We recall the space-hierarchy theorem.

**Theorem 1** (Space-hierarchy theorem). For two space-constructible functions $f$ and $g$ such that $f = o(g)$, we have $DSPACE(f) \subsetneq DSPACE(g)$.

**Exercise 1** (Poly-logarithmic space). 1. Let $polyL = \bigcup_{k \in \mathbb{N}} SPACE(\log^k)$. Show that $polyL$ does not have a complete problem for logarithmic space reduction.\(^1\)

2. Recall that $PSPACE = \bigcup_{k \in \mathbb{N}} SPACE(n^k)$. Does $PSPACE$ have a complete problem for logarithmic space reduction? Why doesn’t the proof of the previous question apply to $PSPACE$?

**Solution 1.** 1. Assume towards a contradiction that there exists a $polyL$-complete problem $L$ for logspace reduction. Then, there exists $k \in \mathbb{N}$ such that $L \in SPACE(\log^k)$.

Let us show that $SPACE(\log^k) = SPACE(\log^{k+1})$, which is a contradiction with the space hierarchy theorem. Let $L' \in SPACE(\log^{k+1}) \subseteq polyL$. There exists a reduction $f$ of $L'$ to $L$ that can be computed in logarithmic space since $L$ is $polyL$-complete.

Now, consider a Turing machine that, on an input $w$, computes $f(w)$ in logarithmic space and then simulates a Turing machine deciding $L$ that runs in space $\log^k$ on $f(w)$. Note that here, it is important not store $f(w)$ on a working tape as this could make the space used exceed the $\log^k$ space bound. Instead, one must use a virtual tape where we only compute bits of $f(w)$ when they are needed without remembering the whole computation. Then, note that $|f(w)| = O(|w|^c)$ for some $c \geq 0$. Hence, the space used to check if $f(w)$ is in $L$ is lower than $\log^k(|f(w)|)$ hence is in $c^k \cdot \log^k(O(|w|)) = O(\log^k(|w|))$. We conclude with the speed-up theorem to get that $L' \in SPACE(\log^k)$. We get $SPACE(\log^k) = SPACE(\log^{k+1})$ which is in contradiction with the space hierarchy theorem. Hence $L$ cannot exist.

2. $PSPACE$ does have complete problems for logarithmic space reductions (such as TQBF). However, if we try to apply the previous proof to establish that $SPACE(n^k) = SPACE(n^{k+1})$, a problem arises: since $|f(w)|$ is in $O(|w|^c)$, we have $|f(w)|^k$ in $O(|w|^{c^k}) \neq O(|w|^k)$ if $c > 1$.

**Exercise 2** (Padding argument). 1. Show that if $DSPACE(n^c) \subseteq NP$ for some $c > 0$, then $PSPACE \subseteq NP$.

Hint: for $L \in DSPACE(n^k)$ one may consider the language $\tilde{L} = \{(x, w_x) \mid x \in L\}$, where $w_x$ is a word written in unary.

2. Deduce that $DSPACE(n^c) \neq NP$.

\(^1\)Note that, from this, we can deduce that $polyL \neq P$. 

---

Complexité avancée - TD 3

Benjamin Bordais

October 06, 2021
Solution 2. 1. Assume DSPACE(n^c) \subseteq NP and consider any \(L \in PSPACE\): we have to prove \(L \in NP\). For some \(k\), we have \(L \in DSPACE(n^k)\). Let \(M\) be a Turing Machine deciding \(L\) in space \(n^k\). Now, consider the language \(\tilde{L} = \{(x, 1^{\lfloor k/c \rfloor}) \mid x \in L\}\) and consider the Turing machine \(\tilde{M}\) that, on an input \(w\), checks that it has the form \(w = (x, 1^\ell)\), verifies that \(\ell = \lfloor |x|^{k/c} \rfloor\), and if so launches a simulation of \(M\) on \(x\). Note that computing \(|x|^{k/c}\) only uses \(k/c\) nested loops going from 1 to \(|x|\), which can be done in logspace since \(k/c\) is a “constant” that depends on \(M\), not \(x\). Then, \(\tilde{M}\) accepts \(\tilde{L}\) and the space used by \(\tilde{M}\) is in \(|x|^c = \lfloor |x|^{k/c} \rfloor \leq |w|^c\). Hence, \(\tilde{L} \in DSPACE(n^c) \subseteq NP\). Thus \(\tilde{L} \in NP\). As we can reduce \(L\) to \(\tilde{L}\) by transforming \(x\) into \((x, 1^{\lfloor k/c \rfloor})\) in logspace, we do have that \(L \in NP\).

2. Assume DSPACE(n^c) = NP, then DSPACE(n^{c+1}) \subseteq PSPACE = NP = DSPACE(n^c) which is in contradiction with the space hierarchy theorem.

Exercise 3 (On the existence of One-way function). A one-way function is a bijection \(f\) from \(k\)-bit integers to \(k\)-bit integers such that \(f\) is computable in polynomial time, but \(f^{-1}\) is not. Prove that for all one-way functions \(f\), we have

\[ A := \{(x, y) \mid f^{-1}(x) < y\} \in (NP \cap coNP) \setminus P \]

Solution 3. 1. \(A \in NP\): consider a Turing machine that, on an input \(w = (x, y)\), guesses a number \(c\) (with \(|c| = |x|\)) and checks in polynomial time that \(f(c) = x\) and \(c < y\). This non-deterministic TM runs in polynomial time and accepts the language \(A\).

2. \(A \in coNP \iff \{(x, y) \mid f^{-1}(x) \geq y\} \in NP\), which we solve as previously.

3. Assume that \(A \in P\). Then we build a Turing machine running in polynomial time that computes \(f^{-1}\): On an input \(x\) such that \(|x| = n\), there is \(2^n\) possibility for the value of \(f^{-1}(x)\). We consider a TM that proceeds by a binary search on the possible values \(v\) of \(f^{-1}(x)\) until it finds some \(v\) with \((x, v-1) \in A\) and \((x, v) \notin A\) and deduce \(f^{-1}(x) = v - 1\). Since at most \(n\) tests are necessary and each test is polynomial time, this TM runs in polynomial time.

Exercise 4 (Regular languages). Let REG denote the set regular/rational languages.

1. Show that for all \(L \in REG\), \(L\) is recognized by a TM running in space \(0\) and time \(n+1\).\(^2\)

2. Exhibit a language recognized by a TM running in space \(\log n\) and time \(O(n)\) that is not in REG.

Solution 4. 1. Consider \(L \in REG\). It is recognized by a finite automaton \(A\). We consider the TM with the same states than \(A\) that, on an input \(w\), simulates the execution of \(w\) on \(A\) and accepts if \(A\) does. This TM does not consumes any space and runs in time \(n+1\) (the \(n+1\)-th step reads the first blank after the input and accepts/rejects).

2. The language \(L = \{a^n \cdot b^n \mid n \geq 0\}\) is not regular and can be recognized by a TM that counts the number of \(a\) with a binary counter, decrements it for each \(b\) seen and accepts if, at the end of the word, the counter equal 0.

\(^2\)In fact, regular languages exactly correspond to languages that can be recognized in such a way.

2
Exercise 5 (Yet another NL-complete problem). For a finite set $X$, a subset $S \subseteq X$, and a binary operator $\ast : X \times X \rightarrow X$ defined on $X$, we inductively define $S_0,\ast := S$ and $S_{i+1,\ast} := S_i,\ast \cup \{x \ast y \mid x, y \in S_i,\ast \}$. The closure of $S$ with regard to $\ast$ is the set $S_\ast = \bigcup_{i\in\mathbb{N}}S_{i,\ast}$.

Show that the following problem AGEN is NL-complete.

- **Input**: A finite set $X$, a binary operation $\ast : X \times X \rightarrow X$ that is associative (i.e. $(x \ast y) \ast z = x \ast (y \ast z)$ for all $x, y, z \in X$), a subset $S \subseteq X$ and a target $t \in X$.
- **Output**: Yes if and only if $t \in S_\ast$.

Solution 5. 1. First, it has to be noted that the predicate: $\ast$ is associative can be checked in logspace (as it only amounts to loop on all triples of $X$ and check the table describing $\ast$).

2. AGEN is in NL: Consider a subset $S \subseteq X$ and an associative law $\ast : X \times X \rightarrow X$. Note that, we have $t \in S_\ast$ if and only if $t = x_1 \ast \ldots \ast x_n$ for some $x_1, \ldots, x_n \in S$. Since $\ast$ is associative, the parentheses do not matter. We denote by $x_{\leq i}$ the word $x_{\leq i} := x_1 \ldots x_i$. For the shortest such sequence generating $t$, we have that all $x_{\leq i}$ are distinct. It follows that we can guess the sequence of $x_i$ while only keeping a pointer on the current element $x_{\leq i}$, with the sequence being of size at most $|X|$.

3. We reduce from Path. Consider a directed graph $G = (V, E)$ and $s, t \in V$. We let $X := V \cup V \times V \cup \{\perp\}$ and $S := E \cup \{s\}$. We then define the law $\ast : X \times X \rightarrow X$ in the following way (every possibility that is not displayed here induces $\perp$):

   - For all $e, e' \in V$, $e \ast (e, e') := e'$;
   - For all $e, e', f \in V$, $(e, e') \ast (e', f) := (e, f)$.

   One can check that this law is associative, that $t$ is reachable from $s$ if and only if $t$ can be generated from $S$ and that this reduction can be computed in logspace.

Exercise 6 (Solving reachability games). A two player (turn-based) game is a directed graph $G = (V, E)$ where the set of vertices $V = V_A \uplus V_B$ is partitioned into vertices belonging to Player $A$ (i.e. $V_A$) and vertices belonging to Player $B$ (i.e. $V_B$) with a distinguished vertex $v_0 \in V$ that is the starting vertex. The graph is non-blocking in the sense that every vertex has a successor, i.e. $\text{Succ}(v) = \{v' \in V \mid (v, v') \in E\} \neq \emptyset$ for all $v \in V$. A play then corresponds to a finite or infinite path $\rho = v_0 \cdot v_1 \cdots \in V^* \cdot V^\omega$ with $v_0$ the starting vertex. If the play is at a node $v_i \in V_A$ then it is Player $A$’s turn to choose the next vertex $v_{i+1} \in \text{Succ}(v_i)$, it is Player $B$’s turn if $v_i \in V_B$. A winning condition sets when a play is winning for Player $A$ (we consider win/loose games, hence if Player $A$ does not win, Player $B$ does). A Player $C \in \{A, B\}$ has a winning strategy (or wins) from a vertex $v \in V$ if she can choose the next move in all vertices in $V_C$ such that she wins in any play that starts in $v$.

1. Assume that the winning condition is a reachability objective: given a target set of states $T \subseteq V$, Player $A$ wins if and only if a state in $T$ is seen at some point. Show that deciding the winner of a reachability game from the vertex $v_0 \in V$ can be done in polynomial time.

Hint: construct inductively the set of states from which Player $A$ can ensure to get closer to the target $T$ (that is called the attractor of the set $T$).
2. Consider some \( k \in \mathbb{N} \). A \( k \)-generalized reachability condition is the following: given \( k \) target sets of states \( T_1, \ldots, T_k \subseteq V \), Player A wins if and only if, for all \( 1 \leq i \leq k \), a state in \( T_i \) is seen at some point. Show that deciding the winner of a \( k \)-generalized reachability game from the vertex \( v_0 \in V \) can be done in polynomial time.

Solution 6. 1. Consider a reachability game \( G = (V,E) \) and \( T \subseteq V \). Let us define inductively the sequence of sets of states \( (X_i)_{i \in \mathbb{N}} \subseteq V^N \) with \( X_0 := T \) and, for all \( i \geq 0 \), we have:

\[
X_{i+1} := X_i \cup \{ v \in V_A \mid \text{Succ}(v) \cap X_i \neq \emptyset \} \cup \{ v \in V_B \mid \text{Succ}(v) \subseteq X_i \}
\]

Finally, let \( X := \bigcup_{i \in \mathbb{N}} X_i \subseteq V \). Then, we claim that Player A wins (i.e. has a winning strategy) if and only if \( v_0 \in X \).

First, for all \( x \in X \), we denote by \( i(x) \) the smallest index \( i \in \mathbb{N} \) such that \( x \in X_i \). That is, \( i(x) := \min\{ i \in \mathbb{N} \mid x \in X_i \} \). Assume that Player A chooses her move so that, for all \( v \in X \cap V_A \) with \( i(v) > 0 \), the successor chosen \( v' \in V \) is so that \( v' \in X \) and \( i(v') < i(v) \) (which is possible by definition of \( X \)). Note that, by definition of \( X \) and \( i \), for all \( v \in X \cap V_B \) such that \( i(v) > 0 \), we have \( \text{Succ}(v) \subseteq \{ v' \in X \mid i(v') < i(v) \} \). Now, any play \( p = x_0 \cdot x_1 \cdots \) starting in a vertex \( v \in X \) ensures \( i(x_0) > i(x_1) > \cdots \) until a vertex \( x_i \in X \) is so that \( i(x_i) = 0 \). That is, \( x_i \in X_0 = T \). It follows that Player A wins from any vertex in \( X \).

Similarly, let us show that Player B wins from any vertex not in \( X \). Assume that Player B chooses her move so that, for all \( v \in (V \setminus X) \cap V_B \), the successor chosen \( v' \in V \) is so that \( v' \in V \setminus X \) which is possible by definition of \( X \). Note that, for all \( v \in (V \setminus X) \cap V_A \), we have \( \text{Succ}(v) \subseteq (V \setminus X) \) by definition of \( X \). Now, consider a play \( p = x_0 \cdot x_1 \cdots \) starting at a vertex \( v_0 \in V \setminus X \). We have that, for all \( i \in \mathbb{N} \), if \( x_i \notin X \) then \( x_{i+1} \notin X \) and \( v_0 \notin X \). That is, no vertex in the play \( p \) is in \( X \), in particular, it never reaches \( T \). Hence, Player B wins from any vertex not in \( X \).

Overall, we have that Player A wins from \( v_0 \) if and only if \( v_0 \in X \). Furthermore, the set \( X \) can be computed in polynomial time (as it requires at most \(|V| \) loops over all states not in \( X \)). Therefore, determining the winner of a reachability game can be done in polynomial time.

2. We reduce to the previous problem. That is, we add a ‘memory’ to each that recalls the sets of targets that have already been seen. Specifically, consider a graph \( G = (V,E) \) and \( k \) targets \( T_1, \ldots, T_k \subseteq V \). For all vertex \( v \in V \), we denote by \( t(x) \subseteq \{1, \ldots, k\} \) the set of indexes of target sets to which it belongs: \( t(x) := \{ i \leq k \mid x \in T_i \} \). Now, we consider the reachability game \( G_k = (V_k, E_k) \) with \( T \subseteq V \) such that:

- \( V_k := V \times 2^{\{1, \ldots, k\}} \);
- \( E_k := \{ ((v, s), (v', s')) \mid (v, v') \in E \land s' = s \cup t(v) \} \);
- \( T := \{ (v, \{1, \ldots, k\}) \mid v \in V \} \)

In this construction, any play from a vertex \( v \in V \) can be translated into a play in \( G_k \) from \( (v, \emptyset) \in V_k \), and reciprocally. It follows that Player A wins in \( G \) from \( v_0 \) the \( k \)-generalized reachability game if and only if she wins from the reachability game \( G_k \) from \( (v_0, \emptyset) \). This can be decided in time \( O(|G_k|) = O(2^k \cdot |G|) = O(|G|) \) because \( k \) is a constant.