COMS3261: Computer Science Theory

Fall 2013

Mihalis Yannakakis

Lecture 23, 11/27/13

Post Correspondence Problem

- Many undecidable problems don't have to do with TMs and programs
- PCP Input: Two finite lists $A=(w_1,\ldots,w_k)$, $B=(x_1,\ldots,x_k)$ with the same number of strings over same alphabet Σ.
- Question: $\exists$? finite sequence of indices $i_1,\ldots i_m$ (repetitions allowed) such that $w_{i1} \ldots w_{im} = x_{i1} \ldots x_{im}$?

PCP Example

	A	B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Solution: 2 1 1 3

A : 1 0 1 1 1 | 1 | 1 | 1 0

B : 1 0 | 1 1 1 | 1 1 1 | 0

2 1 1 3

If only strings 2,3 then no solution (strings of A longer than B)

Modified PCP (MPCP)

- Input: Same as PCP: Two finite lists A=(w_1,…,w_k), B=(x_1,…,x_k) with the same number of strings over same alphabet Σ.
- Question: $\exists$ solution that starts with strings 1?

PCP, MPCP are both undecidable

Will show:

(1) $L_u \leq_m$ MPCP

(2) MPCP $\leq_m$ PCP

Proof of MPCP $\leq_m$ PCP

- Trick: new symbols *, $
- A list: Put * after every symbol: $w_i \rightarrow w'_i$
- B list: Put * before every symbol: $x_i \rightarrow x'_i$
- Add pair 0: $(*w'_1, x'_1)$ and pair k+1: ($, *$)
- Example:

	A	B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

	A'	B'
i	w'_i	x'_i
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	$	*$

Solution must start with 0 pair ⇒ A string has extra * at the end, and this continues ⇒ must finish with (k+1) pair

Proof of $L_u \leq_m$ MPCP

- Given (M,w) compute MPCP instance (lists A,B) such that M accepts w iff MPCP has a solution.
- Assume wlog that M has semi-infinite tape, does not write B (blank – can use substitute B'), and represent ID as before but without the final B if head is at right end reading B, i.e. ID= string over Γ + a unique state
- Represent computation of M on w as string $\#ID_0\#ID_1\#ID_2\ldots$ where $\# \notin \Gamma$ a separator
- Will construct lists A, B so that solution (if $\exists$) is the computation of M on w, where B string is one ID ahead of A, and when accepting state is entered then A can catch up with B to finish up with equal string.
- In general, the two strings in a prefix of a solution can get arbitrarily far from each other before they catch up.

MPCP instance

	A list	B list	
1st pair	#	$\#q_0w\#$	
copy	X	X	$\forall X \in \Gamma$
	#	#	
transitions	qX	Yp	if $\delta(q,X)=(p,Y,R)$
	ZqX	pZY	if $\delta(q,X)=(p,Y,L)$, $\forall Z \in \Gamma$
	q#	Yp#	if $\delta(q,B)=(p,Y,R)$
	Zq#	pZY#	if $\delta(q,B)=(p,Y,L)$, $\forall Z \in \Gamma$
cleaning up	XqY	q	$\forall q \in F$, $\forall X,Y \in \Gamma$
	Xq	q	
	qY	q	
finish	q##	#	$\forall q \in F$

Example

B: # q_0 0 0 1 0 # 1 p 0 1 0 # r 1 1 1 0 # r 1 1 0 …

A: # q_0 0 0 1 0 # 1 p 0 1 0 # r 1 1 1 0 # r 1 1 0…

clean- up

where r $\in$ F

B: … # r 0 # r # #

A: … # r 0 # r # #

finish

- Forced to form the computation of M on w to match.
- If no accepting state, then B string will always be ahead (longer)

PCP to CFL languages

- List A: strings w_1, w_2, …, w_k over Σ; assume 1,2,…,k $\notin \Sigma$
- Language L_A = { w_{i1} … w_{im} i_m …i_1 | i_1,…,i_m $\in$ {1,…,k} } over alphabet T = $\Sigma \cup$ {1,…,k}
- Both L_A, and its complement L_A^c are CFL, in fact DCFL
- CFG G_A for L_A: A $\rightarrow$ w_1A1 | w_2A2 | …| w_kAk | $w_1$1 | …| w_kk
- DPDA for L_A: Read letters from Σ and push on stack. Then for each index i $\in$ {1,…,k} in input, pop stack and verify that it contains the reverse of w_i
- Stop and reject if there is a problem (not a match)
- Accept at end if stack is emptied and input finished.
- DPDA for L_A^c : Similar but always in accepting state except if input in L_A

Undecidable problems for CFL languages

- Emptiness of $\cap$: Is $L(G1) \cap L(G2) = \varnothing$?
- Reduction from PCP: Given instance= lists A,B, construct the CFG G_A, G_B for them with variables A, B

  $L(G_A) \cap L(G_A) \neq \varnothing$ iff PCP instance has a solution

- Ambiguity of a grammar
- Reduction from PCP: Given instance= lists A,B, construct the CFG G_A, G_B for them with variables A, B

  define grammar G: $S \rightarrow A \mid B$ plus productions of G_A, G_B

  G is ambiguous iff PCP instance has a solution

Undecidable problems for CFL languages

- $L(G) = T^*$? where G is grammar with terminal alphabet T

- Proof: Take grammars G'_A, G'_B for $L_A{}^c$, $L_B{}^c$, with start symbols A', B'

  Let G= $S \rightarrow A' \mid B'$, plus productions of G'_A, G'_B

  Then $L(G) = L_A{}^c \cup L_B{}^c = T^* - (L_A \cap L_B) \Rightarrow$

  $L(G) = T^* \Leftrightarrow L_A \cap L_B \neq \varnothing \Leftrightarrow$ PCP has no solution

Undecidable problems for CFL languages

- Corollaries: The following are undecidable
- $L(G_1) = L(G_2)$? for given CFGs G_1, G_2
- $L(G) = L(R)$? for given CFG G and regular expression R
- $L(G_1) \subseteq L(G_2)$? for given CFGs G_1, G_2
- $L(R) \subseteq L(G)$? for given reg expr R, CFG G

- But $L(G) \subseteq L(R)$? for given CFG G, regular expression R is decidable ($\Leftrightarrow L(G) \cap R^c = \emptyset$ and $L(G) \cap R^c$ is CFL)