CSE547 Class 23

Jeremy Buhler

April 19, 2017

1 How Powerful is PSPACE?

- Previously, we saw the idea of NP-completeness.
- A problem in NP is NP-complete if it is "hard enough" that every problem in NP is polytime-reducible to it.
- We can use a similar technique to calibrate the hardness of other classes of problem/language.
- In particular, let's identify a complete problem for PSPACE.
- **Defn**: a language L is *PSPACE-complete* if
  - $L \in$ PSPACE
  - For all $L' \in$ PSPACE, $L' \leq_p L$
- (As before, just satisfying the second condition makes L "PSPACE-hard".)
- Note that we are using the same idea of reducibility – a polynomial-*time* transducer – as before.
- (A polynomial-space transducer would be trivial, since it would be powerful enough to just decide L' directly and spit out a fixed string in L or not in L as needed.)

2 The TQBF Problem

- Consider Boolean formulas augmented with quantifiers $\forall$ and $\exists$.
- A quantifier *binds* a variable inside the formula in which it occurs. For example, $\forall x.\exists y.(x \vee \neg y)$.
- WLOG, a quantified Boolean formula can move all quantifiers to the outside, resulting in *prenex normal form*, without changing the formula's size.
- We can reorder adjacent $\forall$ or adjacent $\exists$ quantifiers, but we can *never* reorder a $\forall$ and an $\exists$!
- (That is, $\forall x.\exists y.\phi$ is not the same as $\exists y.\forall x.\phi$.)
- A *fully-quantified Boolean formula* (QBF) is a formula in which every variable occurrence is bound by some quantifier.

- Given such a formula, we can ask whether it is true.
- "True" is defined for QBFs recursively:
  - If ϕ has no quantifiers (and hence no bound variables), it is true iff evaluating it yields true.
  - If $\phi = \forall x.\psi$, then ϕ is true iff ψ is true under both possible truth assignments to x.
  - If $\phi = \exists x.\psi$, then ϕ is true iff ψ is true under at least one possible truth assignment to x.
- For example, $\forall x.\exists y.(x \vee \neg y)$ is true. Indeed, if x is true, either choice for y will make the formula inside true, while if x is false, set y to false.
- However,
$$\exists y.\forall x.((x \vee y) \wedge (\neg x \vee \neg y))$$
is not true. (Try it!)
- Let TQBF be the language of all QBFs (in prenex normal form) that are true.

3 TQBF is PSPACE-Complete

Thm: TQBF is PSPACE-complete.

- First, let's check that TQBF is indeed in PSPACE.
- We define a procedure CHECK(ϕ, A) that takes a QBF ϕ and a partial truth assignment A to ϕ's variables and determines if ϕ is true.
- In the base case, if ϕ has no quantifiers, then A specifies values for all the variables in ϕ, and we return true iff A satisfies ϕ.
- In general, suppose ϕ has form $\forall x.\psi$.
- Then ϕ is true iff CHECK(ψ, $A \cup \{x = \text{true}\}$) and CHECK($\psi$, $A \cup \{x = \text{false}\}$) are both true.
- Suppose instead that ϕ has form $\exists x.\psi$.
- Then ϕ is true iff CHECK(ψ, $A \cup \{x = \text{true}\}$) or CHECK($\psi$, $A \cup \{x = \text{false}\}$) is true.
- Initially, we call CHECK(ϕ, $\emptyset$).
- Note that A has size $O(|\phi|)$. Hence, CHECK has recursion depth at most $O(|\phi|)$ and so can be done with at most $O(|\phi|^2)$ space.

OK, on to hardness!

- Let L be a language decidable by a TM M in space at most n^b, and hence in time at most 2^{dn^b}, for some constants d and b.

- Given an input w, we will construct a QBF ϕ of size polynomial in w that is true iff M accepts w.
- The construction is a "divide-and-conquer" form of the Cook-Levin tableau method.
- We will define a QBF $\phi_{c,c',t}$ that is true iff M can get from configuration c to configuration c' in at most t steps.
- As before, assume WLOG that M ends in a fixed accepting configuration c_a.
- Then M accepts w iff $\phi_{c_0(w),c_a,2^{dn^b}}$ is true.

OK, how can we build ϕ?

- As before, we can represent a configuration c by a list of $|\Gamma|n^b$ Boolean variables that specify its contents.
- In the base case, $t = 1$.
- $\phi_{c,c',1}$ uses the same 3x2 windowing trick as in Cook-Levin to express that either $c = c'$ or $c \vdash c'$ according to M.
- It is an ordinary Boolean formula with no quantifiers, of size $O(n^b)$ (the max size of configurations c and c').
- Now consider the case where $t > 1$.
- We will assume that t is a power of 2, since the top-level question we want to ask is for t a power of 2.
- *Idea*: as we saw, we can test whether it is possible to get from c to c' in t steps by testing all possible "halfway" points c_m.
- More precisely, $\phi_{c,c',t}$ is true iff there exists c_m such that both $\phi_{c,c_m,t/2}$ and $\phi_{c_m,c',t/2}$.
- In QBF language, we'd write

$$\phi_{c,c',t} = \exists c_m.(\phi_{c,c_m,t/2} \wedge \phi_{c_m,c',t/2})$$

- Here, $\exists c_m$ is actually $O(n^b)$ $\exists$ quantifiers – $|\Gamma|$ for each cell of configuration c_m.

Does this construction work out?

- We can recursively build up $\Phi = \phi_{c_0(w),c_a,2^{dn^b}}$.
- To make sure Φ is fully quantified, we need to stick $O(n^b)$ extra $\exists$ quantifiers on the outside to bind the variables for $c_0(w)$ and c_a.
- Φ is true iff M can get from $c_0(w)$ to c_a using only n^b space, since we only consider configurations of length n^b.
- But how big is Φ?
- The base case $t = 1$ uses $O(n^b)$ space.

- Each recursive step roughly doubles the amount of space used (plus an extra $O(n^b)$ for c_m's quantifiers).
- Conclude that Φ has size at least $O(n^b)2^{\log \ell}$, where ℓ is the length of M's computation.
- But ℓ can be exponential in n, so Φ itself is of size exponential in n.
- Hence, this construction is *invalid* – it is not a polynomial-space (and therefore not a polynomial-time) transducer.

Oh crap. Can we fix this problem?

- We can't double the formula size each time we split the computation.
- But we can take advantage of the fact that the two halves of the formula are the same, except for their configuration arguments.
- We can rewrite $\phi_{c,c',t}$ as

$$\phi_{c,c',t} = \exists c_m.\forall c_1.\forall c_2.[(((c_1 = c) \wedge (c_2 = c_m)) \vee ((c_1 = c_m) \wedge (c_2 = c'))) \implies \phi_{c_1,c_2,t/2}]$$

- The added subformulas have size $O(n^b)$, but they let us use only a single copy of the recursive subformula.
- With this modification, Φ has size $O(n^b) \log \ell$, where again ℓ is the length of M's computation.
- Since $\ell \leq 2^{dn^b}$, we conclude that Φ has size $O(n^{2b})$, which is indeed polynomial in n.
- Moreover, given w, we can construct $c_0(w)$ and the formula Φ in time $O(n^{2b})$, so our reduction is polynomial-time.
- Hence, $L' \leq_p$ TQBF.