Uitwerking van opgaven in Sipser

Opgave 4.10

Laat zien dat

$$\text{INFINITE}_{PDA} = \{\langle M\rangle \mid M \text{ is een PDA en } L(M) \text{ is oneindig }\}$$

beslisbaar is.
De volgende TM beslist INFINITE_{PDA}:
Op input $\langle M\rangle$ waarbij M een PDA is,

1. Converteer M naar een equivalente CFG, en noem die G (dit is duidelijk een berekenbare functie/transformatie).
2. Laat n het maximum aantal symbolen aan de rechter kant van een regel in G zijn en stel dat b het aantal variabelen in G is, en laat $m = b^n + 1$.
3. Zij L de reguliere taal bestaande uit alle woorden van lengte $\geq m$.
4. Construeer een CFG H voor de taal $L \cap L(M)$ (dit kan omdat de doorsnede van een reguliere en een context vrije taal weer context vrij is, opgave 2.18).
5. Test of H in E_{CFG} zit (dit een beslisbare taal). Zo ja, reject, zo nee, accept.

We bewijzen dat dit algoritme INFINITE_{PDA} beslist: als $\langle M\rangle \in \text{INFINITE}_{PDA}$, dan is er een woord s van lengte $\geq m$ in $L(M)$. Omdat m de pomplengte is kunnen we s pompen volgens het pomplemma voor CFG's. Dus bevat $L(M)$ oneindig veel woorden. Omgekeerd, als $L(M)$ oneindig veel woorden bevat, moet het oneindig veel woorden van lengte $\geq m$ bevatten (het bevat natuurlijk oneindig veel woorden van lengte $\geq k$ voor elke k). Dus is $L(H)$ niet leeg, en dus is H niet in E_{CFG}, en dus accepteert het algoritme $\langle M\rangle$.

Opgave 4.12

Laat zien dat

$$\text{INC}_{REG} = \{\langle R, S\rangle \mid\ R \text{ en } S \text{ zijn reguliere expressies en } L(R) \subseteq L(S)\ \}$$

beslisbaar is.
De volgende TM beslist INC_{REG}:
Op input $\langle R, S\rangle$ waarbij R en S reguliere expressies zijn:

1. Converteer R naar de DFA M.
2. Construeer een DFA N voor het complement van $L(S)$ (dit kan omdat het complement van een reguliere taal weer regulier is).

3. Construeer een DFA K voor $L(M) \cap L(N)$.

4. Test of K in E_{DFA} zit (dit een beslisbare taal). Zo ja, accept, zo nee, reject.

Het is duidelijk dat dit algoritme INC_{REG} beslist.

Exercise 5.1

Show that EQ_{CFG} is undecidable. Observe that indeed we cannot copy the proof that EQ_{DFA} is decidable as the the CFL's are not closed under complement, while the regular languages are. We show that EQ_{CFG} is undecidable by showing that if it would be decidable, then so would ALL_{CFG}, which is not true (Theorem 5.13). And thus we have derived a contradiction.

So suppose EQ_{CFG} is decidable and let M be the decider. First we observe that there is a CFG which language is Σ^*, For example (in the case that $\Sigma = \{0, 1\}$), the CFG consisting of the rules $S \to \epsilon \mid 0S \mid 1S$ does the trick. Let us call this CFG H, thus $L(H) = \Sigma^*$. The following N is a decider for ALL_{CFG}:
On input $\langle G \rangle$, where G is a CFG:
Run M on $\langle G, H \rangle$, if it accepts, accept, otherwise, reject.

If is not difficult to see that N is a decider, and indeeds decides ALL_{CFG}. As this language is undecidable we have reached a contradiction, and whence must conclude that EQ_{CFG} is undecidable.

Exercise 7.19

If PATH would be NP-complete, this would, by definition, imply that for all $L \in$NP, $L \leq_P$PATH. But this again implies that for all L in NP, L is in P. Thus P = NP would follow, which we (except maybe some cranks) believe is not true.

We show that proving that PATH is not NP-complete implies that NP$\neq$P. We show this by contraposition: we assume P=NP and then show that PATH is NP-complete: so assume P=NP. Using exercise 7.17 it then follows that PATH is NP-complete. Thus if PATH is not NP-complete, it should be the case that NP$\neq$P, which is what we wanted to prove.

Exercise 7.20 (b)

The proof that LPATH is in NP we leave to the reader. We show that

$$\text{UHAMPATH} \leq_P \text{LPATH}.$$

From the fact that LPATH is in NP, and the definition of $\leq_P$ it then follows that LPATH is NP-complete, as UHAMPATH is NP-complete.

Our reduction takes the form

$$f(\langle G, a, b \rangle) = \langle G, a, b, |G| - 1 \rangle.$$

It is clear that f is a polynomial time function. To see that it is a reduction it remains to show that

$$\langle G, a, b\rangle \in \text{UHAMPATH} \Leftrightarrow \langle G, a, b, |G|-1\rangle \in \text{LPATH}.$$

$\Rightarrow$: If G has a Hamilton path from a to b, that path visits every node exactly once. Since G has $|G|$ many nodes, this path has length $|G|-1$. Hence $\langle G, a, b, |G|-1\rangle$ is in LPATH.
$\Leftarrow$: If G has a simple path from a to b from length at least $|G|-1$, the simplicity of the path (visiting a node at most once) implies that that path has to visit every node, and thus it must be a Hamilton path.

Exercise 7.21

Show that

$$\text{DOUBLESAT} = \{\langle\varphi\rangle \mid \varphi \text{ has at least two satisfying assigments}\}$$

is NP-complete.
The proof that DOUBLESAT is in NP we leave to the reader.
We prove the theorem by giving a polynomial time reduction from 3SAT to DOUBLESAT. Given a formula φ, define $f(\langle\varphi\rangle) = \varphi \vee x$, where x is a variable not occurring in φ. it is not difficult to see that f is a polynomial time reduction from 3SAT to DOUBLESAT.
In the same way one can prove that

$$\text{nSAT} = \{\langle\varphi\rangle \mid \varphi \text{ has at least } n \text{ satisfying assigments}\}$$

is NP-complete.

Exercise 7.23

Let

$$\text{CNF}_k = \{\langle\varphi\rangle \mid \varphi \text{ is a satisfiable cnf-formula where each variable appears in at most } k \text{ places}\}.$$

(a) Show that $\text{CNF}_2 \in P$.
(b) Show that CNF_3 is NP-complete.
(a) We give an informal description of a polynomial time decider M for CNF_2. On input φ M does the following:

1. Consider the first clause of φ. If it is of the form x, and there is a clause $\neg x$ in φ, reject,
2. otherwise the first clause is of the form $x \vee A$. If x does not appear negated in the other clauses, remove every clause of the form $x \vee B$ of φ, and call the result φ, if there remain no clauses in φ, accept,

3. if there are two clauses $x \vee A$ and $\neg x \vee B$ in φ, remove them from φ and add $A \vee B$ to φ and call the result φ, and go to 1.

It is clear that M is a decider. We leave it to the reader to show that its running time is polynomial and that it decides CNF_2.

(b) We leave it to the reader to show that CNF_3 is in NP. We show that $3SAT \leq_P \text{CNF}_3$.

First an example: $\varphi = (p \vee q) \wedge (p \vee r) \wedge (\neg p \vee s) \wedge (p \vee t)$. This formula is not in 3nmf, but the general case will be treated below. This formula is replaced by

$$\psi = (p \vee q) \wedge (\neg p \vee p_1) \wedge (p_1 \vee r) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_2 \vee s) \wedge (\neg p_2 \vee p_3) \wedge (p_3 \vee t) \wedge (\neg p_3 \vee p).$$

Note that the $(\neg p \vee p_1), (\neg p_1 \vee p_2), (\neg p_2 \vee p_3), (\neg p_3 \vee p)$ are expressing that $p \to p_1, p_1 \to p_2, p_2 \to p_3, p_3 \to p$, that is, the p, p_1, p_2, p_3 are all equivalent. Hence in this formula $(p_1 \vee r)$ expresses the same as $(p \vee r)$, etc. And thus φ is satisfiable when ψ is, but in ψ every variable occurs at most 3 times.

The general case. We define a reduction f from $3SAT$ to CNF_3 as follows. $f(\langle\varphi\rangle)$ is the formula that is the result of the following procedure:

1. pick the first propositional variable (reading from left to right) in φ that occurs more than 3 times in the formula, suppose it is p, and suppose it occurs at n places: $(x_1 \vee A_1), \dots, (x_n \vee A_n)$, where the x_i are p or $\neg p$. If no variable occurs more than 3 times, output φ.

2. Choose fresh variables $p_1, \dots, p_n$, remove the conjuncts $(x_i \vee A_i)$ from the formula and add as a conjunct the formula

$$(p_1 \vee A_1) \wedge (\neg p_1 \vee p_2) \wedge (p_2 \vee A_2) \wedge (\neg p_2 \vee p_3) \dots (p_n \vee A_n) \wedge (\neg p_n \vee p_1).$$

3. Call the new formula φ and go to 1.

Clearly, $f(\langle\varphi\rangle)$ is a formula in which every variable occurs at most three times. Also clear:

$$\langle\varphi\rangle \text{ satisfiable iff } f(\langle\varphi\rangle) \text{ satisfiable,}$$

and f is a polynomial function. Done!

Exercise 7.14

Show that P is closed under *.

Let $L \in P$. We show that $L^* \in P$. Let Σ be the alfabet of L. We use dynamic programming (see page 267). The idea of the lagorithm is that on input $w = w_1 \dots w_n$ we make a $n \times n$ grid (step 2.) and put a 1 in cell (i, j) if $w_i \dots w_j \in L^*$ and a 0 otherwise.

Now we fill the grid step by step. First we fill in the diagonals (i, i): a 1 if $w_i \in L$ (and thus in L^*) and a 0 otherwise (step 3.). Then we fill in the cells $(i, i + 1)$, then the cells $(i, i + 2)$, etc.

Here follows the algortihm. On input $w = w_1 \dots w_n$

1. If $w = \epsilon$, accept.
2. Make a $n \times n$ grid.
3. For every w_i test of $w_i \in L$, if so, put a 1 in (i, i) and a 0 otherwise.
4. For $l = 2$ to n,
    (a) For $i = 1$ to $n - l + 1$, let $j = i + l - 1$,
        (b) For $k = i$ to $j - 1$,
            (c) if $(i, i + k)$ and $(i + k + 1, j)$ contain a 1, put a 1 in (i, j).
5. If $(1, n)$ contains a 1, accept, anders reject.

Clearly, this algorithm decides L^*. To see that it runs in polynomial time, first observe that step 3. runs in polynomial time because L is in P, say in time $O(n^k)$. Step 4, (a), and (b) are repeated $O(n)$ times. Step 2. takes $O(n^2)$ time. Thus the whole algortithm takes $O(n^k + n^3 + n^2)$ time. Thus $O(n^k)$ time if $k > 3$ and $O(n^3)$ if $k \leq 3$.

A very simple reduction

We show that

$TUTQBF = \langle\varphi\rangle \mid\ \varphi$ is a true QBF that contains at least two universal quantfiers}

is PSPACE-complete.
The proof that TUTQBF is in PSPACE we leave to the reader. To show that it is PSPACE-complete it then suffices to show that TQBF$\leq_P$TUTQBF. The following reduction shows this:

$$f(\langle\varphi\rangle) = \langle\varphi \wedge \forall p(p \vee \neg p) \wedge \forall q(q \vee \neg q)\rangle.$$

(There are many possible reductions: $\forall a \forall b \varphi$, for fresh a and b is ok too.) It is not difficult to see that f is a polynomial time function. Thus it remains to show that

$$\langle\varphi\rangle \in \text{TQBF} \Leftrightarrow \langle\varphi \wedge \forall p(p \vee \neg p) \wedge \forall q(q \vee \neg q)\rangle \in \text{TUTQBF}.$$

$\Rightarrow$: If $\langle\varphi\rangle$ is in TQBF, the formula is true. But then so is $\langle\varphi \wedge \forall p(p \vee \neg p) \wedge \forall q(q \vee \neg q)\rangle$. Since this formula contains at least to universal quantifiers it follows that it is in TUTQBF.
$\Leftarrow$: $\langle\varphi \wedge \forall p(p \vee \neg p) \wedge \forall q(q \vee \neg q)\rangle$ is in TUTQBF, it follows that it is true. Then φ is true as well. Thus $\langle\varphi\rangle$ is in TQBF.

Twee CLIQUE opgaven

Define

$$\text{CLIQUE}_k = \{\langle G\rangle \mid G \text{ has a k-clique}\}.$$

1. Show for every k that CLIQUE_k is in P.
The following polynomial time algorithm M decides CLIQUE_k.
M: On input $\langle G\rangle$:

1. For every subset X of G of size k,
   (a) check wether it is a clique, if so, accept.
2. If not accepted, reject.

Clearly, step (a), and 2. are in $O(n)$ time. Step (a) is at most as many times repeated as there are subsets of G of size k. There are $O(n^k)$ such sets. Since k is *no part of the input* this is polynomial in the input, namely in the size of G. Thus M is a polynomial time TM, which proves that $\text{CLIQUE}_k \in P$.
The intuition behind this is that for this problem k is no part of the input.Thus we have proved something for infinitely many sets, for CLIQUE_1, CLIQUE_2, $\text{CLIQUE}_3, \ldots$. In CLIQUE_k, thus for a fixed k, if G becomes very big, n^k will be more or less n. In the CLIQUE problem, below, k is part of the input, which changes the situation accordingly:
2. Show that CLIQUE is in NP.
The following polynomial time non-deterministic TM M decides CLIQUE.
M: On input $\langle G\rangle$:

1. Guess a subset X of G is size k,
2. check wether it is a clique, if so, accept, otherwise reject.

This clearly is a polynomial time non-deterministic TM, as every branch runs in $O(n)$ time.