SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

FOUNDATIONS OF COMPUTER SCIENCE
LECTURE 10: Turing machines and Grammars

Prof. Daniele Gorla

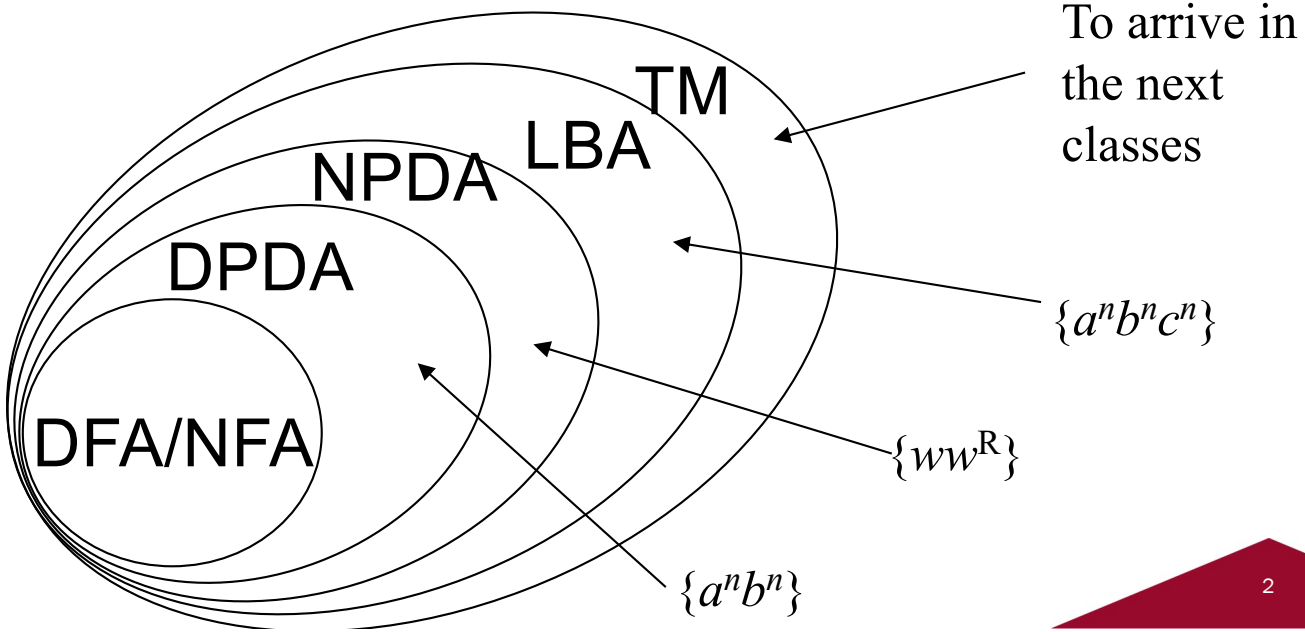
Acceptors for non-context-free languages

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

We're now ready to complete the following table:

Regular languages	*Regular Grammars (Type 3)*	*DFA/NFA*
CF languages	*CF grammars (Type 2)*	*(non-det)PDA*
CS languages	*CS grammars (Type 1)*	*LBA*
RE languages	*Type 0 grammars*	*TM*

This originates the following hierarchy, based on the power of acceptors:



From a Type 0 Grammar to a TM

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: for every type 0 grammar G, there exists an equivalent TM

Proof

Let's construct a nondeterministic two-tape TM M that recognizes $L(G)$.

M's first tape contains the input string w.

The second tape is used to hold derivations within G; it is initialized with the start variable of G

Let α be the current content of the second tape. Then, M repeatedly does the following:

1. Nondeterministically select a position $i \in \{1, \ldots, |\alpha|\}$ (i.e., start at the leftmost position of the second tape and repeatedly choose to move right or select the present position)
2. Nondeterministically select a production $\beta ::= \gamma$ of G
3. If β appears beginning in position i of α, replace β by γ there, by using a proper "shifting" if needed (towards left, if $|\beta| > |\gamma|$; towards right, if $|\beta| < |\gamma|$)
4. Compare the content of tape 2 with w (on tape 1):
   - If they coincide, accept
   - If not, go back to Step (1)

We can prove that $w \in L(G)$ if and only if $w \in L(M)$.

Q.E.D.

From a TM to a Type 0 grammar

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: for every TM M, there exists an equivalent type 0 grammar.

Proof

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$. Let us define G to be

$$(((\Sigma \cup \{\varepsilon\}) \times \Gamma) \cup Q \cup \{A_1, A_2, A_3\} \ , \ \Sigma \ , \ R \ , \ A_1)$$

Intuitively, variables are needed to mimick configurations of M that arise during its computation: the configuration $c_1..c_h q c_{h+1}..c_m$ arising from input $a_1..a_n$ is modeled by the sequence of variables

$$[w_1,c_1]\ldots[w_h,c_h]\ q\ [w_{h+1},c_{h+1}]\ldots[w_m,c_m][w_{m+1}, \sqcup]\ldots[w_{m+k}, \sqcup]$$

where $w_1 \ldots w_{m+k} = a_1..a_n \varepsilon \ldots \varepsilon$ (this contains the original input and never changes) and $m+k$ is the length of the longest string that ever appears on the tape of M with input $a_1..a_n$.

R is defined as follows:

1. $A_1 ::= q_0 A_2$
2. $A_2 ::= [a, a] A_2$, for each $a \in \Sigma$
3. $A_2 ::= A_3$
4. $A_3 ::= [\varepsilon, \sqcup] A_3$
5. $A_3 ::= \varepsilon$
6. $q\ [a, x] ::= [a, y]\ q'$, for each $a \in \Sigma \cup \{\varepsilon\}$, $q,q' \in Q$, $x,y \in \Gamma$ such that $\delta(q, x) = (q', y, \mathrm{R})$
7. $[b, z]\ q\ [a, x] ::= q'\ [b, z]\ [a, y]$, for each $a,b \in \Sigma \cup \{\varepsilon\}$, $q,q' \in Q$, $x,y,z \in \Gamma$ s.t. $\delta(q, x) = (q', y, \mathrm{L})$
8. $q_{accept}\ [a, x] ::= q_{accept}\ a\ q_{accept}$ and $[a, x]\ q_{accept} ::= q_{accept}\ a\ q_{accept}$, for each $a \in \Sigma \cup \{\varepsilon\}$, $x \in \Gamma$
9. $q_{accept} ::= \varepsilon$

From a TM to a Type 0 grammar

Now, if $a_1 \ldots a_n$ belongs to $L(M)$ and M uses at most t cells to the right of the input to accept it, consider the derivation (using rule 1, n times rule 2, rule 3, t times rule 4, and rule 5):

$$A_1 \Rightarrow^* q_0 [a_1 , a_1] \ldots [a_n , a_n] [\varepsilon, \sqcup] \ldots [\varepsilon, \sqcup]$$

This essentially produces the starting configuration of M (viz., $q_0 a_1 \ldots a_n$), with as many blank symbols after the input as needed by M to accept.

From this point on, only rules 6 and 7 can be used until q_{accept} is generated (by mimicking how M computes for accepting the input):

$$q_0 [a_1 , a_1] \ldots [a_n , a_n] [\varepsilon, \sqcup] \ldots [\varepsilon, \sqcup] \Rightarrow^* [a_1 , x_1] \ldots [a_h , x_h] q_{\text{accept}} [a_{h+1} , x_{h+1}] \ldots [a_{n+t} , x_{n+t}]$$

Note that the first components of variables in $(\Sigma \cup \{\varepsilon\}) \times \Gamma$ are never changed.

So, finally, we can use rules 8 and 9 to get rid of q_{accept} and yield a string of terminals:

$$[a_1 , x_1] \ldots [a_h , x_h] q_{\text{accept}} [a_{h+1} , x_{h+1}] \ldots [a_{n+t} , x_{n+t}] \Rightarrow^* a_1 \ldots a_n$$

That is, $a_1 \ldots a_n$ belongs to $L(G)$.

We should also show that every string produced by G is accepted by M (see the book).

Q.E.D.

Linear Bounded Automata

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Def.: A ***linear bounded automaton*** (LBA) is a nondeterministic TM satisfying the conditions:

1. Its tape alphabet Γ includes a special symbol \$, called *endmarker*; and
2. It has no moves right from \$, nor may it print another symbol over \$.

The linear bounded automaton is simply a TM which, instead of having a potentially infinite tape, is restricted to the *finite* portion of the tape containing the input (plus the endmarker).

→ a model of computation closer to actual computers

REMARK: restricting TMs to an amount of tape that, on each input, is bounded by some linear function of the length of the input would result in an identical computational ability as restricting the Turing machine to the portion of the tape containing the input only

→ hence the name "linear bounded" automaton

The language accepted by a LBA is $\{w \in \Sigma^* \mid q_0 w\$$ yields an acceptance configuration$\}$.

From CS grammars to LBA

Thm.: for every CS grammar G, there exists an equivalent LBA

Proof

Let's construct a LBA M that recognizes $L(G)$.

We start the computation with configuration $q_0\, w\$$, with $w \in \Sigma^*$.

If $w = \varepsilon$, then reject.

Otherwise, if α is the current content of the tape, M repeatedly does the following:

1. Nondeterministically select a position $i \in \{1, \ldots, |\alpha|\}$
2. Nondeterministically select a production $\beta ::= \gamma$ of G
3. If γ appears beginning in position i of α, replace γ by β there, by using a proper "left shifting" if $|\beta| < |\gamma|$ (REMARK: right shifting will never be used, since in CSGs $|\beta| \leq |\gamma|$)
4. Compare the content of the tape with $S\sqcup\ldots\sqcup\$$ (where S is the starting symbol of G).
   - If they coincide, accept
   - If not, go back to Step (1)

We can prove that $w \in L(G)$ if and only if $w \in L(M)$.

Q.E.D.

From LBA to CS grammars

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: if $L = L(M)$, for some LBA M, then $L\setminus\{\varepsilon\}$ is context-sensitive.

Proof

The idea is similar to the proof for TM vs Type 0 grammars, but here we can NOT use ε-moves.

Now variables in G are pairs:

$$V = \{A_1, A_2\} \cup (\Sigma \times ((Q \cup \{\varepsilon\})(\Gamma \cup \Gamma\$))$$

Again, variables are needed to mimick configurations of M that arise during its computation: the configuration $c_1..c_h q c_{h+1}..c_m$ arising from input $a_1..a_n$ is now modeled by the sequence of variables

$$[a_1 , c_1]\ldots[a_h , c_h]\,[a_{h+1} , q\, c_{h+1}]\ldots[a_n , c_n\$]$$

where $c_1 \ldots c_n = c_1..c_m \sqcup \ldots \sqcup$.

Rules:

1. $A_1 ::= [a , q_0\, a]\, A_2 \mid [a , q_0\, a\$]$ for all $a \in \Sigma$
2. $A_2 ::= [a , a]\, A_2 \mid [a , a\$]$ for all $a \in \Sigma$
3. $[a, q\,x][b, \beta] ::= [a, y][b, q'\beta]$, for all $a,b \in \Sigma$, $q,q' \in Q$, $x,y \in \Gamma$, $\beta \in \Gamma \cup \Gamma\$$ s.t. $\delta(q, x) = (q', y, \mathrm{R})$
4. $[b, z][a, q\,x\beta] ::= [b, q'z][a, y\beta]$, for all $a,b \in \Sigma$, $q,q' \in Q$, $x,y,z \in \Gamma$, $\beta \in \{\$,\varepsilon\}$ s.t. $\delta(q, x) = (q', y, \mathrm{L})$
5. $[a\ ,\ q_{\text{accept}}\,\beta] ::= a$ for all $a \in \Sigma$, $\beta \in \Gamma \cup \Gamma\$$
6. $[a\ ,\alpha]\, b ::= a\, b$ and $b\,[a\ ,\alpha] ::= b\, a$ for all $a,b \in \Sigma$, $\alpha \in \Gamma \cup \Gamma\$$

From LBA to CS grammars

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Now, if M accepts $a_1 \ldots a_n$, then G produces it:

$A_1 \Rightarrow^* [a_1 , q_0 a_1] [a_2 , a_2] \ldots [a_n , a_n\$]$ (rules 1 and 2 – REMARK: 2 only if $n>1$)

$\Rightarrow^* [a_1 , x_1] \ldots [a_h , q_{\text{accept}} x_h] [a_{h+1} , x_{h+1}] \ldots [a_n , x_n\$]$ (rules 3 and 4, according to the accepting computation of M)

$\Rightarrow [a_1 , x_1] \ldots a_h [a_{h+1} , x_{h+1}] \ldots [a_n , x_n\$] \Rightarrow^* a_1 \ldots a_n$ (rules 5 and 6)

And conversely…

Q.E.D.