



UPPSALA  
UNIVERSITET

# Earley-algoritmen (kanonisk)

5LN449 Parsningsalgoritmer

Marco Kuhlmann

Institutionen för lingvistik och filologi

Uppsala universitet

2009-05-14





UPPSALA  
UNIVERSITET

# Översikt

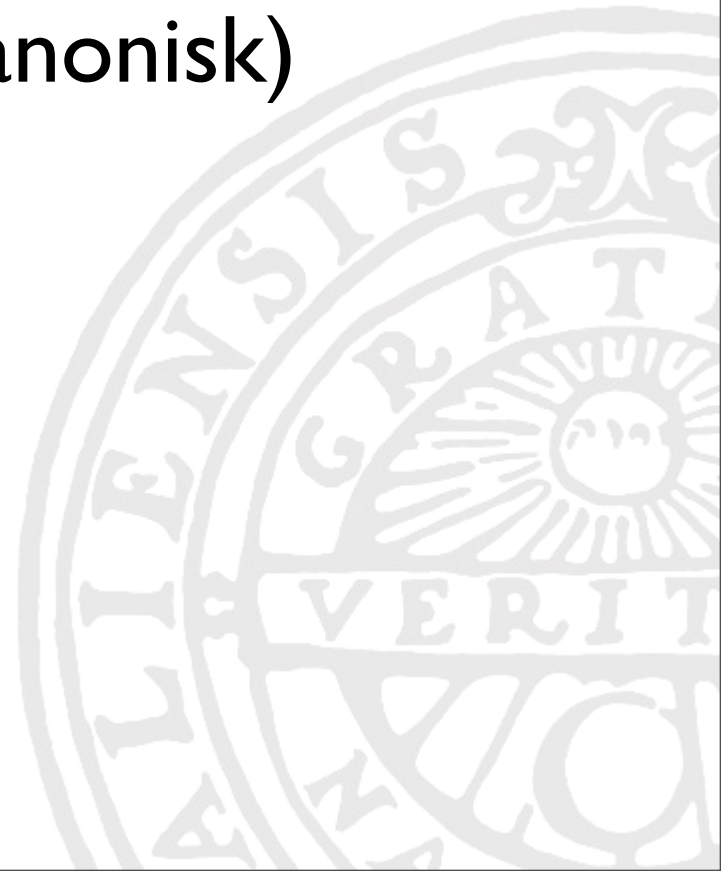
- Repetition: Earley-algoritmen (kanonisk)
- Hur man räknar ut parseträd





UPPSALA  
UNIVERSITET

# Repetition: Earley-algoritmen (kanonisk)





# Axiom

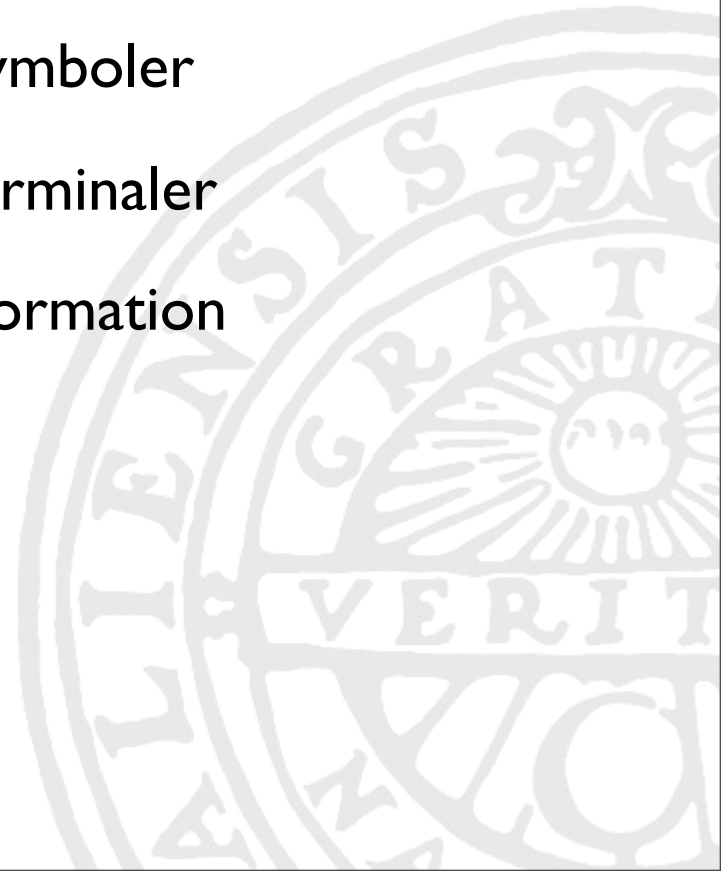
- Axiomen har formen  $[S \rightarrow \bullet \beta, 0, 0]$ ,  
för godtyckliga regler  $S \rightarrow \beta$ .  
**Notera att  $S$  är startsymbolen!**
- I motsats till bottom-up varianten börjar vi inte med vilken regel som helst, utan bara med regler som kan leda till analyser av typ  $S$ .
- I motsats till bottom-up varianten börjar vi inte var som helst, utan bara i början av ordföljden.



# Härledningsregler

Samma härledningsregler som i bottom-up fall,  
och en ny regel:

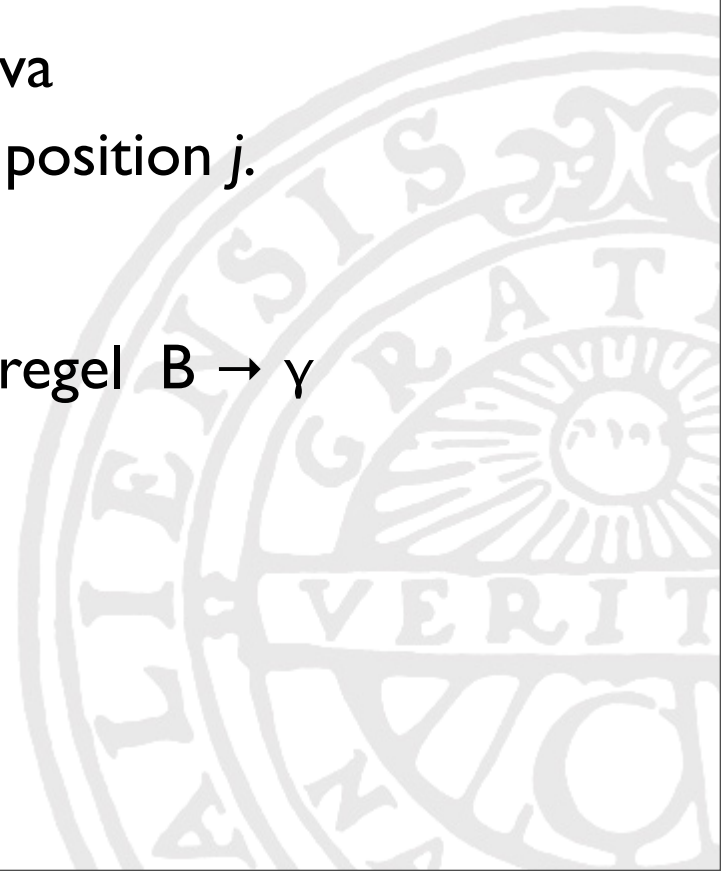
- **scan** behandlar terminala symboler
- **complete** behandlar icke-terminaler
- **predict** tillför top-down information





# Predict

- **Premiss:**  $[A \rightarrow \alpha \cdot B \beta, i, j]$
- För att flytta punkten i detta item kommer vi så småningom behöva en analys av typ B som börjar i position  $j$ .
- **Slutsats:**  $[B \rightarrow \cdot \gamma, j, j]$  ,  
om grammatiken innehåller en regel  $B \rightarrow \gamma$





UPPSALA  
UNIVERSITET

# Hur man räknar ut parseträd





# Parsning är mer än igenkänning

- I praktiska applikationer vill man inte bara veta om en ordföljd är grammatisk eller ej, utan också den konkreta syntaktiska analysen.  
*Exempel:* semantisk konstruktion, översättning
- Hittills är våra algoritmer bara igenkänningsalgoritmer, inte parsningsalgoritmer.



# Komma ihåg hur man har gjort

- Hittills kommer vi bara ihåg *vad* vi har bevisat, men inte *hur* vi har bevisat detta.
- Denna information behövs för att bygga parseträd.
- För att kunna bygga dessa behöver vi spara inte bara konklusionen av en regelanvändning, utan också premisserna och regelns namn.  
→ *jämför med matematiska bevis*



# Exempel: CKY

## Grammatik:

$$S \rightarrow a \mid S S$$

## Input:

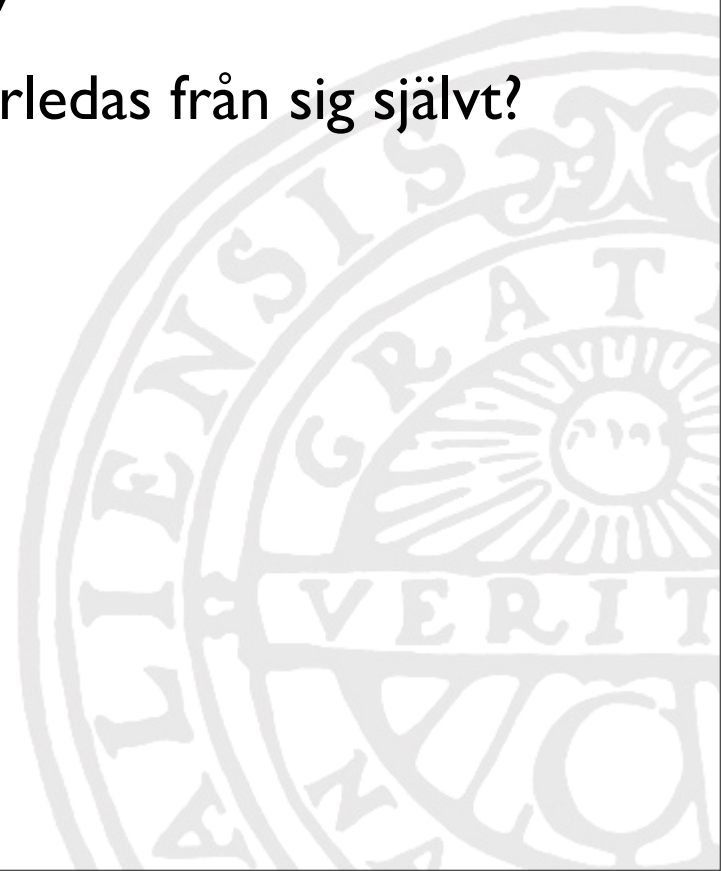
a a a

Skriv ner alla items som produceras för de här invärden. Om ett item är ett axiom, markera det med A. Rita annars pilar som det aktuella item kan härledas ifrån.



# Frågor

- Hur många parseträd finns det?
- Hur kan vi bygga dessa på ett systematiskt sätt?
- Vad händer om ett item kan härledas från sig självt?





# Sammanfattning (I): CKY

- CKY-algoritmen är en bottom-up parsningsalgoritm för kontextfria grammatiker.
- Minnet den behöver växer med grammatikens storlek (lineärt) och strängens längd (kvadratisk).
- CKY-algoritmen (som all chartparsning) använder inte bakåtpårning.
- CKY-algoritmen kan implementeras som ett deduktionssystem.



# Sammanfattning (2): Earley

- Earley-algoritmen är en parsningsalgoritm för kontextfria grammatiker.
- I motsats till CKY är den lämpad för godtyckliga grammatiker och använder top-down information.
- Minnet den behöver växer med grammatikens storlek (lineärt) och strängens längd (kvadratisk).
- Earley-algoritmen kan implementeras i samma deduktiva ramverket som CKY.



## Sammanfattning (3)

- För många praktiska applikationer är vi intresserade i själva parseträden, inte bara i frågan om det finns ett träd för en given ordföljd.
- I vårt deduktiva ramverk kan vi bygga parseträd genom att minnas inte bara *vad* vi har kommit fram till, utan också *hur* vi har gjort det.