Inkrementelle Syntax

CFG und inkrementelles Parsing

Timm Lichte

HHU Düsseldorf, Germany

30.10.2013

Aus der letzten Sitzung

inkrementelle Syntax

Mechanismen bzw. Komponenten eines Grammatikformalismus,

(i) die die Wortkette direkt derivieren/beschränken,
(ii) die irgendwie linguistisch motiviert sind,
(iii) die iterativ entlang der **Wortkette** operieren.

strikte/starke Inkrementalität (Sturt & Lombardo, 2005)

Jedes Wort wird ohne Verzögerung in eine zusammenhängende syntaktische Repräsentation inkorporiert.

Heutige Sitzung

- **Thema**: CFG und inkrementelles Parsing
- **Texte**:
  - Crocker (1999)
- **Fragestellungen:**
  - Wie kann man mit CFGs inkrementell parsen?
  - Was sind die Vorteile/Nachteile/Grenzen der dargestellten Parsingalgorithmen?
  - Wie verhalten sich die dargestellten Parsingalgorithmen zur Definition einer CFG?

Überblick

Definition

Direktionale Parsingalgorithmen (Crocker)

- Bottom-Up (“shift-reduce”, LR)
- Top-Down (LL)
- Left-Corner

Andere Regelformen und Parsingalgorithmen

- Earley Parsing
- PDA

Definition

Eine kontextfreie Grammatik (CFG) ist ein Tupel $G = \langle N, T, P, S \rangle$ bestehend aus

- einer Menge N von Nichtterminalen,
- einer Menge T von Terminalen, wobei N und T disjunkt sind,
- einer Menge von Produktionen P der Form $A \to \beta$ mit $A \in N, \beta \in (N \cup T)^*$,
- einem Startsymbol $S \in N$.

Ableitungsschritt ($\Rightarrow$):

- $w \Rightarrow w'$ mit $w, w' \in (N \cup T)^*$, gdw. $A \to \beta \in P$ und $u, v \in (N \cup T)^*$, so dass $w = uAv$ und $w' = u\beta v$.
- $\overset{*}{\Rightarrow}$ ist die reflexive, transitive Hülle von $\Rightarrow$.

Die **Stringsprache** einer CFG G ist $L(G) = \{w \in T^* | S \overset{*}{\Rightarrow} w\}$.

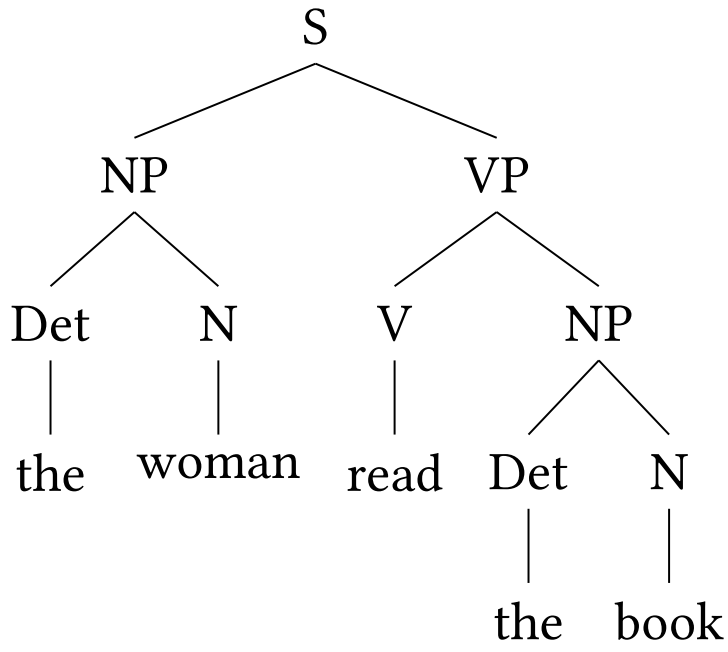
Beispiel

$S \rightarrow NP\ \ VP$
$VP \rightarrow V\ \ NP$
$VP \rightarrow V$
$V \rightarrow$ saw|read|sleeps

$NP \rightarrow Det\ \ N$
$N \rightarrow$ man|woman|book
$Det \rightarrow$ the|a

$S \Rightarrow NP\ \ VP$
$\Rightarrow NP\ \ V\ \ NP$
$\Rightarrow NP\ \ V\ \ Det\ \ N$
$\Rightarrow Det\ \ N\ \ V\ \ Det\ \ N$
$\Rightarrow Det\ \ N\ \ V\ \ Det$ book
$\Rightarrow Det\ \ N$ read $\ Det$ book
$\Rightarrow Det\ \ N$ read a book
$\Rightarrow$ the $\ N$ read a book
$\Rightarrow$ the woman read a book

Parse Tree (Ableitungsbaum)



Parsingalgorithmen (Crocker)

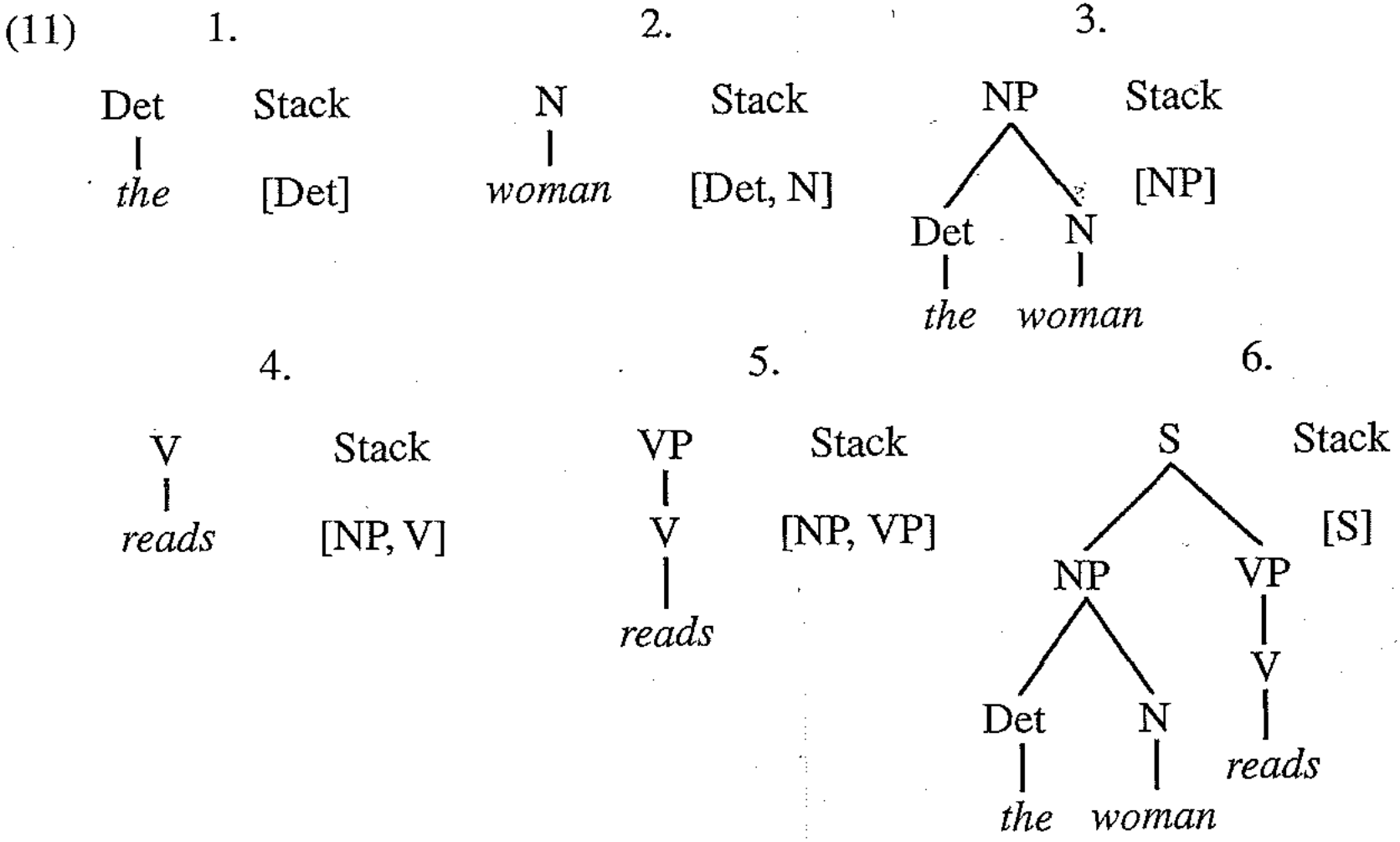
- Bottom-Up ("shift-reduce", LR)
- Top-Down (LL)
- Left-Corner

Beispielgrammatik (Fig. 7.1):

S	→	NP VP	Det	→	{*the, a, every*}
NP	→	PN	N	→	{*man, woman, book, hill, telescope*}
NP	→	Det N	PN	→	{*John, Mary*}
NP	→	NP PP	P	→	{*on, with*}
PP	→	P NP	V	→	{*saw, put, open, read, reads*}
VP	→	V			
VP	→	V NP			
VP	→	V NP PP			

Parsingalgorithmen (Crocker): Bottom-Up

(11)



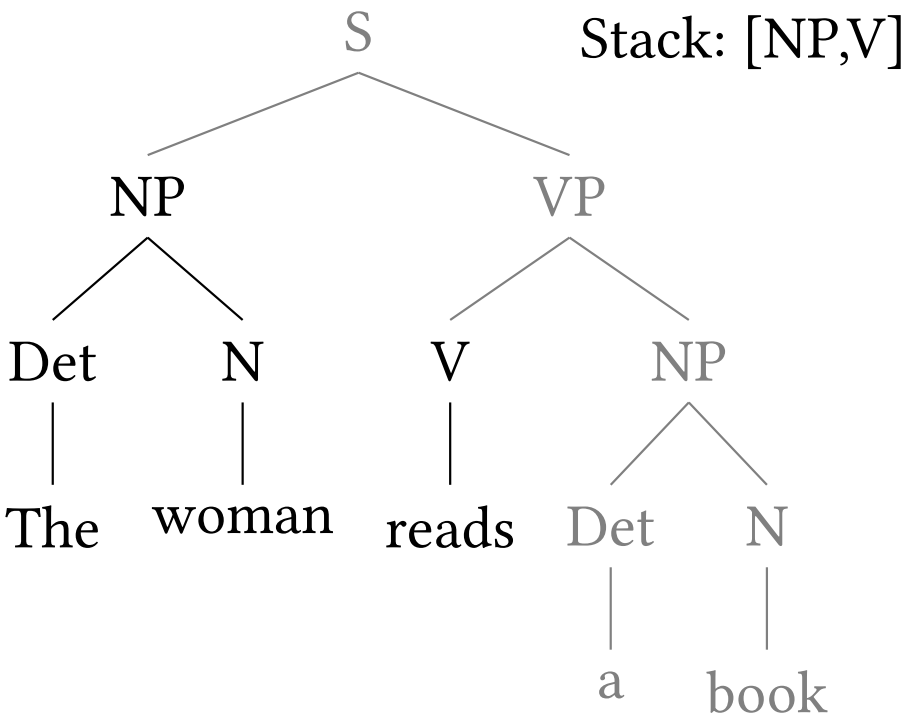
Parsingalgorithmen (Crocker): Bottom-Up

(12)
1. Initialise Stack = [ ] (empty)
2. Either *shift*:
   - Select the next word in the sentence (beginning with the first word)
   - Determine the category of the word in the lexicon
   - Push the category onto the top of the stack
3. Or *reduce*:
   - If the categories on the stack match those on the right-hand side of any grammar rule, then:
     —Remove those categories from the stack
     —Push the category on the left-hand side of the rule onto the top of the stack
4. If there are no more words in the sentence, then:
   - If the Stack = [S], then done
5. Go to step 2.

Parsingalgorithmen (Crocker): Bottom-Up

Eigenschaften:

- nicht-deterministisch, z.B. durch lexikalische Ambiguität ($N \rightarrow$ bank, $V \rightarrow$ bank)
- "input-driven"
- nicht immer strikt inkrementell:



Parsingalgorithmen (Crocker): Top-Down

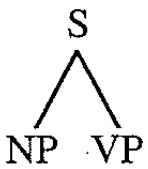
(13)

1.

S Stack

[S]

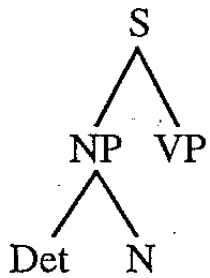
2.



Stack

[NP, VP]

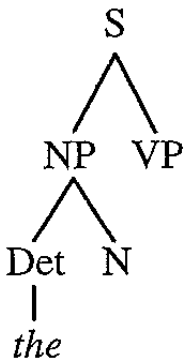
3.



Stack

[Det, N, VP]

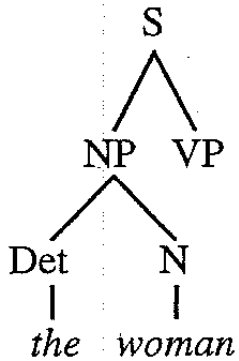
4.



Stack

[N, VP]

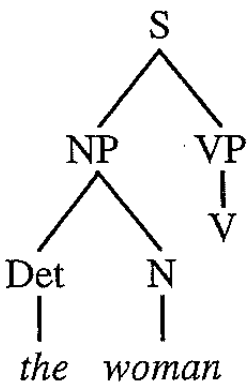
5.



Stack

[VP]

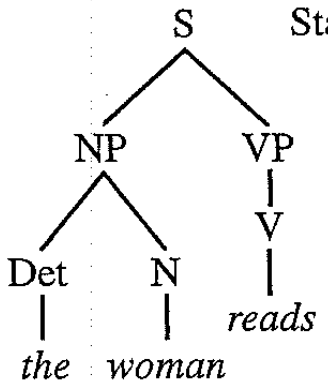
6.



Stack

[V]

7.



Stack

[]

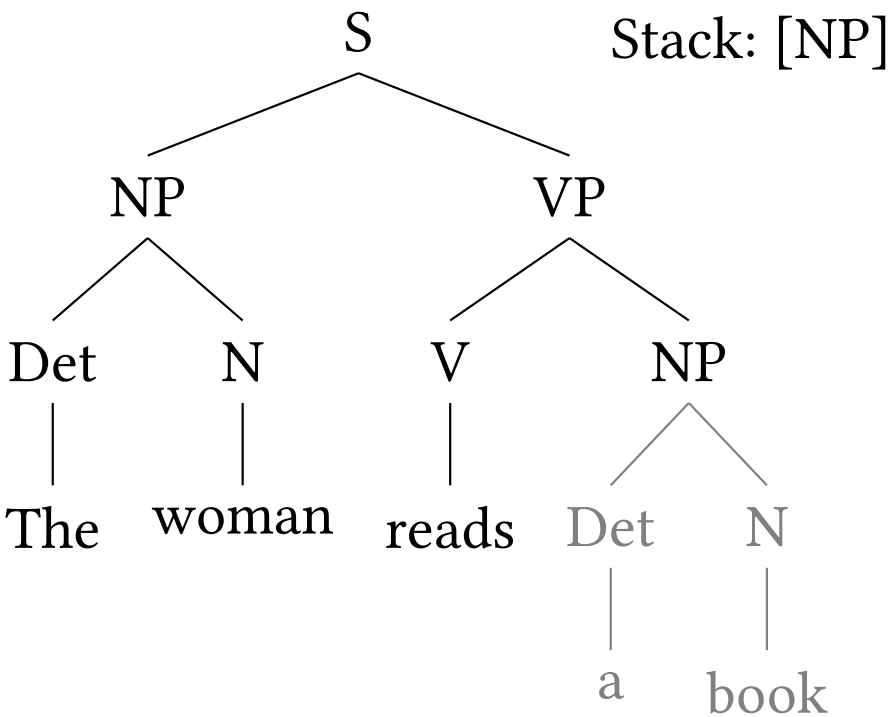
Parsingalgorithmen (Crocker): Top-Down

(14)
1. Initialise Stack = [S]
2. If top element of the stack is a non-terminal N, then:
   - Select a rule which rewrites N → R (where R is the symbol(s) on the right side of the rule)
   - Remove N from the stack
   - Add R to the top of the stack
3. If the top element of the stack is a pre-terminal P, then:
   - Find the next word W in the sentence
   - If there is a rule which rewrites P → W then: remove the pre-terminal from the stack
   - Else fail
4. If there are no more words to parse, then:
   - If the Stack = [ ], then done
5. Go to step 2.

Parsingalgorithmen (Crocker): Top-Down

Eigenschaften:

- nicht-deterministisch ($NP \rightarrow PN$, $NP \rightarrow Det\ N$)
- nicht "input-driven"
- immer strikt inkrementell:



Parsingalgorithmen (Crocker): Left-Corner

Ziel:

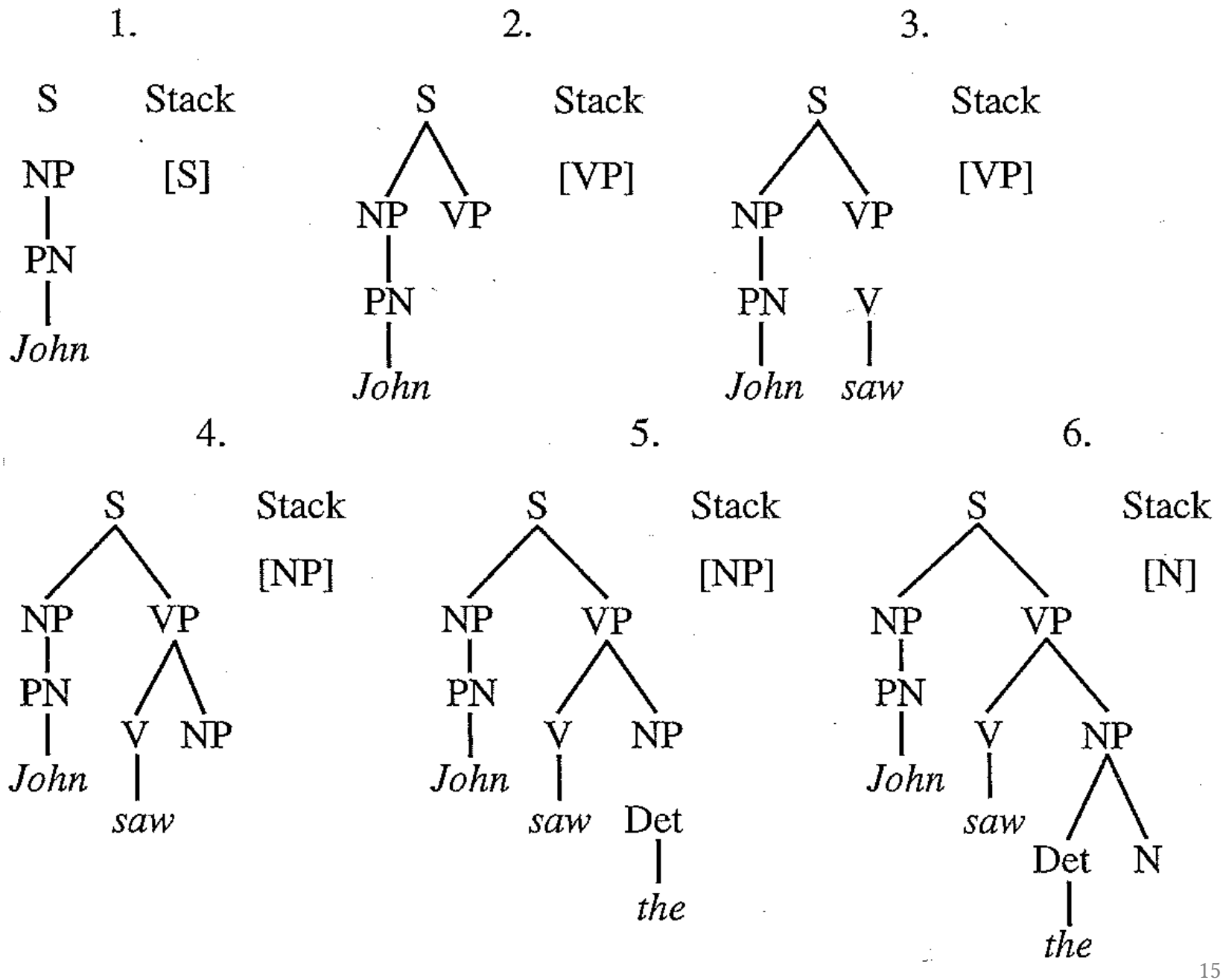
- inkrementeller als bottom-up
- “input-driven”

Idee:

- Die **linke Ecke** (“left corner”) einer Produktion ist das erste Symbol auf der rechten Seite: $A \rightarrow \boxed{B}\ C\ D$
- Die **linke Ecke** einer Produktion (B) wird bottom-up geparst, der Rest der Produktion ($C\ D$) top-down.
- Mit anderen Worten: Die Produktion $A \rightarrow B\ C\ D$ kann erst benutzt werden, wenn B schon geparst worden ist.

Parsingalgorithmen (Crocker): Left-Corner

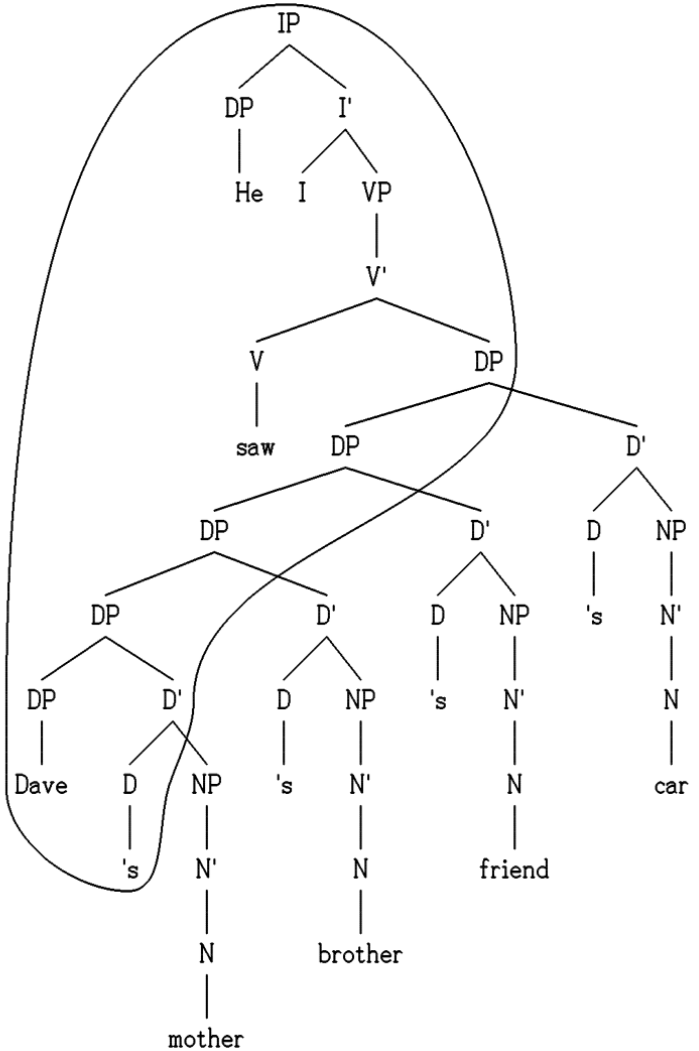
(15)



Parsingalgorithmen (Crocker): Left-Corner

Eigenschaften

- nicht-deterministisch ($VP \rightarrow V$, $VP \rightarrow V\ NP$)
- “input-driven”
- nicht immer strikt inkrementell:



(aus Stabler 1994, Fig. 6)

Parsingalgorithmen (Crocker): Left-Corner

Eigenschaften

- nicht-deterministisch ($VP \rightarrow V$, $VP \rightarrow V\ NP$)
- "input-driven"
- nicht immer strikt inkrementell
- Passt am besten zur Gedächtnislast ("memory load") bei Selbsteinbettungen ("center embeddings"):

(18) The mouse that the cat that the dog chased bit died.

$\Rightarrow$ Johnson-Laird (1983)

Parsingalgorithmen (Crocker): Weitere Fragen

Was tun bei Ambiguität?

- serielle Ansätze [“depth-first”]
  - immer genau eine Analyse
  - “choice points” bei lokaler Ambiguität
  - Backtracking und Reanalyse bei Fehlern
- Determinisierung
  - mittels Look-Ahead
  - nicht strikt inkrementell
  - Marcus parser (Marcus, 1980), ein LR(3)-Parser
- parallele Ansätze [“breadth-first”]
  - mehrere Analysen gleichzeitig
  - unbeschränkt $\Rightarrow$ inkompatibel mit Garden-Path-Effekt
  - beschränkt:
    - kurzzeitiger Parallelismus (Altmann, 1988)
    - gewichtete Analysen (Gibson, 1991)
    - aktivierungsbasierter Parallelismus (MacDonald et al., 1994)

Andere Regelformen und Parsingalgorithmen

Chomsky-Normalform (CNF) $\rightsquigarrow$ Crocker (1999)

$A \to B\ C \,|\, a$ (mit $A, B, C \in N$, $a \in T$)

Greibach-Normalform (GNF)

$A \to a\ \beta \,|\, a$ (mit $A \in N$, $a \in T$, $\beta \in N^*$)

gespiegelte GNF

$A \to \beta\ a \,|\, a$ (mit $A \in N$, $a \in T$, $\beta \in N^*$)

lineare CFG

$A \to a\ B \,|\, B\ a \,|\, a$ (mit $A, B \in N$, $a \in T$)

Giftschrank-Regeln:

- ϵ-Regeln
- rekursive, unäre Regeln (“loops”)
- unerreichbare Regeln

Andere Regelformen und Parsingalgorithmen

	CNF	GNF	gespiegelte GNF	lineare CFG
top-down	+	+	−	−
bottom-up	−	−	−	+
left-corner	−	+	−	+
Earley-Parser	+	+	+	+
PDA	+	+	+	+

Was macht der Earley-Parser anders?

- breadth-first, “bottom-up with top-down component” (Grune & Jacobs, 2008, 206)
- Benutzt “dotted productions”, d.h. er durchläuft die CFG-Produktionen schrittweise.

  Beispiel: $A \to B\ C\ a$
  $[A \to \bullet B\ C\ a] \Rightarrow [A \to B \bullet C\ a] \Rightarrow [A \to B\ C \bullet a] \Rightarrow [A \to B\ C\ a \bullet]$

Andere Regelformen und Parsingalgorithmen

	CNF	GNF	gespiegelte GNF	lineare CFG
top-down	+	+	−	−
bottom-up	−	−	−	+
left-corner	−	+	−	+
Earley-Parser	+	+	+	+
PDA	+	+	+	+

Was machen Push-Down-Automaten (Kellerautomaten, PDA) anders?

- Produktionen operieren auf dem Stack: (Hopcroft et al., 2001)
  - Für jedes Nichtterminal A gibt es Übergänge $\delta(q, \epsilon, A) = \{(q, \beta) | A \to \beta\}$
  - Für jede Terminal a gibt es einen Übergang $\delta(q, a, a) = \{(q, \epsilon)\}$
- Entspricht dem Top-Down-Ansatz (LL), allerdings werden die Produktionen vom Input-String strikt getrennt.

Zusammenfassung

- Der Definition einer CFG kommen Top-Down-Ansatz und Bottom-Up-Ansatz am nächsten.
- Der Left-Corner-Ansatz versucht, deren Vorteile zu kombinieren (strikte Inkrementalität und Input-Nähe). Trotzdem kann nicht jeder String bei jeder Grammatik strikt inkrementell geparst werden.
- Strikte Inkrementalität bei allen String-Grammatik-Paaren erziehlt man möglicherweise nur, wenn man die Produktionen vom Inputstring abstrahiert (Earley-Parser, PDA).

Weder Grammatikformalismen noch Grammatiken sind an sich (nicht-)inkrementell, sondern Grammatiken in Zusammenspiel mit bestimmten Parsingalgorithmen.

Die nächste Sitzung

- **Thema**: Generative Grammatiken (TG,GB,MG)
- **Texte**:
  - Ferreira (2005)
- **Fragen:**
  - Kann man die Varianten der Generativen Grammatik inkrementell parsen?
  - Inwiefern sind sie psychlinguistisch (un-)plausibel?

Altmann, Gerry. 1988. Ambiguity, parsing strategies, and computational models. *Language and Cognitive Processes* 3(2). 73–97.

Crocker, Matthew W. 1999. Mechanisms for sentence processing. In Simon Garrod & Martin J. Pickering (eds.), *Language processing*, 191–232. Hove, UK: Psychology Press.

Ferreira, Fernanda. 2005. Psycholinguistics, formal grammars, and cognitive science. *The Linguistic Review* 22. 365–380.

Gibson, Edward. 1991. *A computational theory of human linguistic processing: Memory limitations and processing breakdown*: Carnegie Mellon University Doctoral dissertation.

Grune, Dick & Ceriel J. Jacobs. 2008. *Parsing techniques: a practical guide* Monographs in Computer Science. New York: Springer 2nd edn.

Hopcroft, John E., Rajeev Motwani & Jeffrey D. Ullman. 2001. *Introduction to automata theory, languages and computation.* Addison-Wesley.

Johnson-Laird, Philip N. 1983. *Mental models.* Cambridge, UK: Cambridge University Press.

MacDonald, Maryellen C., Neal J. Pearlmutter & Mark S. Seidenberg. 1994. The lexical nature of syntactic ambiguity resolution. *Psychological Review* 101(4). 676–703.

Marcus, Mitchell P. 1980. *A theory of syntactic recognition for natural language.* Cambridge, MA: MIT Press.

Stabler, Edward P. 1994. The finite connectivity of linguistic structure. In C. Clifton, L. Frazier & K. Rayner (eds.), *Perspectives on sentence processing*, 303–336. Lawrence Erlbaum Associates.

Sturt, Patrick & Vicenzo Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science* 29(2). 291–305.