CMPUT 675: Computational Complexity Theory **Winter 2019**

Lecture 8 (Jan 31): Time/Space Tradeoff for SAT and Circuits

Lecturer: Zachary Friggstad *Scribe: Noah Weninger*

8.1 Time/Space Tradeoff for SAT

Claim 1 SAT $\notin$ **TISP**(n^a, n^b) *for all* $a \geq 1, b > 0$ *such that* $a(\frac{a+b}{2}) < 1$.

To prove this claim, we instead show that **NTIME**$(n) \not\subseteq$ **TISP**(n^a, n^b). The result then follows from the discussion at the end of the previous lecture. But first, we need two lemmas.

Lemma 1 **TISP**$(n^a, n^b) \subseteq \mathbf{\Sigma_2TIME}(n^{\frac{a+b}{2}})$.

Proof. Let L be a language in **TISP**(n^a, n^b) with machine M deciding it, and take $d = \frac{a-b}{2}$. We can determine whether $x \in L$ by verifying the execution of a sequence of n^d snapshots of M, where adjacent snapshots have $O(n^{a-d})$ steps in between. Formally, $x \in L$ iff there are snapshots $C_0, \ldots, C_{n^d}$ such that:

(a) C_0 is the initial snapshot of M on x.

(b) C_{n^d} is an accepting snapshot.

(c) C_{i+1} is obtained from C_i after $O(n^{a-d})$ steps.

So, we have that

$$x \in L \Leftrightarrow \exists C_0, \ldots, C_{n^d}\ \forall i\ [C_i, C_{i+1} \text{ satisfy (a), (b), (c)}].$$

It takes $O(n^b \cdot n^d)$ time for the TM to seek ahead to the proper C_i when given i. Checking if C_{i+1} can be reached from C_i in $O(n^{a-d})$ steps takes $O(n^{a-d}) = O(n^{\frac{a+b}{2}})$ time by loading dedicated tapes with the tape contents of C_i, moving the heads of these tapes to their position in C_i and then then executing the instructions from M for $O(n^{a-d})$ steps starting with the state encoded in C_i.

Finally, it is easy to verify (a) and (b) hold in $O(n^b)$ once we have seeked ahead to the proper C_i. So, the total time to check if the properties are satisfied for some i in some sequence of configurations is $O(n^{\frac{a+b}{2}})$ and therefore $L \in \mathbf{\Sigma_2TIME}(n^{\frac{a+b}{2}})$. ■

Lemma 2 *For* $c, d \geq 1$, **NTIME**$(n) \subseteq$ **DTIME**$(n^c) \Rightarrow \mathbf{\Sigma_2TIME}(n^d) \subseteq$ **NTIME**(n^{cd}).

Proof. Assume **NTIME**$(n) \subseteq$ **DTIME**(n^c).

Let L be a language in $\mathbf{\Sigma_2TIME}(n^d)$ with a TM M that decides the predicate in n^d steps on input x where $|x| = n$. Since $L \in \mathbf{\Sigma_2TIME}(n^d)$, $x \in L$ iff $\exists u \in \{0,1\}^{|x|^d}$ such that $\forall v \in \{0,1\}^{|x|^d}$ $M(x, u, v) =$ ACCEPT. Note that u and v, the assignments to quantifier variables, may be of larger size than necessary for some particular x. To handle this we can assume that M ignores any extra bits in u and v.

Define $L' = \{(x,u) : |u| = |x|^d \text{ and } \exists v \in \{0,1\}^{|x|^d} \text{ such that } M(x,u,v) = \text{REJECT}\}$. For $n = |(x,u)|$, $L' \in \mathbf{NTIME}(n)$: we can nondeterministically guess a certificate v in $|x|^d \in O(n)$ steps, then deterministically run M in $O(n)$ steps. Then by our assumption $L' \in \mathbf{DTIME}(n^c)$ and so is its complement $\overline{L}'$.

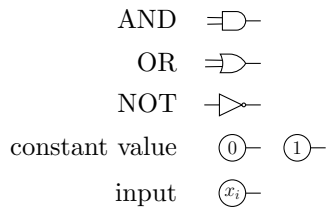
Let $\overline{M}$ be a TM which decides $\overline{L}'$ in time n^c. Then, using the definition of L, $x \in L \Leftrightarrow \exists u \in \{0,1\}^{|x|^d}$ such that $\overline{M}(x,u) = \text{ACCEPT}$. We know $\overline{M}(x,u)$ runs in $O((|x|+|u|)^c) = O(|x|^{cd})$ time, so $L \in \mathbf{NTIME}(n^{cd})$. ■

Proof of Claim 1. Suppose that $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^a, n^b)$. Let $k = \frac{a+b}{2} < 1$. Then $\mathbf{NTIME}(n^{1/k}) \subseteq \mathbf{TISP}(n^{a/k}, n^{b/k})$. By Lemma 1, $\mathbf{TISP}(n^{a/k}, n^{b/k}) \subseteq \mathbf{\Sigma_2 TIME}(n^{\frac{a+b}{2k}})$. By definition, we know $\mathbf{TISP}(n^a, n^b) \subseteq \mathbf{DTIME}(n^a)$ so by our assumption $\mathbf{NTIME}(n) \subseteq \mathbf{DTIME}(n^a)$. Then by Lemma 2, $\mathbf{NTIME}(n^{1/k}) \subseteq \mathbf{NTIME}(n^{a\frac{a+b}{2k}})$. But $\frac{1}{k} > a\frac{a+b}{2k}$ which contradicts the non-deterministic time hierarchy theorem that was not explicitly covered in a previous lecture, but can be found in the Chapter 3 of the course text [AB09]. ■

8.2 Circuits

We will now shift our discussion over to Boolean circuits. For the purposes of this discussion, we will make a few assumptions about the structure of a circuit. Our definition can be generalized in various ways, but this simple form will be sufficient here.

A circuit is a directed acyclic graph which describes the flow of some Boolean computation. Edges are implied to always point towards the right. All vertices belong to one of the following gate types:



There is one output, which is the only vertex that has a fan out of zero. All other nodes have arbitrary non-zero fan out. A circuit with n inputs has vertices with fan in zero labeled $x_1, \ldots, x_n$, though notice that not all vertices with fan in zero are inputs; some may be constants.

The type of the gate imposes a requirement on the fan in of the vertex. As suggested by the visual representations, AND and OR gates have a fan in of two. NOT gates have a fan in of one. Constant gates and inputs have a fan in of zero.

To compute the output of a circuit on some input $x_1, \ldots, x_n$, simply propagate values across edges, computing the appropriate functions at gates, until the output is known. Using this or any similar approach, the output of a circuit can be computed in an amount of time polynomial in its size. For a circuit C, we let $|C|$ denote the number of vertices in the circuit including input and constant vertices.

Definition 1 (Circuit Family)

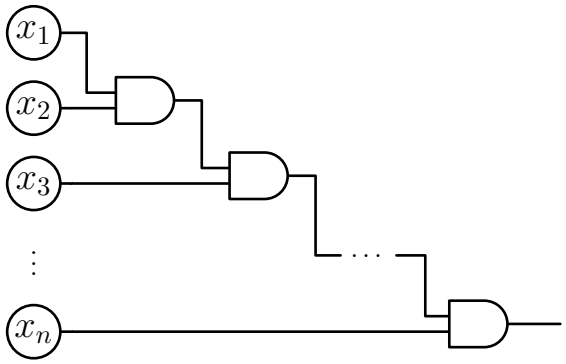
A family of circuits is the set $\{C_n\}_{n\geq 0}$ where each C_n has n inputs. We say a circuit family has size $f(n)$ if $|C_n| \in O(f(n))$.

Definition 2 (SIZE($f(n)$))

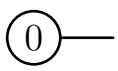
$L \in \mathbf{SIZE}(f(n))$ *iff there exists a family of circuits $\{C_n\}$ of size $f(n)$ where $\forall x \in \{0,1\}^*$ $x \in L \Leftrightarrow C_{|x|}(x) = 1$.*

Claim 2 *All unary languages L are in* **SIZE**(n).

Proof. Recall that a unary language only contains strings of the form 1^n. So, for every $1^n \in L$, the corresponding C_n in the circuit family simply needs to check that every input $x_i = 1$. This is easily accomplished by a chain of AND gates:



These circuits are of size $2n - 1$, which is $O(n)$. When $1^n \notin L$, we use the circuit that always outputs 0:



Which is $O(1)$, so the size of the family overall is linear. ■

Corollary 1 *There are undecidable languages in* **SIZE**(n).

For example, UNARY-HALT $= \{1^n :$ *the bits of n encode a TM which halts on the empty string*$\}$.

This example illustrates how the notion of **SIZE**$(f(n))$ is alone lacking in constraint: it suffices that an appropriate circuit family exists even if that circuit family cannot be constructed within $f(n)$ time (if at all). To remedy this we introduce the notion of a **P**-uniform circuit family.

Definition 3 (P-uniform circuit family)

A language L has a **P***-uniform family of circuits $\{C_n\}$ if $\{C_n\}$ has polynomial size and we can compute any circuit C_n in this family in poly(n) time.*

Theorem 1 *L has a* **P***-uniform family of circuits iff $L \in$* **P**.

Proof. First assume L has a **P**-uniform family of circuits. Then we can construct a polytime TM M to decide L. Given some input x, have M construct $C_{|x|}$ in polytime and then evaluate $C_{|x|}(x)$ in polytime. Therefore $L \in$ **P**.

The proof for the other direction follows a similar structure to the Cook-Levin theorem, which is detailed in Lecture 4. The difference is instead of constructing a SAT instance from a TM we can instead construct a circuit family. Here we present a rough sketch of the proof. Suppose $L \in$ **P** and let M be a poly-time TM deciding L in time $p(n)$.

Fix an input length n. Recall that we had defined the following variables:

$X_{c,\gamma,\tau}$: indicates that cell c contains γ at time τ

$Y_{h,\tau}$: indicates that the tape head is at position h at time τ

$Z_{q,\tau}$: indicates that the state is q at time τ

All variables from the Cook-Levin reduction will be vertices in the circuit, plus additional vertices to help connect them together. All variables with $\tau = 0$ will be constant vertices except for the $|x|$ variables corresponding to the first $|x|$ positions of the input tape, those will be inputs to the circuit. The other constant vertices are determined by the initial snapshot conditions: the start state must be q_{START}, the heads must be at position 0, and all other tape cells contain the blank $\square$.

All other variables correspond to an OR gate. We briefly sketch how they are placed in the circuit. For example, one condition is where the TM writes γ' to c and moves to cell h' and state q' if γ is on the tape cell h and q is the state at time τ:

$$X_{c,\gamma,\tau} \wedge Y_{h,\tau} \wedge Z_{q,\tau} \Rightarrow X_{c,\gamma',\tau+1} \wedge Y_{h',\tau+1} \wedge Z_{q',\tau+1}.$$

Note, this implication uniquely identifies the values of the variables on the right hand side if the left-hand side is true. We can use an AND vertex to and all of the variables (i.e. outputs of their vertices) and then have the output of this AND vertex feed into the inputs of all OR vertices for the variables on the right side of this implication. Using a similar trick with other implications in the proof of the Cook-Levin theorem and noting that every variable will be uniquely defined by just the initial conditions and implications from this theorem can be used in a careful proof by induction that the outputs of the vertices for the variables correspond to their actual value when computing $M(y)$ for any y with $|y| = n$.

Finally, the output vertex is the one corresponding to variable $Z_{q_{\text{ACCEPT}},p(|x|)}$. There may be more than one vertex with fan out 0, we can prune those efficiently to fit our convention that circuits only have one output.

Furthermore, careful inspection of the proof shows this reduction is computable in logarithmic space. Further details are given in Section 6.2 of [AB09]. ■

Corollary 2 CIRCUIT-VALUE $= \{(c, x) : c(x) = 1\}$ *is* $\mathbf{P}$*-complete with respect to implicitly computable log space reductions. The proof follows by hard-coding the inputs of the previous proof.*

Definition 4 ($\mathbf{P_{/poly}}$)

$\mathbf{P_{/poly}} = \cup_{c\geq 1}\mathbf{SIZE}(n^c)$ *(languages having a polynomial size circuit family)*

Theorem 2 (Karp-Lipton '80)

$\mathbf{NP} \subseteq \mathbf{P_{/poly}} \Rightarrow \mathbf{PH} = \mathbf{\Sigma}_2^p$.

Proof. Assume $\mathbf{NP} \subseteq \mathbf{P_{/poly}}$.

Recall $\mathbf{\Pi}_2$SAT $= \{\phi : \phi$ is an unquantified boolean formula where $\forall u \in \{0,1\}^n \ \exists v \in \{0,1\}^n \ \phi(u, v) = \text{TRUE}\}$.

Consider $L = \{(\phi, u) : \exists v \in \{0,1\}^{|u|} \ \phi(u, v) = \text{TRUE}\}$. Clearly $L \in \mathbf{NP}$: an assignment to v such that $\phi(u, v) = \text{TRUE}$ can be used as a polytime verifiable certificate. So by our assumption there exists a polynomial size circuit family $\{C_n\}$ for L.

Up until this point we have only considered circuits with a single output. However, we can safely extend our definition to allow multiple outputs, by observing that a circuit with m outputs can be effectively simulated with only polynomial slowdown by a list of circuits $C_1, \ldots, C_m$ which respectively produce each of the m outputs.

Given that a polysize circuit family can decide L, there must exist a polysize circuit family $\{C'_n\}$ with multiple outputs that can produce certificates for L, that is $\forall(\phi, u) \in L$, $C'_{|(\phi,u)|}(\phi, u) = v$ and $\phi(u, v) = \text{TRUE}$. This follows using arguments similar to how one shows that if $\mathbf{P} = \mathbf{NP}$ then in fact we can efficiently produce a circuit for any yes instance of any language in $\mathbf{NP}$. Then for some ϕ, we can "guess" a circuit C' to produce a certificate:

$$\phi \in \mathbf{\Pi}_2\text{SAT} \Leftrightarrow \exists w \in \{0,1\}^{poly(n)}\ \forall u \in \{0,1\}^n\ [w \text{ encodes a circuit } C' \text{ and } \phi(u, C'(\phi, u)) = \text{TRUE}]$$

Therefore $\mathbf{\Pi}_2\text{SAT} \in \mathbf{\Sigma}_2^p$. Since we know $\mathbf{\Pi}_2\text{SAT}$ is $\mathbf{\Sigma}_2^p$-complete with respect to Karp reductions, we have that $\mathbf{\Pi}_2^p \subseteq \mathbf{\Sigma}_2^p$. If we take some $L \in \mathbf{\Sigma}_2^p$ then $\overline{L} \in \mathbf{\Pi}_2^p$ so $\overline{L} \in \mathbf{\Sigma}_2^p$ and thus $L \in \mathbf{\Pi}_2^p$. It follows that $\mathbf{\Pi}_2^p = \mathbf{\Sigma}_2^p$, which implies $\mathbf{PH} = \mathbf{\Sigma}_2^p$ by a theorem from lecture 7, which can also be found in [AB09] as Theorem 5.4. ■

References

AB09 S. Arora and B. Barak, Computational Complexity: A Modern Approach, *Cambridge University Press, New York, NY, USA*, 2009.

BM04 P. Beame and T. Moore, Time-Space Tradeoffs for SAT, 2004.