Foundations of Probabilistic Proofs (Fall 2022)
Note 3: IP for PSPACE
Date: 2022.09.27

This note contains definitions, theorems, facts, *etc.* that are not fully explained in lectures due to limited time. If you think there are anything missing or any mistakes, please contact ziyi.guan@epfl.ch.

Some of the definitions and the exercise presented in this note are adapted from the course materials of *Greats Ideas in Theoretical Computer Science* taught by Professor Anıl Ada at Carnegie Mellon University. We direct interesting readers to refer to the course website `https://www.cs251.com/` for more information.

1 IP=PSPACE by TSQBF

In the lecture, we see a proof for $\mathsf{IP} = \mathsf{PSPACE}$ by designing an interactive proof system for TQBF, a canonical PSPACE-complete problem. We adapt the idea for sumcheck protocol to this scenario, with the help of the degree reduction operation. However, there is another possible approach, which is the original proof Shamir gives. In particular, there is another language called TSQBF that has nice arithmetization that we can exploit. We explain here the definition of TSQBF, and the actual arithmetization is left as an exercise.

Definition 1. *We say that a fully quantified boolean formula is* **simple** *if every occurrence of every variable is separated from its quantification point by at most one universal quantifier ($\forall$) and arbitrarily many other symbols.*

This might be abstract, but let's look at some simple examples.

Example 1. *Determine if the following formulae are simple and compute their values:*

- $\forall x_1 \forall x_2 \exists x_3 \left((x_1 \vee x_2) \wedge x_3\right)$
  - *This is a simple formula.*
  - *It evaluates to* 0*, consider* $x_1 = 0, x_2 = 0$.
- $\exists x_1 \forall x_2 \left((x_1 \vee x_2) \wedge \forall x_3 (x_1 \vee x_3)\right)$
  - *This formula is not simple, the second* x_1 *is 2* $\forall$*'s away from its quantification.*
  - *It evaluates to 1, draw an evaluation tree as shown in lecture.*

Definition 2. TSQBF *is the set of languages that contains all simple true fully quantified Boolean formulae.*

We want to use this notion of simpleness in our proof of $\mathsf{IP} = \mathsf{PSPACE}$, and luckily it turns out that we can efficiently transform a fully quantified Boolean formula into a simple one with the same value. The general idea is to define a fresh variable for each occurrence of each variable in the original form. We formulate this claim in a more formal way below.

Lemma. *Let* Φ *be a fully quantified boolean formula with variables* $x_1, \ldots, x_n$*. We define a new formula* Ψ *that has a variable for each universal quantifier crossed by each variable in* Φ*. For example, if* x_1 *crosses* k *universal quantifiers in* Φ*, then* Ψ *has variables* $x_{1,1}, \ldots, x_{1,k}$.

Proof. We give a step-by-step proof idea of the lemma, the missing details are left as exercises to the readers.

1. Give a boolean formula which is true if and only if x_1 and x_2 are equal.
   - $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2)$
2. Let $\Phi = \exists x_1 \forall x_2 ((x_1 \vee x_2) \wedge \forall x_3 (x_1 \vee x_3))$. By replacing the two occurrences of x_1 with $x_{1,1}$ and $x_{1,2}$, and adding constraints and quantifiers, obtain a simple formula Ψ that has the same value as Φ.
   - $\exists x_{1,1} \forall x_2 ((x_{1,1} \vee x_2) \wedge \exists x_{1,2} (x_{1,1} \wedge x_{1,2}) \vee (\neg x_{1,1} \wedge \neg x_{1,2}) \wedge (\forall x_3 (x_{1,2} \vee x_3)))$
3. Give an efficient algorithm that transforms a QBF Φ into an equisatisfiable simple QBF Ψ.
4. Prove the algorithm's correctness.

□

2 TQBF is PSPACE-complete

In the lecture, we prove that IP = PSPACE. We describe PSPACE as the set of problems decidable within polynomial space by a Turing machine. However, we do not characterize Turing machines using the most formal mathematical languages. In this note, we revisit the definitions of Turing machines and PSPACE and give a formal treatment, which is useful in proving the PSPACE-completeness of TQBF.

For the sake of completeness, we present the definition of TQBF here. It is explained in detail in the lecture.

Definition 3 (True quantified Boolean formula)**.** *A fully quantified Boolean formula is a formula in quantified propositional logic where every variable is quantified (or bound), using either existential or universal quantifiers, at the beginning of the sentence. In particular, a fully quantified Boolean formula is of the form* $\Phi = \mathsf{Q}_1 x_1 \mathsf{Q}_2 x_2 \dots \mathsf{Q}_n x_n \varphi(x_1, x_2, \dots, x_n)$ *where* $\mathsf{Q}_i \in \{\forall, \exists\}$ *for* $i \in [n]$ *and* φ *is a boolean formula. The language* TQBF *is a formal language consisting of all the true quantified Boolean formulas.*

Since there is no free variables in a fully quantified Boolean formula Φ, Φ is equal to either true or false. We define TQBF to be the set of all true fully quantified Boolean formulae.

Example 2. *Let* $\Phi \coloneqq \forall x \exists y \exists z\ ((x \vee z) \wedge y)$, $\Phi \in$ TQBF *because* $(x \vee z) \wedge y$ *always evaluates to 1 when* $y = z = 1$.

Definition 4 (Turing machine)**.** *A Turing machine is a 7-tuple* $M \coloneqq (Q, \Gamma, \Sigma, \delta, q_{\text{start}}, Q_{\text{accept}}, Q_{\text{reject}})$ *where:*

- Q *is a finite, non-empty set, which we call a state set.*
- Γ *is a non-empty finite set that does not contain the blank symbol* $\sqcup$*, which we call the tape alphabet.*
- Σ *is a finite set such that* $\sqcup \in \Gamma$ *and* $\Sigma \subset \Gamma$*, which we call the input alphabet.*

- $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ *is a function, which we call the transition function.*
- $q_{\text{start}} \in Q$ *is the initial state.*
- $Q_{\text{accept}} \subseteq Q$ *is set of the accepting states.*
- $Q_{\text{reject}} \subseteq Q$ *is the set of rejecting states and* $Q_{\text{reject}} \cap Q_{\text{accept}} = \emptyset$.

Notice that for Turing machines, we always talk about its accompanying tape that is used as memory. The tape is just a sequence of cells that can hold any symbol from the tape alphabet. The tape can be defined so that it is infinite in two directions (so we could imagine indexing the cells using the integers $\mathbb{Z}$), or it could be infinite in one direction, to the right (so we could imagine indexing the cells using the natural numbers $\mathbb{N}$). Fixing to an indexing standard, we can imagine that there is a tape head that initially points to index 0. The symbol that the tape head points to at a particular time is the symbol that the Turing machine reads. The tape head moves left or right according to the transition function of the Turing machine. The following exercise is there to check your understanding of Turing machines.

Exercise 1 (Turing machines with different tapes)**.** *Show that the following four types of Turing machines are equivalent:*

- *A Turing machine with one singly-infinite tape (infinite in one direction).*
- *A Turing machine with one doubly-infinite tape (infinite in both directions).*
- *A Turing machine with however many tape heads you want.*
- *A Turing machine with 4 tapes and each with a separate tape head.*

One observation we can make after the definition of the Turing machines is that we are able to label the configuration of a Turing machine with countably many tuples. Note that a Turing machine is defined over a finite state set, a finite alphabet, and a tape indexable by natural numbers, which is countable. Therefore, at a given time during the running of a Turing machine, we could label the configuration of the Turing machine by a 3-tuple (q, i, x), where q is the state of the Turing machine at this time, the position of the tape head, and the content of the tape cell the head is pointing to.

We now define the configuration graphs of a Turing machine, which plays an important role in proving TQBF is PSPACE-complete.

Definition 5 (Configuration Graph)**.** *A configuration graph $G_{M,y}$ for a Turing machine M and input y is a directed graph. In particular, $V(G_{M,y}) = \{(q, i, x)\colon (q, i, x)$ are configurations of $M(y)\}$, and $E(G_{M,y}) = \{(u, v)\colon \delta(u) = v\}$. Note that here we are overloading the transition function δ because its input should not be a configuration tuple, here it means that u can be taken to v with a one-step computation from the transition function.*

Moreover, using the language of configuration graph, we say that $M(y)$ is accepting if there is a path from the starting configuration to an accepting configuration in $G_{M,y}$.

Theorem 1. TQBF *is* PSPACE*-complete.*

Proof. Following the standard paradigm of proving $\mathcal{C}$-completeness of a problem, we need to show that TQBF $\in$ PSPACE and then TQBF is PSPACE-hard.

- TQBF $\in$ PSPACE:
  - Our goal is to evaluate a fully quantified Boolean formula $\Phi = \mathsf{Q}_1 x_1 \mathsf{Q}_2 x_2 \dots \mathsf{Q}_n x_n \varphi(x_1, x_2, \dots, x_n)$, which has n variables and m clauses, within poly(n, m) space.
  - We follow something similar to the Shamir's protocol, peeling off the quantifiers one by one, starting with Q_1.
  - Let $\Phi_i(x_1, \dots, x_{n-i})$ be defined as follow:

    $$\Phi_i(x_1, \dots, x_{n-i}) \coloneqq \mathsf{Q}_{n-i+1} x_{n-i+1} \dots \mathsf{Q}_n x_n \varphi(x_1, \dots, x_n)$$

    Note that $\Phi_0 = \varphi$ and $\Phi_n = \Phi$.
  - We could recursively write Φ_i in terms of Φ_{i-1}:

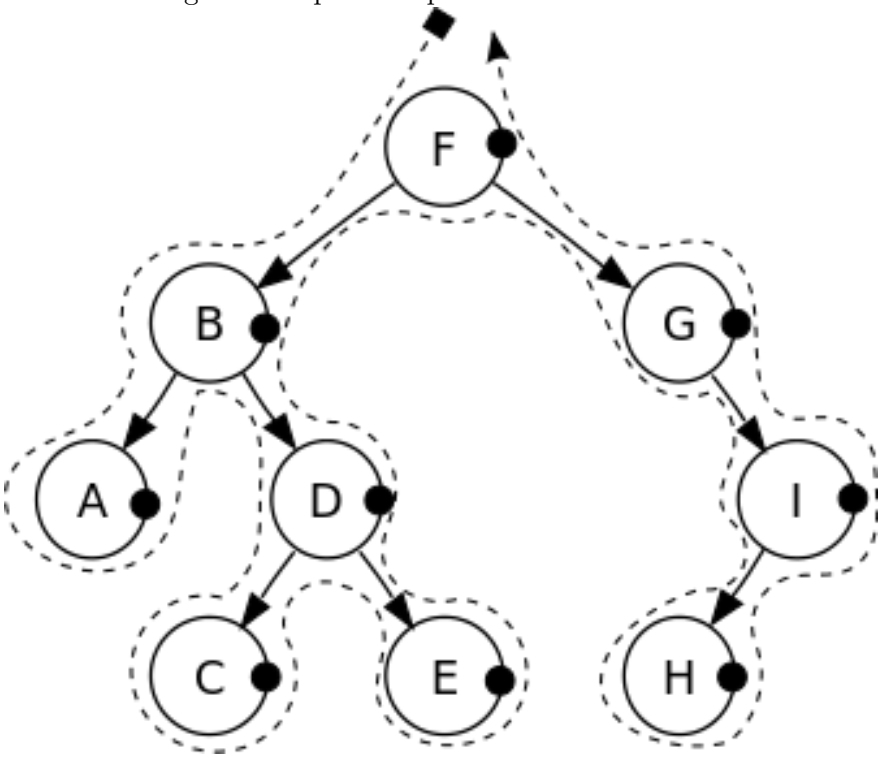
    $$\Phi_i(x_1, \dots, x_{n-i}) = \mathsf{Q}_{n-i+1} x_{n-i+1} \Phi_{i-1}(x_1, \dots, x_{n-i+1})$$

  - The recurrence can be viewed as a full binary tree on 2^n leaves.
    - A leaf node is $\Phi_0 = \varphi$ at some assignment in $\{0,1\}^n$. Evaluating a leaf node would cost precisely the size of the Boolean formula φ, which is $S_0 = O(mn)$ space.
    - An internal node is Φ_i at some assignment in $\{0,1\}^{n-i}$. It suffices to evaluate Φ_i by evaluating Φ_{i-1} with the current assignment plus $x_i = 0$ and $x_i = 1$ and recording this assignment in $\{0,1\}^{n-i}$. The space needed to evaluate an internal node is $S_i = S_{i-1} + O(n)$.
  - Evaluating $\Phi = \Phi_n$ costs $S_n = O(mn + n^2)$, which means that TQBF $\in$ PSPACE.
  - Notice that we are doing a depth-first post-order tree traversal (left subtree then right subtree then node). For example, the depth-first post-order tree traversal sequence for the tree in **??** is:

    $$A \to C \to E \to D \to B \to H \to I \to G \to F.$$

    We can reuse the same space by first evaluating the left subtree, storing a bit and then evaluating the right subtree.
  - Remark that we are ***not*** computing Φ_n in a bottom-up fashion layer by layer, which will cost $O(2^n)$ space. That is because the 2^n leaves have 2^{n-1} parent-nodes, and we should keep recording the first even if we are proceeding to the last, which costs $O(2^n)$ space.
- TQBF is PSPACE-hard:
  - Let $\mathcal{L}$ be a language in PSPACE, and M be the corresponding TM deciding $\mathcal{L}$ in space $S(n)$ where S is some polynomial $S \colon \mathbb{N} \to \mathbb{N}$.
  - Our goal is to show $\mathcal{L} \leq_m^P$ TQBF, that is, there exists a polynomial-time function f such that for any instance x, $x \in \mathcal{L}$ iff $f(x) \in$ TQBF.
  - Let $G_{M,x}$ be the configuration graph of M with input x, we have $x \in \mathcal{L}$ iff there is a path in $G_{M,x}$ from C_{start} to C_{accept}.
  - The goal can be reformulated as: find a polynomial f such that for any instance x, there is a path in $G_{M,x}$ from C_{start} to C_{accept} iff $f(x) \in$ TQBF.

Figure 1: depth-first post-order tree traversal



- We define a family of QBFs $\{\Phi_i\}_{i\in[O(S(n))]}$ recursively such that given two configurations C and C', $\Phi_i(C, C') = 1$ iff there exists a path in $G_{M,x}$ from C to C' of length at most 2^i.
- $f(x) \coloneqq \Phi_{O(S(n))}(C_{\text{start}}, C_{\text{accept}})$ is what we need for the reduction, and we have to argue that $\Phi_{O(S(n))}$ has polynomial size.
- The base case is to construct $\Phi_0(C, C')$ (which says that there is a one-step computation from C to C').
- We can write C and C' as:

$$\begin{aligned} C &\coloneqq (q, i, \gamma_0, \gamma_1, \ldots, \gamma_{O(S(n))}) \\ C' &\coloneqq (q', i', \gamma'_0, \gamma'_1, \ldots, \gamma'_{O(S(n))}) \end{aligned}$$

  where q, i and γ_j (respectively, q', i' and γ'_j) are a state in Q, the head position of the tape and the symbol in the j^{th} cell of the tape.
- The QBF $\Phi_0(C, C')$ have to capture the following propositions, which can be characterized as CNFs (see **??**):
  1. For every $j \neq i$, $\gamma_j = \gamma'_j$;
  2. Apply the transition function δ on q and γ_i will return q', γ'_i and $i' - i$.
- The recursive case is to construct Φ_i from Φ_{i-1}, that is:

$$\Phi_i(C, C') \coloneqq \exists C''\forall D_1\forall D_2((D_1 = C \wedge D_2 = C'') \vee (D_1 = C'' \wedge D_2 = C') \rightarrow \Phi_{i-1}(D_1, D_2)$$

Note that $\varphi_1 \to \varphi_2$ is equivalent to $\neg\varphi_1 \vee \varphi_2$.

- $|\Phi_i| = |\Phi_{i-1}| + p(S(n))$ where p is a polynomial.
- We conclude that $\Phi_{O(S(n))}$ has poly(n) size, which means that TQBF is PSPACE-hard.

□

Exercise 2. *For every Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, show that there is an n-variable CNF formula φ_f of size at most $n2^n$ such that $\varphi_f(u) = f(u)$ for every $u \in \{0,1\}^n$.*

Solution. Let $\varphi_f \coloneqq \bigwedge_{v\in\{0,1\}^n \wedge f(v)=0} C_v(z_1, \ldots, z_n)$, where for each v, $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. Hence, the constructed φ_f satisfies. ■