Computational Models — Lecture 12

Handout Mode

Iftach Haitner.

Tel Aviv University.

January 8, 2018

---

1 Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.
Talk Outline

- Reminder
- Satisfiability
- Cook-Levin theorem
- CSAT $\in \mathcal{NPC}$
- 3SAT $\in \mathcal{NPC}$
- Clique is NPC

- Sipser’s book, 7.4–7.5
Section 1

Satisfiability
Boolean variables

- A Boolean variable assumes values
  - TRUE (written 1), and FALSE (written 0).
- Boolean operations:
  - and: $\wedge$
  - or: $\vee$
  - not: $\neg$
- Examples:

\[
\begin{align*}
0 \wedge 1 &= 0 \\
0 \vee 1 &= 1 \\
\neg 0 &= 1
\end{align*}
\]
Boolean formulas and SAT

A Boolean formula is an expression involving Boolean variables and operations.

\[ \phi = (\overline{x} \land y) \lor (x \land \overline{z}) \]

Definition 1 (satisfiable formula)

A formula is satisfiable, if some Boolean assignment to its variables, makes the formula evaluate to 1.

The formula \( \phi = (\overline{x} \land y) \lor (x \land \overline{z}) \) is satisfiable by the assignment

\[
\begin{align*}
x &= 0 \\
y &= 1 \\
z &= 0
\end{align*}
\]

The language of satisfied formulas:

\[ \text{SAT} = \{ \langle \phi \rangle : \phi \text{ is a satisfiable Boolean formula} \} \]
Theorem 2 (Cook-Levin (early 70s))

$\text{SAT} \in \mathcal{NP}$.

- The “most important" $\mathcal{NP}$-complete language.
- It is easy to see that $\text{SAT} \in \mathcal{NP}$
Section 2

Proving $SAT \in NPC$
The proof, high level

- Let $L \in \mathcal{NP}$ and let $N = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be an $t$-time NTM that accepts $L$, for some $t \in \text{poly}$.

- Given the string $w \in \{0, 1\}^*$, construct in time $O(p(|w|)^2)$ a formula $\phi_{N,w}$ such that: $\phi_{N,w} \in \text{SAT}$ iff $N$ accepts $w$.

- Hence, the mapping $w \mapsto \phi_{N,w}$ is a poly-time reduction from $L$ to $\text{SAT}$, establishing $L \leq_P \text{SAT}$.

- In the following fix $L$, $N$ and $w \in \{0, 1\}^n$.

- We assume for simplicity that $M(w)$ halts after exactly $t = t(n)$ steps.
The configuration-history Tableau

Consider the $t$-by-$t$ Tableau that describes a possible accepting computation history of $N$ on input $w$.

- First row represents initial configuration of $N$ on input $w$.
- $i$’th row represents the $i$-th configuration in a possible computation of $N$ on input $w$. 

\[
\begin{array}{cccccc}
1 & 2 & 3 & \ldots & t(n) \\
q_0 & 0 & 0 & 1 & 0 & \ldots \\
\end{array}
\]
The formula $\phi_{N,w}$

- Let $S = Q \cup \Gamma$ (the alphabet of the configuration history).
- $\phi_{N,w}$ uses Boolean variables $\{x_{i,j,s}\}_{i,j \in [t], s \in S}$.

$$\phi_{N,w} = \phi_{\text{Cell}(N)} \land \phi_{\text{Start}(w)} \land \phi_{\text{Move}(N)} \land \phi_{\text{Accept}(N)}$$

- Given an assignment $z$ for $\phi_{N,w}$, let $T(z)$ be the $t \times t$ Tableau, defined by setting the $j$-th cell in $i$'th configuration to $s$, if $x_{i,j,s} = 1$ in $z$.

$(T(z)$ is undefined, if $x_{i,j,s'} = x_{i,j,s} = 1$ for some $s \neq s' \in S$, or $x_{i,j,s} = 0$ for all $s \in S$).

- $T(z)$ will represent a (possible) accepting execution of $N(w)$ iff $z$ is an a satisfying assignment for $\phi_{N,w}$. 
The formula $\phi_{\text{Cell}(N)}$

$\phi_{\text{Cell}(N)}$ guarantees that the variables encode legal configurations:

- Each cell $(i, j)$ has at least one letter: $\bigvee_{s \in S} x_{i,j,s}$.
- No cell $(i, j)$ has two or more letters $\bigwedge_{s \neq s' \in S} \overline{x_{i,j,s}} \land \overline{x_{i,j,s'}}$.

Together:

$$\phi_{\text{Cell}(N)} = \bigwedge_{i,j} \left[ \left( \bigvee_{s \in S} x_{i,j,s} \right) \land \left( \bigwedge_{s \neq s' \in S} \overline{x_{i,j,s}} \land \overline{x_{i,j,s'}} \right) \right]$$

Claim 3

If an assignment $z$ satisfies $\phi_{\text{Cell}(N)}$, then $T(z)$ is defined.
The formula $\phi_{\text{Start}(w)}$

$\phi_{\text{Start}(w)}$ guarantees that the first row encodes the initial configuration (i.e., $q_0 w$).

$$
\phi_{\text{start}(w)} = x_{1,1} q_0 \land x_{1,2} w_1 \land x_{1,3} w_2 \land \ldots \land x_{1,n+1} w_n
$$

$$
\land x_{1,n+2} \land \ldots \land x_{1,t}
$$

Claim 4

If $z$ satisfies $\phi_{\text{Cell}(N)} \land \phi_{\text{Start}(w)}$, then the first line of $T(z)$ is $q_0 w \underbrace{\ldots \ldots \ldots}_{t-n-1}$. 
The formula $\phi_{\text{Move}(N)}$

$\phi_{\text{Move}(N)}$ is the “heart” of $\phi_{N,w}$. To construct it, we employ locality of computations.

Observation: Configuration $C$, with head location $h$, yields configuration $C'$ (with respect to $\delta$), if the following holds.

- $C'_i = C_i$ for any $i \notin \{h - 1, h, h + 1\}$
- $C'_{h-1,h,h+1}$ is consistent (with respect to $\delta$) with $C_{h-1,h,h+1}$.

We check that each configuration in $T(z)$ yields the next one, by inducing local “checks” on $z$.  

\[ \phi_{\text{Move}(N)} - \text{Rectangles} \]

- A rectangle is a \(2 \times 3\) configuration sub-table.
- Assume \(\delta(q_1, a) = \{(q_1, b, R)\}\) and \(\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}\).
- (some) Legal \(2 \times 3\) rectangles:
  
  \[
  \begin{array}{ccc}
  a & q_1 & b \\
  q_2 & a & c \\
  \end{array}
  \quad
  \begin{array}{ccc}
  a & q_1 & b \\
  a & a & q_2 \\
  \end{array}
  \quad
  \begin{array}{ccc}
  a & a & q_1 \\
  a & a & b \\
  \end{array}
  
  \begin{array}{ccc}
  a & b & a \\
  a & b & q_2 \\
  \end{array}
  \quad
  \begin{array}{ccc}
  b & b & b \\
  c & b & b \\
  \end{array}
  \quad
  \begin{array}{ccc}
  a & a & b \\
  a & a & b \\
  \end{array}
  
- (some) Illegal \(2 \times 3\) rectangles:
  
  \[
  \begin{array}{ccc}
  a & b & a \\
  a & a & a \\
  \end{array}
  \quad
  \begin{array}{ccc}
  a & q_1 & b \\
  q_1 & a & a \\
  \end{array}
  \quad
  \begin{array}{ccc}
  b & q_1 & b \\
  q_2 & b & q_2 \\
  \end{array}
  
- There is a constant number of legal rectangles (determined by \(\delta\)).
- Denote this set by \(C = C(\delta)\)
  (we will be somewhat liberal, allowing some illegal if notation simpler)
\[ \phi_{\text{Move}(N)} \] – Characterizing legal rectangles

The formula “verifies” that all \(2 \times 3\) rectangles in the Tableau are in the list \(C\):

1. \[
\begin{array}{ccc}
a & b & c \\
* & b & *
\end{array}
\]

2. \[
\begin{array}{ccc}
a & q & b \\
q' & a & b'
\end{array} \quad \begin{array}{ccc}
a & q & b \\
q' & a & b'
\end{array}
\]

\((L, q', b') \in \delta(q, b) \quad (R, q', b') \in \delta(q, b)\)

3. \[
\begin{array}{ccc}
q & * & * \\
* & * & *
\end{array} \quad \begin{array}{ccc}
* & * & q \\
* & * & *
\end{array}
\]

- Some rectangles in \(C\) are clearly illegal.
- For rectangles on the left-most and right-most side of Tableau, we use slightly different first type rectangles.
\( \phi_{\text{Move}}(N) \) – formal definition

- For each entry \((i, j) \in [t] \times [t]\) and \(c \in C\), let \(\phi_{\text{Move},i,j,c}\) be the formula taking the value 1 iff the \(2 \times 3\) table of cells in the Tableau whose upper-left corner is \((i, j)\) is \(c\).

For instance, for entry \((1, 1)\) and \(c = \begin{array}{ccc}
    a & q_1 & b \\
    q_2 & a & d
\end{array}\), let \(\phi_{\text{Move},1,1,c} = x_{1,1,a} \land x_{1,2,q_1} \land x_{1,3,b} \land x_{2,1,q_2} \land x_{2,2,a} \land x_{2,3,d}\).

Finally, let \(\phi_{\text{Move}}(N) = \bigwedge_{(i,j)} \bigvee_{c \in C} \phi_{\text{Move},i,j,c}\).

**Claim 5**

If \(z\) satisfies \(\phi_{\text{Cell}}(N) \land \phi_{\text{Start}}(w) \land \phi_{\text{Move}}(N)\), then \(T(z)\) is a possible configuration history of \(N(w)\).

**Proof:** By induction on the row index. Base case: \(z\) satisfies \(\phi_{\text{Cell}}(N) \land \phi_{\text{Start}}(w)\). Assume configuration defined in rows 1, \ldots, \(i\) is possible and head is in cell \(j\). The configuration of rows 1, \ldots, \(i + 1\) is also possible: Cells of indices not in \(\{j - 1, j, j + 1\}\), by first type of rectangles in \(C\). Other cells, by second type rectangles in \(C\). Q: Why do we need the third type of cells?
The formula $\phi_{\text{Accept}}(N)$

$\phi_{\text{Accept}}(N) \equiv \text{Accept}(N)$

guarantees that some row encodes an accepting configuration (i.e., $uqav$):

$$\phi_{\text{Accept}}(N) = \bigvee_{i,j} x_{i,j,q_a}$$

Claim 6

If $z$ satisfies $\phi_{N,w} = \phi_{\text{Cell}}(N) \land \phi_{\text{Start}}(w) \land \phi_{\text{Move}}(N) \land \phi_{\text{Accept}}(N)$, then $T(z)$ is an accepting configuration history of $N(w)$. 
Correctness of reduction

- The transformation $w \mapsto \phi_{N,w}$ is computable in time $O(n^{2c})$.
- An assignment satisfying $\phi_{N,w}$ corresponds to an accepting configuration history of $N(w)$.
- An accepting configuration history of $N(w)$ corresponds to an assignment satisfying $\phi_{N,w}$. (?)

Therefore, $N$ accepts $w$ iff $\phi_{N,w} \in SAT$.

- For complete details, consult Sipser chapter 7.4.
Section 3

$\text{CSAT} \in \mathcal{NPC}$
CSAT

It is useful to consider a special version of SAT.

- A literal is a variable or negated variable: $x$ or $\overline{x}$.
- A clause is several literals joined by $\lor$s: $(x_1 \lor x_2 \lor x_3)$
- A Boolean formula is in conjunctive normal form (CNF) if it consists of clauses, connected with $\land$s.
- For example: $(x_1 \lor \overline{x_2} \lor \overline{x_3} \lor x_4) \land (x_3 \lor \overline{x_5} \lor x_6) \land (x_3 \lor \overline{x_6})$
- The language of satisfied CNF formulas:

$$CSAT = \{ \langle \phi \rangle : \phi \text{ is satisfiable CNF formula} \}$$

- Clearly, $CSAT \leq_P SAT$ (no need Cook-Levin for that).
- The proof of Cook-Levin can be modified to show that $CSAT$ is NPC (take home exercise..)
Section 4

3SAT ∈ \mathcal{NPC}
3SAT

Definition 7

A Boolean formula is in **k-CNF form**, if it is a **CNF** formula, and all clauses have **k** literals.

Example of 3CNF: \((x_1 \lor \overline{x_2} \lor \overline{x_3}) \land (x_3 \lor \overline{x_5} \lor x_6) \land (x_3 \lor \overline{x_6} \lor x_4)\)

The language of satisfied 3CNF formulas:

\[3\text{SAT} = \{ \langle \phi \rangle : \phi \text{ is satisfiable 3CNF formula} \}\]

- Clearly \(3\text{SAT} \leq_P \text{SAT}\).
- and \(3\text{SAT} \in \overline{\text{NP}}\).
- We show next that \(\text{CSAT} \leq_P 3\text{SAT}\), yielding that \(3\text{SAT} \in \overline{\text{NP}}\).
- Since \(3\text{SAT}\) has more **structure** than \(\text{CSAT}\), it is simpler to reduce it to other languages.
1SAT, 2SAT $\in \mathcal{P}$

**Question 8**
Is $1\text{SAT} \in \mathcal{P}$?

**Question 9**
Is $2\text{SAT} \in \mathcal{P}$?

Yes, see appendix...
CSAT $\leq_P$ 3SAT

- The reduction maps CNF formulae to 3CNF ones “clause by clause”.
- A clause with $d \leq 3$ literals is mapped to equivalent clause with 3 literals.
  \[(x_1 \lor x_2) \mapsto (x_1 \lor x_2 \lor x_2)\]
- A clause with $d > 3$ literals is mapped to $d - 2$ clauses, built on the original literals together with $(d - 3)$ new variables.
  \[
\phi = (x_1 \lor \overline{x_2} \lor \overline{x_3} \lor x_4 \lor x_8) \mapsto \\
\phi_3 = (x_1 \lor \overline{x_2} \lor y_1) \land (\overline{y_1} \lor \overline{x_3} \lor y_2) \land (\overline{y_2} \lor x_4 \lor x_8)
\]
The reduction works!

Claim 10

$\phi$ has a satisfying assignment iff $\phi_3$ does.

Proof’s idea: (for case $d > 3$)

$\Leftarrow$ Given a satisfying assignment for $\phi_3$, use the settings of the $x_i$’s to get satisfying assignment for $\phi$. Why does it work? For each set of clauses that came from a clause in $\phi$, note that each $y_i$ sets exactly one clause to true. There are $k - 2$ clauses but only $k - 3$ $y_i$’s. So at least one clause needs to be satisfied by one of the $x_j$ or $\overline{x_j}$ literals. Set this literal to true in $\phi$.

$\Rightarrow$ An assignment satisfying $\phi$, makes at least one literal per clause happy. In the “$\phi_3$ clause” of this literal the new variables are under no constraints. Use the $d - 2$ new variables to set the remaining $d - 2$ clauses to “true”.
CSAT $\leq_P$ 3SAT, cont.

$\phi = (x_1 \lor \overline{x_2} \lor \overline{x_3} \lor x_4 \lor x_8) \quad \rightarrow \quad \phi_3 = (x_1 \lor \overline{x_2} \lor y_1) \land (\overline{y_1} \lor \overline{x_3} \lor y_2) \land (\overline{y_2} \lor x_4 \lor x_8)$

- Doing the above mapping to each clause of a CNF formula, we get a 3CNF that is satisfied iff the original one is.
- Since this mapping is polynomial time (?), we get CSAT $\leq_P$ 3SAT. ♣.
Section 5

Clique is NPC
Claim 11

\[ 3SAT \leq_P CLIQUE. \]

Since \( CLIQUE \in \mathcal{NP} \), it follows that \( CLIQUE \in \mathcal{NPC} \).

Since \( IND-SET \in \mathcal{NP} \), and \( CLIQUE \leq_P IND-SET \), it follows that \( IND-SET \in \mathcal{NPC} \).

Proof’s idea: We’ll construct a poly-time reduction \( f \) that maps 3\textit{CNF} formulae \( \phi \) to pairs \( \langle G, k \rangle \) of graphs and numbers.

The function \( f \) will have the property that \( \phi \) is satisfiable, iff \( G \) has a clique of size \( k \).
Proving $\text{3SAT} \leq_P \text{CLIQUE}$

On input $\phi$, a 3CNF formula, the mapping reduction is defined as follows:

let $k$ be the number of clauses in $\phi$.

We construct a graph $G = G(\phi)$, see below, and output $(G, k)$.

The graph $G$ is defined as follows:

- Nodes in $G$ are organized into triples $t_1, \ldots, t_k$.
- Each triple corresponds to a clause of $\phi$.
- Each node in a triple corresponds to a literal in corresponding clause.

Ongoing example:

$$(x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_3} \lor x_5 \lor x_6) \land (x_3 \lor \overline{x_6} \lor x_4)$$
Nodes of $G$

$$
\phi = (x_1 \lor \overline{x_2} \lor x_3) \land (x_3 \lor x_5 \lor x_6) \land (x_3 \lor x_4 \lor \overline{x_6})
$$

Add a node per literal
Edges of $G$

Add edges between all vertex pairs, except

- within same triple
- between contradictory literals (e.g., $x_3$ and $\overline{x_3}$)

$$\phi = (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_3} \lor x_5 \lor x_6) \land (x_3 \lor x_4 \lor \overline{x_6})$$
If \( \phi \in 3\text{SAT} \rightarrow \langle G, k \rangle \in \text{CLIQUE} \)

**Proof:** Suppose \( \phi \) is satisfiable by an assignment \( \psi \).

With respect to \( \psi \):

- At least one literal is assigned to 1 in every clause (\( ? \))
- Select a 1-literal in every tuple;
- These literals can be joined by edges (\( ? \))

Yielding a \( k \)-clique in \( G \).
\[ \langle G, k \rangle \in \text{CLIQUE} \implies \phi \in \text{3SAT} \]

Proof: Suppose \( G \) has a \( k \)-clique.

- No two of the cliques nodes are in the same triple. (?)
- \( G \) has \( 3k \) vertexes and \( k \) clauses, so each triple has exactly one clique node.
- Assign 1 to each node in clique
- Assignment has no contradictions (?)
  Yielding a satisfying assignment to \( \phi \).
Recap

- We’ve constructed a poly-time computable function $f$.
- We saw that $f$ has the property that $\phi \in 3SAT$ iff $f(\phi) \in CLIQUE$.
- Therefore, $f$ is a poly-time reduction from $3SAT$ to $CLIQUE$

$$\implies 3SAT \leq_P CLIQUE.$$ ⌲
Section 6

Hamiltonian Paths and Hamiltonian Cycles are \( \text{NPC} \)
Reminder – Hamiltonian Path

A Hamiltonian path in a directed $G$ visits each node exactly once.

$$\text{HAMPATH} = \{⟨G, s, t⟩: G \text{ has Hamiltonian path from } s \text{ to } t\}$$

**Theorem 12**

$$\text{HAMPATH} \in \mathcal{NPC}.$$
A Hamiltonian cycle in a graph is an Hamiltonian path that ends up where it starts (i.e., \( s = t \)).

\[
\text{HAMCYCLE} = \{ \langle G \rangle : G \text{ has Hamiltonian cycle} \}
\]

Clearly \( \text{HAMCYCLE} \in \mathcal{NP} \). We will show that \( \text{HAMPATH} \leq_P \text{HAMCYCLE} \), and deduce that \( \text{HAMCYCLE} \in \mathcal{NPC} \)
HAMPATH $\leq_P$ HAMCYCLE.

Proof’s idea:

Hey, is the new vertex really needed? Why not just add an edge from $t$ to $s$?
HAMPATH $\leq_P$ HAMCYCLE.

Why the new vertex really needed? Why not just add an edge from $t$ to $s$?
Claim 13

\[ \text{HAMCYCLE} \leq_P \text{HAMPATH} \]

Left as an easy (recommended) exercise.
**Undirected Hamiltonian Circuit**

\[ \text{UHAMCYCLE} = \{ \langle G \rangle : G \text{ is undirected & has Hamiltonian cycle} \} \]

where Hamiltonian cycle/path in an undirect graph, is defined analogously to the direct case.

Clearly \( \text{UHAMCYCLE} \in \mathcal{NP} \).

It is not hard to show (see Sipser 7.55, for a similar proof) that \( \text{HAMCYCLE} \leq_p \text{UHAMCYCLE} \), and deduce that \( \text{UHAMCYCLE} \in \mathcal{NPC} \).
CSAT $\leq_P$ HAMPATH

For any CNF formula $\phi$ with clauses $c_1, \ldots, c_\ell$ and variables $x_1, \ldots, x_k$, we construct a directed graph $G$ with vertices $s$ and $t$, such that $\phi$ is satisfiable iff $\exists$ a directed Hamiltonian path from $s$ to $t$.

thus establishing

**Theorem 14**

HAMPATH, HAMCYCLE $\in \mathcal{NP}$

Turn to a separate pdf presentation:

Part II

Appendix
Not Taught in Class
Section 7

$2\text{SAT} \in \mathcal{P}$
A graph characterization for 2CNF

**Definition 15**

For 2CNF formula $\phi$ with variables $x_1, \ldots, x_k$, define a directed graph $G(\phi) = (V, E)$ as

- $V = \{x_1, \overline{x_1}, \ldots, x_k, \overline{x_k}\}$
- $E = \{((\ell), h), (\overline{h}, \ell) : (\ell \lor h) \in \phi\}$ (taking $\overline{x} = x$).

**Claim 16**

$G(\phi)$ contains path from $\ell$ to $h \implies$ in any satisfying assignment of $\phi$ with $\ell = 1$, it holds that $h = 1$.

Proof?

**Claim 17**

$G(\phi)$ contains path from $\ell$ to $h \implies G(\phi)$ contains path from $\overline{h}$ to $\overline{\ell}$

Proof?
A graph characterization for $2CNF$, cont

**Definition 18**
A variable $x$ in $2CNF$ $\phi$ is bad, if $\exists$ paths in $G(\phi)$ from $x$ to $\overline{x}$ and from $\overline{x}$ to $x$.

**Claim 19**
A $2CNF$ formula is satisfiable iff it contains no bad variables.

- $\phi$ has bad variable $\implies \phi$ is unsatisfiable. Proof? by Claim 16
- $\phi$ has no bad variables $\implies \phi$ is satisfiable. Proof?
A graph characterization for $2CNF$, cont.

**Definition 20**

A variable $x$ in $2CNF \phi$ is **bad**, if there exist paths in $G(\phi)$ from $x$ to $\overline{x}$ and from $\overline{x}$ to $x$.

**Claim 21**

A $2CNF$ formula is satisfiable **iff** it contains no bad variables.

**Algorithm 22**

While $\phi$ has unassigned variables:

1. Pick **unassigned** literal $\ell \in V$ that has no path from $\ell$ to $\overline{\ell}$
2. Assign 1 to all literals reachable from $\ell$, and 0 to their negations.
3. Recompute graph for "reduced formula"

Can there be conflicts (i.e., $x$ and $\overline{x}$ assigned the same value)?

- **Same iteration?** Assume $\ell \rightsquigarrow h \rightsquigarrow \overline{h}$. \(\Longrightarrow\) $h \rightsquigarrow \overline{h} \rightsquigarrow \overline{\ell} \Longrightarrow \ell \rightsquigarrow \overline{\ell}$.
- **Different iterations?** Removing edges cannot add bad variables.

Final assignment satisfies $\phi$. Assume $h$ is set to 1, then all clauses it participates in are satisfied.
Poly-time algorithm for 2SAT

Algorithm 23 (TwoSatSolver)

Input: 2CNF \( \phi \)
Return TRUE if there exist no bad variables in \( \phi \):

- Efficiency?
- Correctness?