

Chapter 7

Other language classes

7.1 Context-sensitive languages

7.2 The Chomsky hierarchy

7.3 2DPDAs and Cook's theorem

7.1 Context-sensitive languages

CS

context-sensitive $G = (V, \Sigma, P, S)$ productions $\alpha B \gamma \rightarrow \alpha \beta \gamma$

$$B \in V, \alpha, \beta, \gamma \in (V \cup \Sigma)^*, \beta \neq \epsilon$$

 $(\alpha, B, \gamma) \rightarrow \beta$ $B \rightarrow \beta$ in context (α, γ) rewriting $\xi_1 \alpha B \gamma \xi_2 \Rightarrow \xi_1 \alpha \beta \gamma \xi_2$

$$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

$$S \rightarrow ABSc$$

$$S \rightarrow Abc$$

$$BA \rightarrow XA$$

$$XA \rightarrow XB$$

$$XB \rightarrow AB$$

$$Bb \rightarrow bb$$

$$A \rightarrow a$$

$$\{ a^n b^n c^n \mid n \geq 1 \} \text{ in CS - CF}$$

$$BA \Rightarrow XA \Rightarrow XB \Rightarrow AB$$

$$S \Rightarrow ABSc \Rightarrow ABABSc \Rightarrow^* (AB)^{n-1} Sc^{n-1} \Rightarrow$$

$$(AB)^{n-1} Abc^n = A(BA)^{n-1} bc^n \Rightarrow^*$$

$$A(AB)^{n-2} ABbc^n \Rightarrow$$

$$A(AB)^{n-2} Abbc^n = AA(BA)^{n-2} bbc^n \Rightarrow^*$$

$$AA(AB)^{n-2} bbc^n = AA(AB)^{n-3} ABbbc^n \Rightarrow$$

$$AA(AB)^{n-3} Abbbc^n = AAA(BA)^{n-3} bbbc^n \Rightarrow^*$$

$$A^n b^n c^n \Rightarrow^* a^n b^n c^n$$

MON

length-increasing (monotone)

$$G = (V, \Sigma, P, S)$$

$$\text{productions } \alpha \rightarrow \beta \in P \quad |\alpha| \leq |\beta|$$

$$\text{rewriting } \gamma_1 \alpha \gamma_2 \Rightarrow \gamma_1 \beta \gamma_2$$

$$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

Ex. $S \rightarrow aBSc$

$$S \rightarrow abc$$

$$Ba \rightarrow aB$$

$$Bb \rightarrow bb$$

$$L(G) = \{ a^n b^n c^n \mid n \geq 1 \}$$

MON=CS

Thm. length-increasing iff context-sensitive

$$CDE \rightarrow JKLMN$$

$$CDE \rightarrow X_1DE$$

$$X_1DE \rightarrow X_1X_2E$$

$$X_1X_2E \rightarrow X_1X_2X_3$$

$$X_1X_2X_3 \rightarrow JX_2X_3$$

$$JX_2X_3 \rightarrow JKX_3$$

$$JKX_3 \rightarrow JKLMN$$

special symbols X_i used here only

only one production for X_i

reorder steps

$$\alpha_1 CDE\beta_1 \Rightarrow \alpha_1 X_1DE\beta_1 \Rightarrow^* \alpha_2 X_1DE\beta_2 \Rightarrow \alpha_2 X_1X_2E\beta_2$$

$$\alpha_1 \Rightarrow^* \alpha_2 \quad DE\beta_1 \Rightarrow^* DE\beta_2$$

$$\alpha_1 CDE\beta_1 \Rightarrow^* \alpha_1 CDE\beta_2 \Rightarrow \alpha_1 X_1DE\beta_2 \Rightarrow \alpha_1 X_1X_2E\beta_2 \Rightarrow^* \alpha_2 X_1DE\beta_2$$

etcetera

LBA

linear-bounded automaton

one-tape nondeterministic Turing machine

with tape markers $\triangleright \triangleleft$ ($\#, b$)

\sim linear space TM

Thm. $CS \subseteq LBA$

introduce 'tracks'

technically $\Sigma \rightsquigarrow \Sigma \times \Sigma_1 \times \Sigma_2$

new alphabets Σ_i

simulate derivation steps nondeterministically

Thm. LBA \subseteq CS

basic simulation

$\triangleright a b X q a B a A \triangleleft$ in state q reading a

	LBA	MON
move right	$\delta(p, X) = (q, Y, R)$	$pX \rightarrow Yq$
stationary	$\delta(p, X) = (q, Y, S)$	$pX \rightarrow qY$
move left	$\delta(p, X) = (q, Y, L)$	$ZpX \rightarrow qZY$

state q and markers $\triangleright, \triangleleft$ have to disappear

need composite symbols

combine with neighbouring tape symbol

$[pX], [\triangleright X], [X\triangleleft], [p\triangleright X], [\triangleright pX], [pX\triangleleft], [Xp\triangleleft],$
 $[p\triangleright X\triangleleft], [\triangleright pX\triangleleft], [\triangleright Xp\triangleleft]$

$\triangleright a b X q a B a A \triangleleft \rightsquigarrow [\triangleright a] b X [qa] B a [A\triangleleft]$

add *many* cases for the rules

initialize tape $(a \in \Sigma)$

$$S \rightarrow [q_0 \triangleright a \triangleleft] \mid [q_0 \triangleright a \triangleleft] S_1$$

$$S_1 \rightarrow aS_1 \mid [a \triangleleft]$$

move right $\delta(p, X) = (q, Y, R) \quad (X \neq \triangleright)$

$$[pX] Z \rightarrow Y [qZ] \quad + \text{variants}$$

$$[\triangleright pX] Z \rightarrow [\triangleright Y] [qZ]$$

$$[pX] [Z \triangleleft] \rightarrow Y [qZ \triangleleft]$$

$$[\triangleright pX] [Z \triangleleft] \rightarrow [\triangleright Y] [qZ \triangleleft]$$

$$[pX \triangleleft] \rightarrow [Yq \triangleleft] \quad (\text{i.e., } Z = \triangleleft)$$

$$[\triangleright pX \triangleleft] \rightarrow [\triangleright Yq \triangleleft]$$

at left marker $\delta(p, \triangleright) = (q, \triangleright, R)$

$$[p \triangleright Z] \rightarrow [\triangleright qZ]$$

$$[p \triangleright Z \triangleleft] \rightarrow [\triangleright qZ \triangleleft]$$

halt and clean

$$[h \triangleright X] Y \rightarrow X [Y \$]$$

$$[X \$] Y \rightarrow X [Y \$]$$

$$[X \$] [Y \triangleleft] \rightarrow XY$$

$$[h \triangleright X \triangleleft] \rightarrow X$$

$$[h \triangleright X] [Y \triangleleft] \rightarrow XY$$

cases

.. pXZ ..

$\triangleright pXZ$..

.. $pXZ \triangleleft$

$\triangleright pXZ \triangleleft$

.. $pX \triangleleft$

$\triangleright pX \triangleleft$

* stationary $\delta(p, X) = (q, Y, S)$

$[pX] \rightarrow [qY] + \text{variants}$

$[\triangleright pX] \rightarrow [\triangleright qY] \quad [pX\triangleleft] \rightarrow [qY\triangleleft] \quad [\triangleright pX\triangleleft] \rightarrow [\triangleright qY\triangleleft]$

* move left $\delta(p, X) = (q, Y, L) \quad (X \neq \triangleleft)$

$Z[pX] \rightarrow [qZ]Y + \text{variants}$

$[\triangleright Z][pX] \rightarrow [\triangleright qZ]Y \quad Z[pX\triangleleft] \rightarrow [qZ][Y\triangleleft] \quad [\triangleright Z][pX\triangleleft] \rightarrow [\triangleright qZ][Y\triangleleft] \quad [\triangleright pX] \rightarrow [q\triangleright Z]Y \quad [\triangleright pX\triangleleft] \rightarrow [q\triangleright Z][Y\triangleleft]$

$\delta(p, \triangleleft) = (q, \triangleleft, L)$

$[Zp\triangleleft] \rightarrow [qZ\triangleleft] \quad [\triangleright Zp\triangleleft] \rightarrow [\triangleright qZ\triangleleft]$

Thm. Every CSL is recursive

$x \in L$ is decidable

technically: by an always halting TM

generate *all* strings up to length $|x|$

Thm. $CSL \subset REC$ [strict]

diagonalization

fix alphabet $\{0, 1\}$

(effectively) enumerate all CSG's G_i .

(effectively) enumerate all strings x_i .

define $L \subseteq \{0, 1\}^*$ $x_i \in L$ iff $x_i \notin L(G_i)$

$x_i \in L$ is decidable as $x_i \notin L(G_i)$ is decidable.

$L \neq L_i$ for all i

twenty boxes contain six bombs

opening a box with a bomb will detonate it

how can you 'prove' that box number one contains a bomb, that is, without opening it

use a nondeterministic approach

nondeterminism

Immerman and Szelepcsényi (1988)

Thm. CSL closed under complement

configuration:

state, tape contents, position head
 on $|x| = n$, at most $C = |Q||\Gamma|^n(n+2)$ moves
 accept with empty tape: $q_0 \triangleright x \triangleleft \vdash^* h \triangleright B^n \triangleleft$

Prf. 1. count reachable configurations

2. complementation

3. implement on LBA

2. input: x string,

R number of reachable configurations

accept x when not accepted by \mathcal{M} ,

when R different configurations γ , not $h \triangleright B^n \triangleleft$,

can be guessed together with computation

$q_0 \triangleright x \triangleleft \vdash^* \gamma$

inductive counting

1. R_i configurations reachable in **at most i** steps
(from $q_0 \triangleright x \triangleleft$)

$$R_0 = 1$$

$$R_i \rightsquigarrow R_{i+1}$$

$r = 0$ counter

for each configuration β

check all possible predecessors (as follows)

guess R_i different configurations γ together

with computation in $\leq i$ moves

if $\gamma = \beta$ or $\gamma \vdash \beta$ then $r++$

(when the guesses do not work, nothing is accepted)

3. add 'track' to tape
linear bounded (\sim linear space TM)

technically $\Sigma \rightsquigarrow \Sigma \times \Sigma_1 \times \Sigma_2$
new alphabets Σ_i

store number of configurations,

$$\text{at most } C = |Q| |\Gamma|^n (n + 2)$$

$n \lg |\Gamma| + \lg |Q| + \lg(n + 2)$ bits linear

generate different configurations

(in lexicographic order)

7.2 The Chomsky hierarchy

7.3 2DPDAs and Cook's theorem

2DPDA push-down automaton

deterministic, two-way, endmarkers $\triangleright \triangleleft (\#, b)$

$$\mathcal{M} = (Q, \Sigma, \Gamma, \triangleright, \triangleleft, \delta, q_0, Z_0, F)$$

Q finite set states

Σ input alphabet

Γ tape alphabet

$\triangleright, \triangleleft$ tape endmarkers (not in Σ)

$q_0 \in Q$ initial state

$Z_0 \in \Gamma$ initial stack symbol

$F \subseteq Q$ final states

δ (partial) transition function

$$\delta : Q \times (\Sigma \cup \{\triangleright, \triangleleft\}) \times \Gamma \rightarrow Q \times \{-1, 0, 1\} \times \Gamma^*$$

$$\delta(q, a, X) = (p, j, \alpha)$$

no ϵ -moves

Ex. $\{ 0^i 1^i 2^i \mid i \geq 1 \}$ in 2DPDA - CFL

check input belongs to $\{ 0^i 1^i \mid i \geq 1 \} 2^*$
as ordinary (deterministic) PDA

return to left tape marker

check input belongs to $0^* \{ 1^i 2^i \mid i \geq 1 \}$

open: is CFL \subseteq 2DPDA ?

Ex. $\{ ww \mid w \in \{0, 1\}^* \}$ in 2DPDA - CFL

find middle

 move left: push length on stack

 move right, pop two symbols every step

copy second half on stack

 move right, from middle to end

find middle

 again, see above, on stack above w

compare 1st half (tape) with 2nd half (stack)

 note direction

aba**abb**abab c abb


Ex. $\{ xcy \mid x \in \{a, b\}^*, y \text{ subword of } x \}$ in 2DPDA

pattern matching: pattern y , text x

find match: move y along x

here:

move x over y

▷ *a b b a b b a b a b c a b a* ◁

□

Z find *c*

...

...

□

Z

□

Z store *x*

□

bZ

...

...

□

abbabbababZ

...

... find *c*

□

abbabbababZ

□

abbabbababZ check

□

bbabbababZ

□

babbababZ mismatch

□

babbababZ restore *x*

□

bbabbababZ

□

abbabbababZ

□

bbabbababZ pop letter *x*

□

bbabbababZ check, mismatch

A 2DPDA can be simulated in linear time on a RAM.

	RLIN REG	DPDA	CF PDA _e	DLBA	MON LBA	REC	TYPE0 RE
intersection	+	-	-	+	+	+	+
complement	+	+	-	+	+	+	-
union	+	-	+	+	+	+	+
concatenation	+	-	+	+	+	+	+
star, plus	+	-	+	+	+	+	+
ϵ -free morphism	+	-	+	+	+	+	+
morphism	+	-	+	-	-	-	+
inverse morphism	+	+	+	+	+	+	+
intersect reg lang	+	+	+	+	+	+	+
mirror	+	-	+	+	+	+	+
	fAFL		fAFL	AFL	AFL	AFL	fAFL

$\cap^c \cup$ boolean operations

$\cup \cdot *$ regular operations

$h h^{-1} \cap R$ (full) trio operations

transparencies made for

Second Course in
Formal Languages and
Automata Theory

based on the book by Jeffrey Shallit
of the same title

Hendrik Jan Hoogeboom, Leiden
<http://www.liacs.nl/~hoogeboo/second/>