Notes inserted in the green book SPM in Section 6.3.1: Summary on $LR(0)$ and $LR(1)$ Parsing. 20-05-2016

# PARSING

**1. *General context-free parsing.***

Chomsky normal form:

$S \to \varepsilon \qquad A \to BC \qquad A \to a$

Cocke-Younger-Kasami parser in Chomsky Normal Form: $O(n^3)$ time (Dynamic Programming)
(Actually, same complexity of matrix multiplication: Valiant's result.)
Earley parser for context-free languages: $O(n^3)$ time

**2. *Chop-Expand.*** Parsing non-left recursive context-free grammars.

• *Nondeterministic Parsing*

- for Context-free grammars. *existsev* $p$ $f$ $L$ : higher order can be avoided if $p$ and $f$ do not depend on the node being visited.
  Tail recursive program which keeps the list of the *frontier nodes* to be visited.
  Chop-expand parser (by Burstall-Dijkstra) [3, pages 35–49].
- for Regular grammars. Backtracking as do-while and recursion.
  (1) do-while is avoided in favour of tail recursion.
  (2) recursion is implemented by keeping (as a stack) the list of the *ancestor nodes.*
  Regular grammar parser (program by me) (ATFL) [2, page 87].

• *Deterministic Parsing* with lookahead: $O(n)$ parsing

| | |
|---|---|
| $LR(1)$: | deterministic context-free languages (and $LALR(1)$ parsing) |
| $LR(0)$: | prefix-free, deterministic context free |
| $LL(1)$: | non left-recursive grammars. chop-expand parsing. |
| context-free: | recursive descent parsing: bottom-up deterministic (see the Propositional Theorem Prover [3, page 172]) |
| regular: | deterministic finite automaton for regular grammars [2, page 79] |

A language $L$ enjoys the *prefix property* (or it is *prefix-free*) iff no word in $L$ is a proper prefix of another word in $L$.

• *operator-precedence grammar parsing*
Every context-free language $L$ is such that $L-\{\varepsilon\}$ can be generated by an operator-precedence grammar.

**3. Rosenkrantz-Stearns' result.**

$$\begin{array}{l}
LR(1) \ = \text{deterministic context-free languages} \\
\ \cup \\
\ \vdots \\
\ \cup \\
LL(k) \\
\ \cup \\
\ \vdots \\
\ \cup \\
LL(2) \\
\ \cup \\
LL(1) \\
\ \cup \\
LL(0) \quad \text{(either empty or singleton languages)}
\end{array}$$

We have: $LL(0) \ \subset \ LR(0)$ (= prefix-free, deterministic context-free languages) $\subset \ LR(1)$

- For all $k \geq 1$,

$$\begin{array}{ll} S \to a\,T & \\ T \to S\,A & \mid A \\ A \to b\,B & \mid c \\ B \to b^{k-1}\,d & \mid \varepsilon \end{array} \quad \text{is an } LL(k) \text{ grammar and } \textit{not} \text{ an } LL(k-1) \text{ grammar.}$$

- For all $k \geq 1$,
  $\{a^n\, w \mid n \geq 1 \text{ and } w \in \{b, c, b^k, d\}^n\}$ is an $LL(k)$ language and it is *not* an $LL(k-1)$ language.
- A language is $LL(0)$ iff it is empty or it is a singleton.

If we assume that in the grammars there are no useless symbols, then a language is $LL(0)$ iff it is a singleton.

---

Note that we do not define the parsing tables for $LL(0)$ parsing.
The following two examples show how to construct the parsers for $LL(0)$ languages.

**Example 1.** Given the alphabet $\Sigma = \{a, b\}$, the algorithm for accepting the $LL(0)$ language which is empty, is any finite automaton without final states (see Figure 1.1).

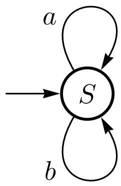


Figure 1.1: A finite automaton accepting the empty language. $S$ is not a final state.

Given the alphabet $\Sigma = \{a, b\}$, the algorithm for accepting the $LL(0)$ language which is the singleton $\{abaa\}$, is a finite automaton with a sequence of states, no cycles and exactly one final state (see Figure 1.2). There are $n+1$ states in the sequence if $n$ is the length of the word in the singleton.

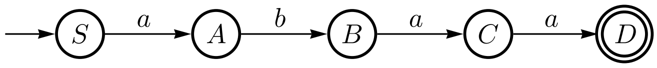


Figure 1.2: The finite automaton accepting the word $a\,b\,a\,a$ only.

---

- For all $k \geq 1$, every $LR(k)$ language is an $LR(1)$ language. That is, for every $k \geq 1$, for every $LR(k)$ language $L$ (that is, for every language $L$ generated by an $LR(k)$ grammar), there exists an $LR(1)$ grammar which generates $L$.
- For all $k \geq 0$, there are $LR(k+1)$ grammars which are *not* $LR(k)$ grammars.
- For all $k \geq 0$,

$$\begin{array}{ll} S \to a\,b^k\,c & \mid A\,b^k\,d \\ A \to a & \end{array} \quad \text{is an } LR(k+1) \text{ grammar and it is not an } LR(k) \text{ grammar.}$$

---

$\boxed{LR(0) \textbf{ and } LR(1) \textbf{ PARSING}}$

• A language $L$ is deterministic context-free, that is, it is parsable by a deterministic pda (dpda, for short) with acceptance by final state,
*iff* $L$ is an $LR(1)$ language
*iff* $L\$$, with $\$ \notin V_T$, is an $LR(0)$ language.

For a deterministic pda, acceptance *by final state* is more powerful than acceptance *by empty stack.*

• Every deterministic context-free language $L$ which enjoys the prefix property is recognized by a dpda by final state,
*iff* $L$ is a language recognized by a dpda by empty stack
*iff* $L$ is an language $LR(0)$.

• $D = \{0^i\, 1^k\, a\, 2^i \mid i,k \geq 1\} \cup \{0^i\, 1^k\, b\, 2^k \mid i,k \geq 1\}$ is a deterministic context-free language
*and* every grammar for $D$ in Greibach normal form must have at least two productions of the form: $A \to a\,\alpha$ and $A \to a\,\beta$, with $\alpha \neq \beta$,
*and* the dpda which accepts by final state should make at least an $\varepsilon$-move.

We can always take this dpda such that if it has to make an $\varepsilon$-move, then it makes that $\varepsilon$-move while the input is not completely read. This follows from a theorem holding for any dpda which: (i) accepts a language by final state, and (ii) should perform an $\varepsilon$-move [2].

Note that the language $D$ enjoys the prefix property (it is in zone (B) of Figure 1.3).
A context-free grammar which generates the language $D$ has axiom $S$ and the following productions:

$$\begin{array}{llllll} S \to 0\,L\,T \mid 0\,R & & L \to 0\,L\,T \mid 1\,A & & R \to 0\,R \mid 1\,B\,T \\ T \to 2 & & A \to 1\,A \mid a & & B \to 1\,B\,T \mid b \end{array}$$

• The language $\{a^n b^n \mid n > 0\}$ generated by the grammar with axiom $S$ and the following productions:
$S \to a\,S\,b \mid a\,b$
is a prefix-free deterministic context-free language (it is in zone (B) of Figure 1.3).

• The language $D \cup \{c,\ c\,c\}$ is a deterministic context-free language, but it is not prefix-free (it is in zone (A) of Figure 1.3).

The grammar with axiom $S$ and the following productions: $S \to a\,S\,b \mid \varepsilon$
generates the deterministic context-free language $\{a^n b^n \mid n \geq 0\}$ which is not prefix-free (it is in zone (A) of Figure 1.3).

• The language $\{0\,w\,w^R\,\$\,0 \mid w \in \{0,1\}^*\} \cup \{1\,w\,w^R\,\$\,1 \mid w \in \{0,1\}^*\}$, where by $w^R$ we denote the reverse of $w$, generated by the grammar with axiom $S$ and the following productions:

$$S \to 0\,A\,\$\,0 \mid 1\,A\,\$\,1 \qquad\qquad A \to 0\,A\,0 \mid 1\,A\,1 \mid \varepsilon$$

is prefix-free, but it is not deterministic context-free (it is in zone (C) of Figure 1.3).

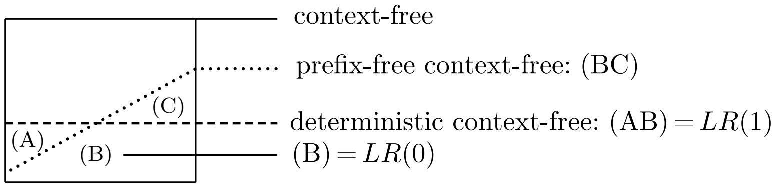


Figure 1.3: Deterministic context-free languages: (AB). Prefix-free context-free languages: (BC). Deterministic, prefix-free context-free languages: (B).

---

It is decidable whether or not a deterministic context-free language (given by a context-free grammar) is prefix-free [1, page 355].

It is undecidable whether or not a context-free language (given by a context-free grammar) is prefix-free [1, page 262].

The class of the deterministic context-free languages (see zones (AB) of Figure 1.3) is a proper superset of the class of the deterministic context-free languages which are prefix-free (see zone (A) of Figure 1.3). (Deterministic context-free languages which are prefix-free are also called strict deterministic context-free languages in [1, page 355–358].)

---

**Conventions for $LL(k)$ and $LR(k)$ parsing**

In Table 1 below we recall some hypotheses we made concerning the parsing of various kinds of $LL(k)$ and $LR(k)$ languages and, in particular:

(i) the use of a rightmost, new symbol \$ in the input string,
(ii) the use of augmented grammars with a new production for the axiom $S'$,
(iii) the initial configuration of the stack, and
(iv) the lookahead sets.

We have to consider augmented grammars for having the new axiom $S'$ not to occur on the right hand side of any production.

| | input string | augmented grammar | production of the axiom | initial configuration of the stack | lookahead set |
|---|---|---|---|---|---|
| $LL(k)$ | ended by \$ | no | axiom $S$ | $S\,\$$ ▲ | none |
| $LR(0)$ and $SLR(1)$ | ended by \$ | yes | axiom $S'$ <br> add: $S' \rightarrow S\,\$$ | $q_0$ ▲ | none |
| $LR(1)$ and $LALR(1)$ | ended by \$ | yes | axiom $S'$ <br> add: $S' \rightarrow S$ | $q_0$ ▲ | $\{\$\}$ |

Table 1: Our conventions on the input string, the augmented grammar with the production of the axiom, the initial stack configuration ($q_0$ is the initial state), and the lookahead set for various classes of context-free grammars.

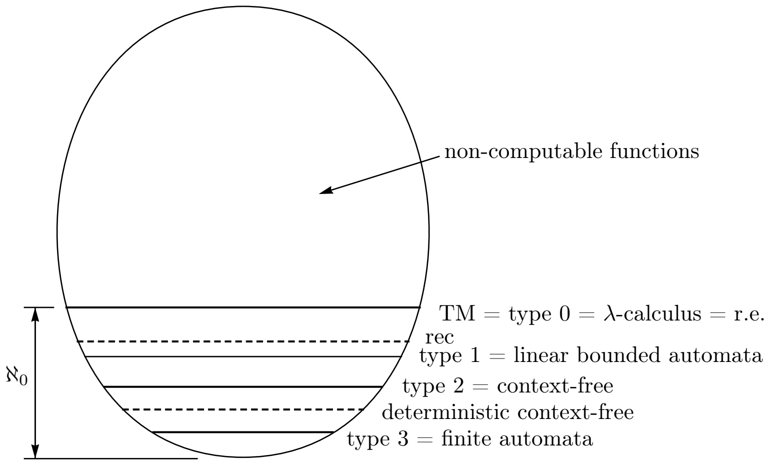


Figure 1.4: Let $N$ denote the set of the natural numbers. In this figure we show the set $N^N$ of the computable and non-computable functions from $N$ to $N$. The cardinality of $N^N$ is $\aleph_1$, which is the cardinality of the set of the real numbers. $\aleph_0$ is the cardinality of the set $N$ of the natural numbers.

# References

[1] M. A. Harrison. *Introduction to Formal Language Theory.* Addison Wesley, 1978.

[2] A. Pettorossi. *Automata Theory and Formal Languages.* Aracne Editrice, Fourth edition, 2013.

[3] A. Pettorossi. *Techniques for Searching, Parsing, and Matching.* Aracne Editrice, Third edition, 2011.