qquad [j, C, k]}{[i, A, k]} \qquad A \rightarrow B\ C \text{ ist Regel}$$

$$O(n) \quad \times O(n) \quad \times O(n) \quad = O(n^3)$$

(mal Anzahl der Produktionsregeln)

Laufzeit von Earley

$$\frac{[i, A \to \alpha \bullet B\,\beta, k] \qquad B \to \gamma \text{ ist Regel}}{[k, B \to \bullet\, \gamma, k]} \text{ Predict}$$

$$\frac{[j, A \to \alpha \bullet wi\,\beta, i]}{[j, A \to \alpha\, wi \bullet \beta, i+1]} \text{ Scan}$$

$$\frac{[i, A \to \alpha \bullet B\,\beta, j] \qquad [j, B \to \gamma\, \bullet, k]}{[i, A \to \alpha\, B \bullet \beta, k]} \text{ Complete}$$

Laufzeit von Earley

$$\frac{[i, A \rightarrow \alpha \bullet B\, \beta, k] \qquad B \rightarrow \gamma \text{ ist Regel}}{[k, B \rightarrow \bullet\, \gamma, k]} \text{ Predict} \qquad O(n^2)$$

$$\frac{[j, A \rightarrow \alpha \bullet wi\, \beta, i]}{[j, A \rightarrow \alpha\, wi \bullet \beta, i+1]} \text{ Scan}$$

$$\frac{[i, A \rightarrow \alpha \bullet B\, \beta, j] \qquad [j, B \rightarrow \gamma\, \bullet, k]}{[i, A \rightarrow \alpha\, B \bullet \beta, k]} \text{ Complete}$$

Laufzeit von Earley

$$\frac{[i, A \rightarrow \alpha \bullet B\, \beta, k] \qquad B \rightarrow \gamma \text{ ist Regel}}{[k, B \rightarrow \bullet\, \gamma, k]} \text{ Predict} \qquad O(n^2)$$

$$\frac{[j, A \rightarrow \alpha \bullet wi\, \beta, i]}{[j, A \rightarrow \alpha\, wi \bullet \beta, i+1]} \text{ Scan} \qquad O(n^2)$$

$$\frac{[i, A \rightarrow \alpha \bullet B\, \beta, j] \qquad [j, B \rightarrow \gamma\, \bullet, k]}{[i, A \rightarrow \alpha\, B \bullet \beta, k]} \text{ Complete}$$

Laufzeit von Earley

$$\frac{[i, A \rightarrow \alpha \bullet B\, \beta, k] \qquad B \rightarrow \gamma \text{ ist Regel}}{[k, B \rightarrow \bullet\, \gamma, k]} \text{ Predict} \qquad O(n^2)$$

$$\frac{[j, A \rightarrow \alpha \bullet wi\, \beta, i]}{[j, A \rightarrow \alpha\, wi \bullet \beta, i+1]} \text{ Scan} \qquad O(n^2)$$

$$\frac{[i, A \rightarrow \alpha \bullet B\, \beta, j] \qquad [j, B \rightarrow \gamma\, \bullet, k]}{[i, A \rightarrow \alpha\, B \bullet \beta, k]} \text{ Complete} \qquad O(n^3)$$

Laufzeit von Earley

$$\frac{[i, A \to \alpha \bullet B\, \beta, k] \qquad B \to \gamma \text{ ist Regel}}{[k, B \to \bullet\, \gamma, k]} \text{ Predict} \qquad O(n^2)$$

$$\frac{[j, A \to \alpha \bullet wi\, \beta, i]}{[j, A \to \alpha\, wi \bullet \beta, i+1]} \text{ Scan} \qquad O(n^2)$$

$$\frac{[i, A \to \alpha \bullet B\, \beta, j] \qquad [j, B \to \gamma \bullet, k]}{[i, A \to \alpha\, B \bullet \beta, k]} \text{ Complete} \qquad O(n^3)$$

Laufzeit: $O(n^2) + O(n^2) + O(n^3) = O(n^3)$

Zusammenfassung

- Parsingschemata: Beweisregeln für Parsing-Items.
- Allgemeiner Ansatz, um Chartparsing-Algorithmen aufzuschreiben.
    - Vergleichbarkeit von Parsern
    - Laufzeit direkt ablesbar
    - Einheitliche Rezepte für Korrektheitsbeweise