Informatics 2 – Introduction to Algorithms and Data Structures

Tutorial 8: Parsing algorithms and polytime reductions

1. Consider the following context free grammar with start symbol S:

S	$\rightarrow$	NP VP	PP	$\rightarrow$	Pre NP
S	$\rightarrow$	*I* VP PP	V	$\rightarrow$	*ate*
NP	$\rightarrow$	Det N	Det	$\rightarrow$	*the* | *a*
VP	$\rightarrow$	*ate* NP	N	$\rightarrow$	*fork* | *salad*
VP	$\rightarrow$	V	Pre	$\rightarrow$	*with*

   (a) Convert this grammar to Chomsky Normal Form (see Lecture 22).

   (b) Use the CYK algorithm from Lecture 22 to parse the sentence

   I ate the salad with a fork

   (c) How many complete analyses of the sentence do you get? Draw their syntax trees.

   (d) Now add a further production rule to your CNF grammar to allow for the alternative prepositional phrase attachment, i.e. 'the salad with a fork'. Revise your CYK chart or graph to include any new entries this introduces.

   How many complete analyses are there now for the above sentence?

2. Consider the following grammar for arithmetic expressions such as $(\,\mathsf{n} * \mathsf{n} * \mathsf{n}\,)$. Here n stands for the lexical category of *numeric literals* such as 5 and -23.

Terminals:	(,), $*$, n
Nonterminals:	Exp, Ops
Productions:	$\mathsf{Exp} \rightarrow \mathsf{n}\ \mathsf{Ops} \mid (\,\mathsf{Exp}\,)$
	$\mathsf{Ops} \rightarrow \epsilon \mid * \mathsf{n}\ \mathsf{Ops}$
Start symbol:	Exp

   This grammar is somewhat restrictive — for example, it does not admit the string $(\,\mathsf{n} * \mathsf{n}\,) * \mathsf{n}$ — but it will do as an example.

   This is in fact an LL(1) grammar with the following parse table:

	(	)	$*$	n	$
Exp	(Exp)			n Ops	
Ops		ϵ	$*$ n Ops		ϵ

   Here, for example, the top left entry $(\,\mathsf{Exp}\,)$ stands for the production $\mathsf{Exp} \rightarrow (\,\mathsf{Exp}\,)$.

(a) Using this table, apply the LL(1) parsing algorithm from Lecture 23 to the input

$$(\ \mathsf{n} * \mathsf{n}\ )$$

At each step, show the operation applied, the input string remaining, and the stack state, as in the lecture.

(b) For each of the following three input strings, explain how and where an error arises in the course of the LL(1) parsing algorithm. In each case, suggest an error message that an LL(1) parser could issue to the author of the input string.

$$(\ )\qquad \mathsf{n}\ )\qquad \mathsf{n} *$$

3. Consider the MAXIMUM INDEPENDENT SET problem: We are given an undirected graph $G = (V, E)$ and we are asked to find an independent set I of V of maximum size. An independent set $I \subseteq V$ of the graph is a set of nodes such that for every two nodes $v, u \in I$, we have $(v, u) \notin E$.

(a) Define the appropriate *decision* version of the MAXIMUM INDEPENDENT SET set problem, called INDEPENDENT SET.

(b) Show that INDEPENDENT SET is NP-complete. For the NP-hardness, construct a polynomial-time reduction from 3SAT.

Hint: Construct "clause gadgets", i.e., triangles of nodes corresponding to the three literals of a clause, similarly to the reduction from 3SAT to VERTEX COVER presented in class to show the latter problem is NP-hard.

(c) Assume that you have an *oracle* (i.e., an algorithm that runs in time $O(1)$ every-time it is called) to solve the MAXIMUM INDEPENDENT SET problem. Explain how to use this oracle to solve the INDEPENDENT SET problem in polynomial time.

(d) Assume that you have an oracle to solve the INDEPENDENT SET problem. Explain how to use this oracle to solve the MAXIMUM INDEPENDENT SET problem in polynomial time.

4. (*Optional) A k-colouring of a graph G is a function $f : V \to \{1, 2, \ldots, k\}$ mapping nodes to *colours*, such that for any nodes u and v such that $(u, v) \in E$, it holds that $f(u) \neq f(v)$.

Consider the 3-COLOURING problem: Given a graph G as input, decide whether there is a 3-colouring of G. Prove that 3-COLOURING is NP-complete. For the NP-hardness, construct a polynomial-time reduction from 3SAT.

John Longley and Aris Filos-Ratsikas

February 2024