Automata and Formal Languages (7)

Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)
Slides: KEATS (also home work is there)

CFGs

A context-free grammar (CFG) G consists of:

- a finite set of nonterminal symbols (upper case)
- a finite terminal symbols or tokens (lower case)
- a start symbol (which must be a nonterminal)
- a set of rules

$$A \rightarrow rhs_1 | rhs_2 | \ldots$$

where rhs are sequences involving terminals and nonterminals (can also be empty).

CFGs

A context-free grammar (CFG) G consists of:

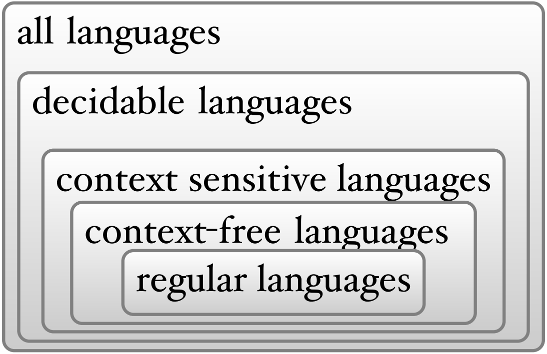
- a finite set of nonterminal symbols (upper case)
- a finite terminal symbols or tokens (lower case)
- a start symbol (which must be a nonterminal)
- a set of rules

$$A \rightarrow \mathrm{rhs}_1 | \mathrm{rhs}_2 | \ldots$$

where rhs are sequences involving terminals and nonterminals (can also be empty).

Hierarchie of Languages

Recall that languages are sets of strings.



Arithmetic Expressions

A grammar for arithmetic expressions and numbers:

$$\begin{array}{lcl} \boldsymbol{E} & \rightarrow & \boldsymbol{E} \cdot + \cdot \boldsymbol{E} \mid \boldsymbol{E} \cdot * \cdot \boldsymbol{E} \mid (\cdot \boldsymbol{E} \cdot) \mid \boldsymbol{N} \\ \boldsymbol{N} & \rightarrow & \boldsymbol{N} \cdot \boldsymbol{N} \mid 0 \mid 1 \mid \ldots \mid 9 \end{array}$$

Unfortunately left-recursive (and ambiguous).

A problem for recursive descent parsers
(e.g. parser combinators).

Arithmetic Expressions

A grammar for arithmetic expressions and numbers:

$$\begin{array}{lcl} \boldsymbol{E} & \rightarrow & \boldsymbol{E} \cdot + \cdot \boldsymbol{E} \mid \boldsymbol{E} \cdot * \cdot \boldsymbol{E} \mid (\cdot \boldsymbol{E} \cdot) \mid \boldsymbol{N} \\ \boldsymbol{N} & \rightarrow & \boldsymbol{N} \cdot \boldsymbol{N} \mid 0 \mid 1 \mid \ldots \mid 9 \end{array}$$

Unfortunately left-recursive (and ambiguous).

A problem for recursive descent parsers (e.g. parser combinators).

Numbers

$$N \;\rightarrow\; N \cdot N \mid 0 \mid 1 \mid \ldots \mid 9$$

A non-left-recursive, non-ambiguous grammar for numbers

$$N \;\rightarrow\; 0 \cdot N \mid 1 \cdot N \mid \ldots \mid 0 \mid 1 \mid \ldots \mid 9$$

Operator Precedences

To disambiguate

$$E \;\rightarrow\; E \cdot + \cdot E \mid E \cdot * \cdot E \mid (\cdot E \cdot) \mid N$$

Decide how many precedence levels, say

highest for $()$, medium for $*$, lowest for $+$

$$\begin{array}{lcl} E_{low} & \rightarrow & E_{med} \cdot + \cdot E_{low} \mid E_{med} \\ E_{med} & \rightarrow & E_{hi} \cdot * \cdot E_{med} \mid E_{hi} \\ E_{hi} & \rightarrow & (\cdot E_{low} \cdot) \mid N \end{array}$$

Operator Precedences

To disambiguate

$$E \rightarrow E \cdot + \cdot E \mid E \cdot * \cdot E \mid (\cdot E \cdot) \mid N$$

Decide how many precedence levels, say

highest for $()$, medium for $*$, lowest for $+$

$$\begin{array}{lcl} E_{low} & \rightarrow & E_{med} \cdot + \cdot E_{low} \mid E_{med} \\ E_{med} & \rightarrow & E_{hi} \cdot * \cdot E_{med} \mid E_{hi} \\ E_{hi} & \rightarrow & (\cdot E_{low} \cdot) \mid N \end{array}$$

What happens with $1 + 3 + 4$?

Removing Left-Recursion

The rule for numbers is directly left-recursive:

$$N \;\rightarrow\; N \cdot N \mid 0 \mid 1 \quad (\ldots)$$

Translate

$$\begin{array}{rcl} N & \rightarrow & N \cdot \alpha \\ & \mid & \beta \end{array} \quad\Rightarrow\quad \begin{array}{rcl} N & \rightarrow & \beta \cdot N' \\ N' & \rightarrow & \alpha \cdot N' \\ & \mid & \epsilon \end{array}$$

Removing Left-Recursion

The rule for numbers is directly left-recursive:

$$N \;\rightarrow\; N \cdot N \mid 0 \mid 1 \quad (\ldots)$$

Translate

$$\begin{array}{lcl} N & \rightarrow & N \cdot \alpha \\ & \mid & \beta \end{array} \quad\Rightarrow\quad \begin{array}{lcl} N & \rightarrow & \beta \cdot N' \\ N' & \rightarrow & \alpha \cdot N' \\ & \mid & \epsilon \end{array}$$

Which means

$$\begin{array}{lcl} N & \rightarrow & 0 \cdot N' \mid 1 \cdot N' \\ N' & \rightarrow & N \cdot N' \mid \epsilon \end{array}$$

Chomsky Normal Form

All rules must be of the form

$$A \rightarrow a$$

or

$$A \rightarrow B \cdot C$$

No rule can contain ϵ.

ϵ-Removal

1. If $A \rightarrow \alpha \cdot B \cdot \beta$ and $B \rightarrow \epsilon$ are in the grammar, then add $A \rightarrow \alpha \cdot \beta$ (iterate if necessary).
2. Throw out all $B \rightarrow \epsilon$.

$$\begin{aligned} N &\rightarrow 0 \cdot N' \mid 1 \cdot N' \\ N' &\rightarrow N \cdot N' \mid \epsilon \end{aligned} \qquad \begin{aligned} N &\rightarrow 0 \cdot N' \mid 1 \cdot N' \mid 0 \mid 1 \\ N' &\rightarrow N \cdot N' \mid N \mid \epsilon \end{aligned}$$

CYK Algorithm

$$\begin{array}{lcl} \boldsymbol{S} & \rightarrow & \boldsymbol{N} \cdot \boldsymbol{P} \\ \boldsymbol{P} & \rightarrow & \boldsymbol{V} \cdot \boldsymbol{N} \\ \boldsymbol{N} & \rightarrow & \boldsymbol{N} \cdot \boldsymbol{N} \\ \boldsymbol{N} & \rightarrow & \text{students} \mid \text{Jeff} \mid \text{geometry} \mid \text{trains} \\ \boldsymbol{V} & \rightarrow & \text{trains} \end{array}$$

Jeff trains geometry students

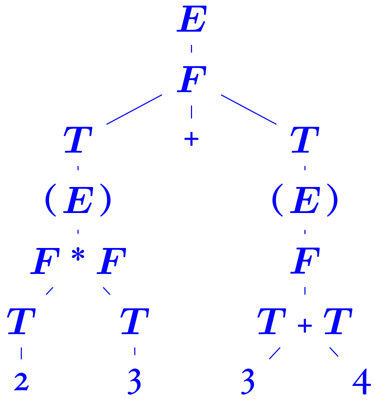
CYK Algorithm

- runtime is $O(n^3)$
- grammars need to be transferred into CNF

Parse Trees

$$
\begin{array}{lcl}
E & \rightarrow & F \mid F \cdot * \cdot F \\
F & \rightarrow & T \mid T \cdot + \cdot T \mid T \cdot - \cdot T \\
T & \rightarrow & num\_token \mid (\cdot E \cdot)
\end{array}
$$

$(2*3)+(3+4)$



Ambiguous Grammars

A CFG is ambiguous if there is a string that has at least parse trees.

$$
\begin{array}{lcl}
E & \rightarrow & num\_token \\
E & \rightarrow & E \cdot + \cdot E \\
E & \rightarrow & E \cdot - \cdot E \\
E & \rightarrow & E \cdot * \cdot E \\
E & \rightarrow & (\cdot E \cdot)
\end{array}
$$

$1 + 2 * 3 + 4$

Dangling Else

Another ambiguous grammar:

$$\begin{array}{lcl} E & \rightarrow & \text{if } E \text{ then } E \\ & | & \text{if } E \text{ then } E \text{ else } E \\ & | & \text{id} \end{array}$$

$\text{if a then if x then y else c}$

A CFG Derivation

1. Begin with a string with only the start symbol S
2. Replace any non-terminal X in the string by the right-hand side of some production $X \rightarrow \mathrm{rhs}$
3. Repeat 2 until there are no non-terminals

$$S \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots$$