Grundlegende Parsingalgorithmen

Top-Down & Bottom-Up Parsing

Kurt Eberle

k.eberle@lingenio.de

(Viele Folien, Teile von Folien, Materialien von **Helmut Schmid**'s Parsing-Kurs WS14 Tübingen, u.a.)

24. Februar 2020

Überblick

Überblick

Classification of Parsing Methods

- top-down vs. bottom-up
- derivation-oriented vs. table-driven vs. chart-based

Top-Down Parser

Idea: Systematically enumerate all left-most derivations until the input string has been derived.

Left-most derivation: The left-most non-terminal is expanded in each step.

Input: *a n v a n*

Top-Down Parser: Example

Input: *a n v a n*

Grammar:

S → NP VP
NP → a n
VP → v NP NP
VP → v NP
VP → v

Left-most derivation:

S ⇒ NP VP
⇒ *a n* VP
⇒ *a n v* NP NP
⇒ *a n v a n* NP
⇒ *a n v a n a n* (go 3 steps back)
⇒ *a n v* NP
⇒ *a n v a n*

Other name: *recursive descent parser*

Top-Down Parser

- Non-deterministic regarding
  - the choice of the non-terminal
  - the choice of the rule
- Convention for NT selection: left-most derivation
- **Backtracking** in order to try different grammar rules

Formal Characterization TD

- Configuration:
  Pair (α, r), s.t. $\alpha \in (V \cup \Sigma)^*, r \in \Sigma^*$
  pair $=$ $<$ *sentential form, remaining string to be recognized* $>$
  Start configuration: (S, w)
- configuration transitions:
  - $(a\alpha, aw) \mapsto (\alpha, w)$
    (*"consumption"* of an expected terminal symbol)
  - $(A\beta, r) \mapsto (\alpha\beta, r)$ with $A \to \alpha \in P$
    (*Expansion*, left-most derivation step)
- End configuration: $(\varepsilon, \varepsilon)$ (complete parse found)

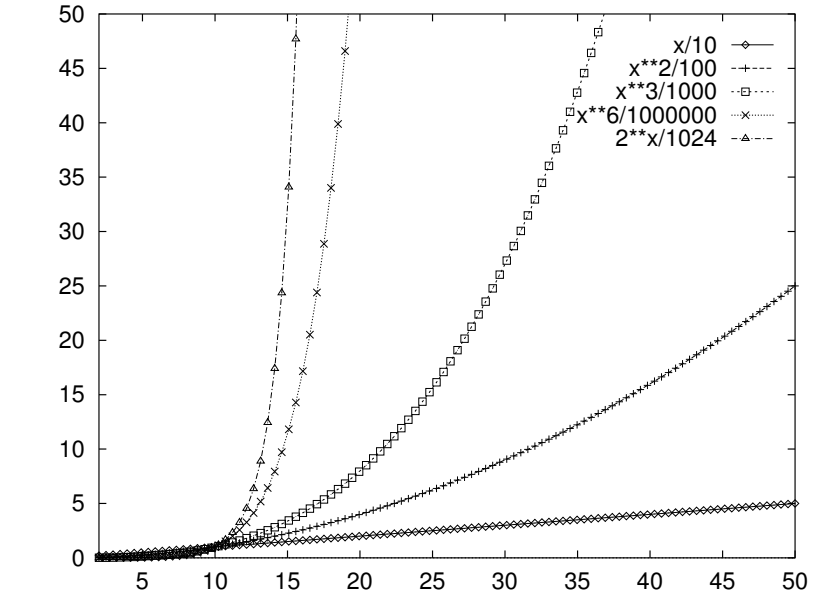
Configuration Transitions TD

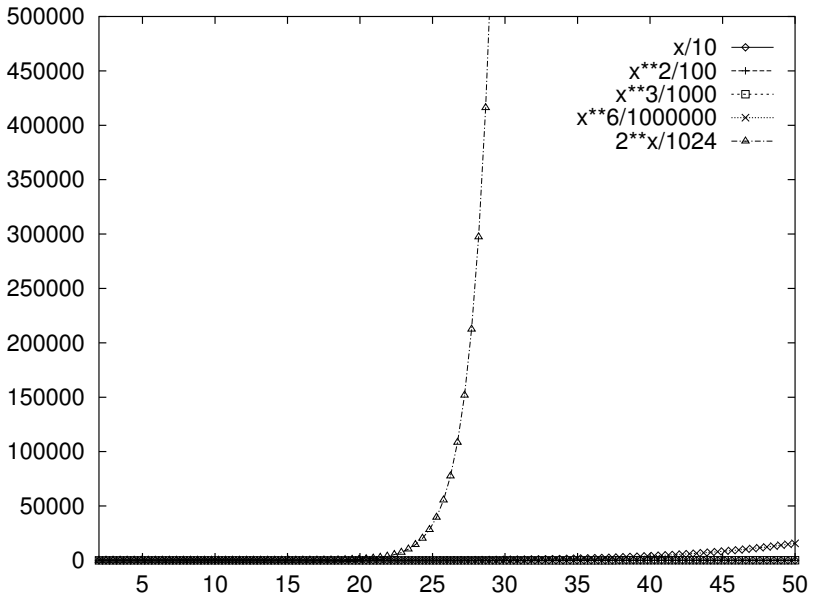
(S , *a n v a n a n*)
(NP VP , *a n v a n a n*)
(*a n* VP , *a n v a n a n*)
(*n* VP , *n v a n a n*)
(VP , *v a n a n*)
(*v* NP NP , *v a n a n*)
(NP NP , *a n a n*)
(*a n* NP , *a n a n*)
(*n* NP , *n a n*)
(NP , *a n*)
(*a n* , *a n*)
(*n* , *n*)
(ε , ε)

Problems of the TD Parser

- Left-recursive Non-terminals: $A \overset{+}{\Rightarrow} A\beta$
  Danger of an infinite loop
- Rule selection:
  "blind" expansion
- Inefficiency of Backtracking:
  Partial analyses are repeated causing an exponential runtime
- Advantage:
  easy to implement

Growth Curves





$2^{50}/1024$ seconds $\approx$ 35000 years

Überblick

Bottom-Up Parser

Idea: Backward application of grammar rules (reductions) produces an inverted right-most derivation.

Input: *a n v a n a n*

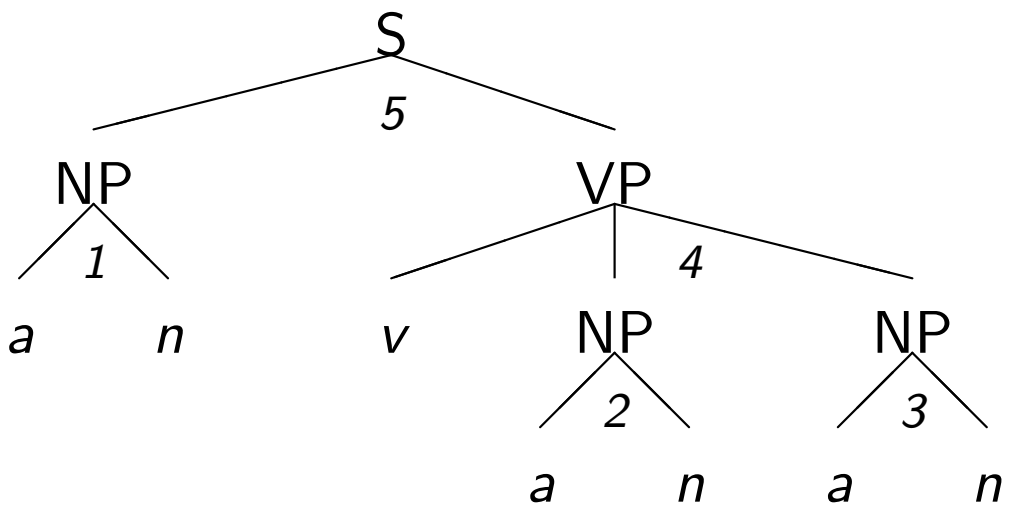
Grammar:

$$
\begin{aligned}
\mathrm{S} &\rightarrow \mathrm{NP\ VP} \\
\mathrm{NP} &\rightarrow \mathrm{a\ n} \\
\mathrm{VP} &\rightarrow \mathrm{v\ NP\ NP} \\
\mathrm{VP} &\rightarrow \mathrm{v\ NP} \\
\mathrm{VP} &\rightarrow \mathrm{v}
\end{aligned}
$$

Left-most Reduction:

$$
\begin{aligned}
\underline{a\ n}\ v\ a\ n\ a\ n &\Leftarrow \mathrm{NP}\ v\ \underline{a\ n}\ a\ n \\
&\Leftarrow \mathrm{NP}\ v\ \mathrm{NP}\ \underline{a\ n} \\
&\Leftarrow \mathrm{NP}\ \underline{v\ \mathrm{NP}\ \mathrm{NP}} \\
&\Leftarrow \underline{\mathrm{NP\ VP}} \\
&\Leftarrow \mathrm{S}
\end{aligned}
$$

Tree



Formal Charakterization BU

- Configuration:
  Pair (α, r), with $\alpha \in (V \cup \Sigma)^*, r \in \Sigma^*$
  pair $=$ $<$ *sentential form, remaining string to be recognized* $>$
- Start configuration: (ε, w)
- Configuration transitions:
  - $(\alpha, ar) \mapsto (\alpha a, r)$ *(Shift action)*
  - $(\beta\alpha, r) \mapsto (\beta A, r)$ with $A \to \alpha \in P$ *(Reduce action)*
- End configuration: (S, ε)

$\Rightarrow$ "Shift-Reduce"-Parser

Configuration Transitions BU

$$
\begin{aligned}
(\varepsilon \ , \ a\ n\ v\ a\ n\ a\ n) \\
(a \ , \ n\ v\ a\ n\ a\ n) \\
(a\ n \ , \ v\ a\ n\ a\ n) \\
(\mathrm{NP} \ , \ v\ a\ n\ a\ n) \\
(\mathrm{NP}\ v \ , \ a\ n\ a\ n) \\
(\mathrm{NP}\ v\ a \ , \ n\ a\ n) \\
(\mathrm{NP}\ v\ a\ n \ , \ a\ n) \\
(\mathrm{NP}\ v\ \mathrm{NP} \ , \ a\ n) \\
(\mathrm{NP}\ v\ \mathrm{NP}\ a \ , \ n) \\
(\mathrm{NP}\ v\ \mathrm{NP}\ a\ n \ , \ \varepsilon) \\
(\mathrm{NP}\ v\ \mathrm{NP}\ \mathrm{NP} \ , \ \varepsilon) \\
(\mathrm{NP}\ \mathrm{VP} \ , \ \varepsilon) \\
(\mathrm{S} \ , \ \varepsilon)
\end{aligned}
$$

Problems BU

- Rule cycles and ε productions may result in infinite loops.
- Rule selection:
  "Blind shift"
- Inefficiency of Backtracking