Complexité avancée - Homework 5

Benjamin Bordais

October 21, 2020
Due at 11.59 p.m., November 2, 2020

Some P-complete Problems

1. Show that the following problems are P-hard.

   - Recall that for a finite set X, a subset $S \subseteq X$, and a binary operator $* : X \times X \to X$ defined on X, we inductively define $S_{0,*} = S$ and $S_{i+1,*} = S_{i,*} \cup \{x * y \mid x, y \in S_{i,*}\}$. Then, the closure of S with regard to $*$ is the set $S_* = \cup_{i \in \mathbb{N}} S_{i,*}$.

     BinOpGen:

     – INPUT: A finite set X, a binary operator $* : X \times X \to X$ defined on X, a subset $S \subset X$ and $x \in X$;

     – OUTPUT: $x \in S_*$?

     Hint: reduce from MonotoneCircuitValue *with all nodes having at most two predecessors.*

   - Recall that a context-free grammar is a grammar $G = (V, T, I, \mathcal{P})$ where V is the set of non-terminal symbols, T is the alphabet of terminal symbols, $I \subseteq V$ is the axiom (i.e. set of initial variables) and $\mathcal{P} \subseteq V \times (V \cup T)^*$ is the finite set of production rules (the "context-free" part can be seen in the fact that the left-hand member of a rule in $\mathcal{P}$ has length one). The language $\mathcal{L}(G)$ of G is the set of words $w \in T^*$ that can be derived from I by applying the production rules.

     CFG-Derivability:

     – INPUT: G a context-free grammar on an alphabet T, and $w \in T^*$ a word;

     – OUTPUT: $w \in \mathcal{L}(G)$?

     Hint: reduce from the previous problem.

2. In fact these problems are P-complete. Show that BinOpGen is in P. Do you know a polynomial-time algorithm for CFG-Derivability ?

Solution:

1. - This question was in fact given in the exam of the year 2019-2020, and solutions for Questions 8 and 9 can be found on the web page of the course.
   - Consider an instance $(X, S, x, *)$ of BinOpGen. We define a grammar $G = (V, T, I, \mathcal{P})$ and a word $w \in T^*$ in the following way: the set of variables is

$V = X$, there is only one terminal symbol $T = \{a\}$, the initial variable is $I = \{x\}$, the set of production rules is : $\mathcal{P} = \mathcal{P}_* \cup \mathcal{P}_S$ with:

$$\mathcal{P}_* = \{x \to yz \mid x, y, z \in V,\ y * z = x\} \text{ and } \mathcal{P}_S = \{x \to \epsilon \mid x \in S\}$$

and $w = \epsilon$ is the empty string.
Then, we can prove that:

$$x \in S_* \Leftrightarrow \epsilon \ can\ be\ derived\ from\ x\ in\ G$$

First note that since this grammar is context-free and by definition of the production rules, any word that can be derived from x may be derived by first applying only rules in $\mathcal{P}_*$ and then rules in $\mathcal{P}_S$.
$\Leftarrow$: First, we can prove by induction on the number of rules used that if a word $v = v_1 \dots v_k$ is derived from x by only using production rules from $\mathcal{P}_*$, then $x \in \{v_i \mid 1 \leq i \leq k\}_*$. Second, if ϵ can be derived from x, then by considering the word $v = v_1 \dots v_k \in T^*$ that can be derived from x by only applying rules in $\mathcal{P}_*$ such that ϵ can be derived from v by only applying rules in $\mathcal{P}_S$, we obtain that $\{v_i \mid 1 \leq i \leq k\} \subseteq S$ and $x \in \{v_i \mid 1 \leq i \leq k\}_*$. That is, $x \in S_*$.
$\Rightarrow$: First we prove that if a word $v = v_1 \dots v_k \in T^*$ is such that there exists $j \geq 1$ such that for all $1 \leq i \leq k$ we have $v_i \in S_{j,*}$, then there exists a word $v' = v'_1 \dots v'_{k'} \in T^*$ for some $k' \geq k$ that can be derived from v by only applying rules in $\mathcal{P}_*$ such that for all $1 \leq i \leq k'$ we have $v'_i \in S_{j-1,*}$. Second, if we assume that $x \in S_*$, then there exists $j \geq 1$ such that $x \in S_{j,*}$. It follows that there exists a word $u = u_1 \dots u_n \in T^*$ that can be derived from x such that for all $1 \leq i \leq n$, we have $u_i \in S_{0,*} = S$. Then, ϵ can be derived from u by applying a rule in $\mathcal{P}_S$ for each of its letter. Hence, ϵ can be derived from x.
Overall, we have:

$$(X, S, x, *) \in \mathsf{BinOpGen} \Leftrightarrow (V, T, I, \mathcal{P}) \in \mathsf{CFG\text{-}Derivability}$$

As the reduction we described can be done in logspace, it follows that CFG-Derivability is P-hard.

2. CFG-Derivability can be solved in polynomial time using dynamic programming techniques, see for example the CYK algorithm on Wikipedia.