Undecidable Problems for CFGs

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2015

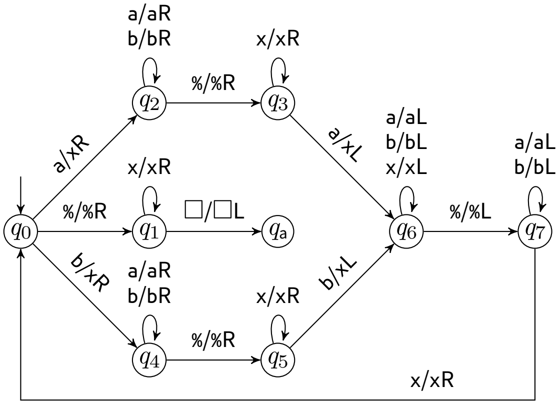
Decidable vs undecidable

Decidable	Undecidable
DFA D accepts w	TM M accepts w
CFG G generates w	TM M halts on w
DFAs D and D' accept same inputs	TM M accepts some input
	TM M and M' accept the same inputs

CFG G generates all inputs?
CFG G is ambiguous?

Representing computations

$$L_1 = \{w\%w \mid w \in \{\mathsf{a}, \mathsf{b}\}^*\}$$



q_0 abb%abb

q_2 abb%abb

⋮

q_2 abb%abb

q_3 abb%abb

q_6 abb%xbb

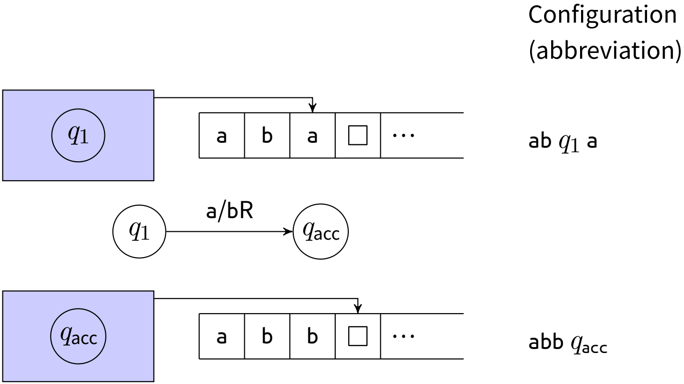
⋮

q_1 xxx%xxx□

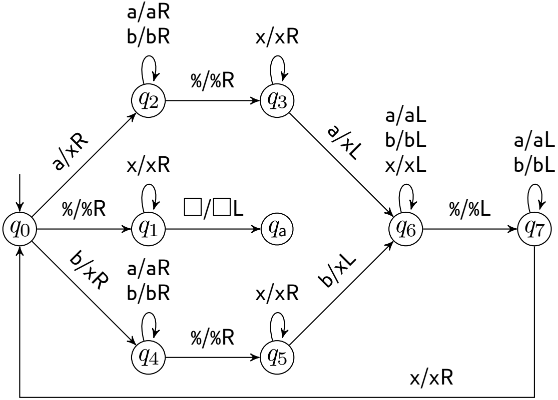
q_a xxx%xxx

Configurations

A configuration consists of current state, head position, and tape contents



Computation histories



q_0 abb%abb

a q_2 bb%abb

⋮

abb q_2 %abb

abb% q_3 abb

abb q_2 %xbb

⋮

xxx%xxx q_1

xxx%xx q_a x

computation history

Computation histories as strings

If M halts on w, the computation history of (M, w) is the sequence of configurations $C_1, \ldots, C_k$ that M goes through on input w

q_0 ab%ab
a q_2 b%ab
$\vdots$
xx%xx q_1
xx%x q_a x

$$\#\underbrace{q_0\,\text{ab\%ab}}_{C_1}\#\underbrace{\text{x}\,q_1\,\text{b\%ab}}_{C_2}\#\ldots\#\underbrace{\text{xx\%x}\,q_\text{a}\,\text{x}}_{C_k}\#$$

The computation history can be written as a string h over alphabet $\Gamma \cup Q \cup \{\#\}$

accepting history:	M accepts w	$\Leftrightarrow$	q_acc appears in h
rejecting history:	M rejects w	$\Leftrightarrow$	q_rej appears in h

Undecidable problems for CFGs

$$\mathsf{ALL}_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG that generates all strings}\}$$

The language $\mathsf{ALL}_{\mathsf{CFG}}$ is undecidable

We will argue that

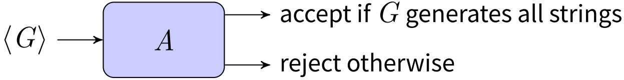
If $\mathsf{ALL}_{\mathsf{CFG}}$ can be decided, so can $\overline{A_{\mathsf{TM}}}$

$$\overline{A_{\mathsf{TM}}} = \{\langle M, w\rangle \mid M \text{ is a TM that rejects or loops on } w\}$$

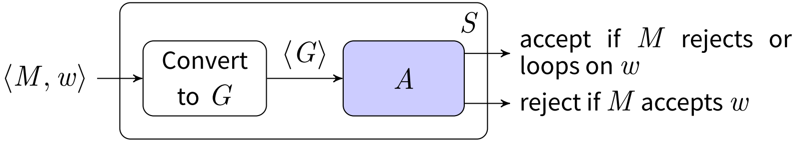
Undecidable problems for CFGs

Proof by contradiction

Suppose some Turing machine A decides ALL_{CFG}



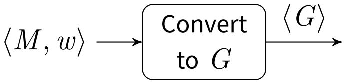
We want to construct a Turing machine S that decides $\overline{A_{\text{TM}}}$



G generates all strings if M rejects or loops on w

G fails to generate some string if M accepts w

Undecidable problems for CFGs



G fails to generate some string

$$\Updownarrow$$

M accepts w

The alphabet of G will be $\Gamma \cup Q \cup \{\#\}$

G will generate all strings except
accepting computation histories of (M, w)

First we construct a PDA P, then convert it to CFG G

Undecidablility via computation histories

candidate computation history h of (M, w) → P → accept everything except accepting h

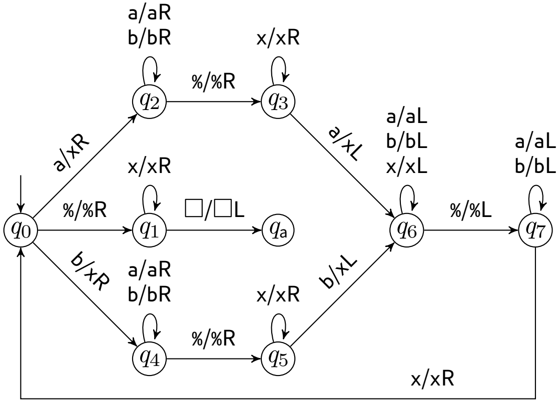
#$q_0$ab%ab#xq_1b%ab#...#xx%xq_ax# $\Rightarrow$ Reject

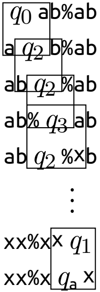
$P =$ on input h (try to spot a mistake in h)

- If h is not of the form #w_1#w_2#...#w_k#, accept
- If $w_1 \neq q_0 w$ or w_k does not contain q_a, accept
- If two consecutive blocks w_i#w_{i+1} do not follow from the transitions of M, accept

Otherwise, h must be an accepting history, reject

Computation is local

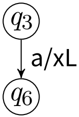




Changes between configurations always occur around the head

Legal and illegal transitions windows

legal windows	illegal windows
⋯ abx ⋯	⋯ $q_3$ab ⋯
⋯ abx ⋯	⋯ abq_3 ⋯
⋯ aq_3a ⋯	⋯ q_3q_3a ⋯
⋯ $q_6$ax ⋯	⋯ q_3q_3x ⋯
⋯ aba ⋯	⋯ aq_3a ⋯
⋯ abq_6 ⋯	⋯ $q_6$ab ⋯
⋯ aa$\square$ ⋯	⋯ aq_3a ⋯
⋯ xa$\square$ ⋯	⋯ aq_6x ⋯



Implementing P

If two consecutive blocks w_i#w_{i+1} do not follow from the transitions of M, accept

#xb%$q_3$ab
#xbq_5%xb

For every position of w_i:
- Remember offset from # in w_i on stack
- Remember first row of window in state

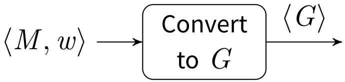
After reaching the next #:
- Pop offset from # from stack as you consume input
- Remember second row of window in state

If window is illegal, accept; Otherwise reject

The computation history method

$\text{ALL}_{\text{CFG}} = \{\langle G \rangle \mid G \text{ is a CFG that generates all strings}\}$

If ALL_{CFG} can be decided, so can $\overline{A_{\text{TM}}}$



G accepts all strings except accepting computation histories of (M, w)

We first construct a PDA P, then convert it to CFG G

Post Correspondence Problem

Input: A fixed set of tiles, each containing a pair of strings

bab	c	a	baa	a	bab
cc	ab	ab	a	baba	ε

Given an infinite supply of tiles from a particular set, can you match top and bottom?

a	baa	bab	c	c	bab	a
ab	a	ε	ab	ab	cc	baba

Top and bottom are both abaababccbaba

Undecidability of PCP

PCP $= \{\langle T \rangle \mid T$ is a collection of tiles that contains a top-bottom match$\}$

The language PCP is undecidable

Ambiguity of CFGs

$$\text{AMB} = \{\langle G \rangle \mid G \text{ is an ambiguous CFG}\}$$

The language AMB is undecidable

We will argue that

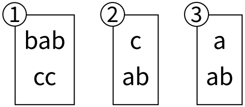
If AMB can be decided, then so can PCP

Ambiguity of CFGs

T (collection of tiles) $\longmapsto$ G (CFG)

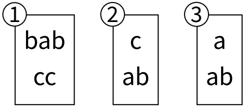
If T can be matched, then G is ambiguous
If T cannot be matched, then G is unambiguous

First, let's number the tiles



Ambiguity of CFGs

T (collection of tiles) $\longmapsto$ G (CFG)



Terminals: a, b, c, 1, 2, 3

Variables: S, T, B

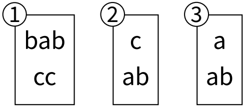
Productions:

$$S \to T \mid B$$

$T \to \text{bab}\,T1$	$T \to \text{c}\,T2$	$T \to \text{a}\,T3$
$B \to \text{cc}\,B1$	$B \to \text{ab}\,B2$	$B \to \text{ab}\,B3$
$T \to \text{bab1}$	$T \to \text{c2}$	$T \to \text{a3}$
$B \to \text{cc1}$	$B \to \text{ab2}$	$B \to \text{ab3}$

Ambiguity of CFGs

T (collection of tiles) $\longmapsto$ G (CFG)



Terminals: a, b, c, 1, 2, 3

Variables: S, T, B

Productions:

$$S \to T \mid B$$

$$T \to \text{bab}\,T\text{1} \qquad T \to \text{c}\,T\text{2} \qquad T \to \text{a}\,T\text{3}$$

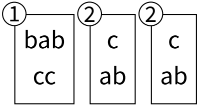
$$B \to \text{cc}\,B\text{1} \qquad B \to \text{ab}\,B\text{2} \qquad B \to \text{ab}\,B\text{3}$$

$$T \to \text{bab1} \qquad T \to \text{c2} \qquad T \to \text{a3}$$

$$B \to \text{cc1} \qquad B \to \text{ab2} \qquad B \to \text{ab3}$$

Ambiguity of CFGs

Each sequence of tiles gives a pair of derivations



$$S \Rightarrow T \Rightarrow \text{bab}\,T1 \Rightarrow \text{babc}\,T21 \Rightarrow \text{babcc221}$$
$$S \Rightarrow B \Rightarrow \text{cc}\,B1 \Rightarrow \text{ccab}\,B21 \Rightarrow \text{ccabab221}$$

If the tiles match, these two derive the same string
(with different parse trees)

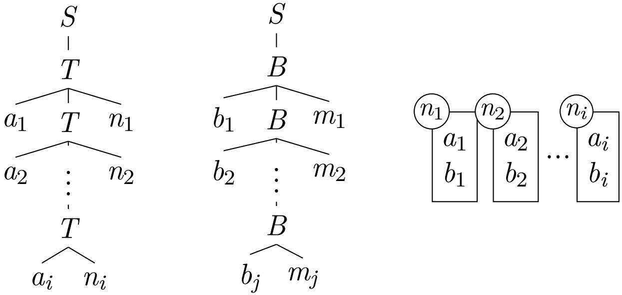
Ambiguity of CFGs

T (collection of tiles) $\longmapsto$ G (CFG)

If T can be matched, then G is ambiguous ✓

If T cannot be matched, then G is unambiguous ✓

If G is ambiguous, then the two parse trees will look like



Therefore $n_1 n_2 \dots n_i = m_1 m_2 \dots m_j$, and there is a match