CSC 373: Algorithm Design and Analysis
Lecture 18

Allan Borodin

March 1, 2013

Material for NP completeness of SAT is from MIT Open Courseware spring 2011 course at http://tinyurl.com/bjde5o5 .

Announcements and Outline

Announcements

- Term test 2 on Monday in tutorial rooms.
- Please hand back your test if you have not already done do. I need to record the grade. I am still missing some tests to be recorded.
- You must keep all graded work until the term is over just in case there is some inconsistency in the grades recorded and what you have.
- Please refer to the web page for my policy on regrading.

Today's outline

- Answer questions about assignment
- Brief introductio to Turing machines.
- The NP completeness of SAT

Brief introduction to Turing machines

- We are using the classical one tape TM. This is the simplest variant to formalize which will enable the proof for the NP completeness of SAT. In the proof, we are assuming (without loss of generality) that all time bounds $T(n)$ are computable in polynomial time.
- Claim: Any reasonable (classical) computing model algorithm running in time $T(n)$, can be simulated by a TM in time $T(n)^k$ for some k. Hence we can use the TM model in the definition of P and NP.
- Since we are only considering decision problems we will view TMs that are defined for decision problems and hence do not need an output other than a reject and accept state.
- Following the notation in the MIT lecture notes, formally, a specific TM is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$
- We briefly explain (using the board) the model and notation. Note that Q, Σ, Γ are all finite sets.
- $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$ is the (finite) transition function

Satisfiability is NP-Complete

- SAT = { < ϕ > | ϕ is a satisfiable Boolean formula }
- Theorem: SAT is NP-complete.
- Lemma 1: SAT $\in$ NP.
- Lemma 2: SAT is NP-hard.
- Proof of Lemma 1:
  - Recall: L $\in$ NP if and only if ($\exists$ V, poly-time verifier) ($\exists$ p, poly)
    x $\in$ L iff ($\exists$ c, |c| $\leq$ p(|x|)) [V(x, c) accepts]
  - So, to show SAT $\in$ NP, it's enough to show ($\exists$ V) ($\exists$ p)
    ϕ $\in$ SAT iff ($\exists$ c, |c| $\leq$ p(|x|)) [V(ϕ, c) accepts]
  - We know: ϕ $\in$ SAT iff there is an assignment to the variables such that ϕ with this assignment evaluates to 1.
  - So, let certificate c be the assignment.
  - Let verifier V take a formula ϕ and an assignment c and accept exactly if ϕ with c evaluates to true.
  - Evaluate ϕ bottom-up, takes poly time.

Satisfiability is NP-Complete

- Lemma 2: SAT is NP-hard.
- Proof of Lemma 2:
  - Need to show that, for any $A \in NP$, $A \leq_p SAT$.
  - Fix $A \in NP$.
  - Construct a poly-time f such that

    $w \in A$ if and only if $f(w) \in SAT$.

    A formula, write it as ϕ_w.
  - By definition, since $A \in NP$, there is a nondeterministic TM M that decides A in polynomial time.
  - Fix polynomial p such that M on input w always halts, on all branches, in time $\leq p(|w|)$; assume $p(|w|) \geq |w|$.
  - $w \in A$ if and only if there is an accepting computation history (CH) of M on w.

Satisfiability is NP-Complete

- Lemma 2: SAT is NP-hard.
- Proof, cont'd:
  - Need $w \in A$ if and only if $f(w)$ $(= \phi_w) \in$ SAT.
  - $w \in A$ if and only if there is an accepting CH of M on w.
  - So we must construct formula ϕ_w to be satisfiable iff there is an accepting CH of M on w.
  - Recall definitions of computation history and accepting computation history from Post Correspondence Problem: # C_0 # C_1 # C_2 …
    - Configurations include tape contents, state, head position.
  - We construct ϕ_w to describe an accepting CH.
  - Let M = (Q, Σ, Γ, δ, q_0, q_{acc}, q_{rej}) as usual.
  - Instead of lining up configs in a row as before, arrange in (p(|w|) + 1) row $\times$ (p(|w|) + 3) column matrix:

Proof that SAT is NP-hard

- ϕ_w will be satisfiable iff there is an accepting CH of M on w.
- Let M = (Q, Σ, Γ, δ, q_0, q_{acc}, q_{rej}).
- Arrange configs in (p(|w|) + 1) $\times$ (p(|w|) + 3) matrix:

#	q_0	w_1	w_2	w_3	…	w_n	--	--	…	--	#
#	…										#
#	…										#
⋮											⋮
#	…										#

- Successive configs, ending with accepting config.
- Assume WLOG that each computation takes exactly p(|w|) steps, so we use p(|w|) + 1 rows.
- p(|w|) + 3 columns: p(|w|) for the interesting portion of the tape, one for head and state, two for endmarkers.

Proof that SAT is NP-hard

- ϕ_w is satisfiable iff there is an accepting CH of M on w.
- Entries in the matrix are represented by Boolean variables:
  - Define $C = Q \cup \Gamma \cup \{ \# \}$, alphabet of possible matrix entries.
  - Variable $x_{i,j,c}$ represents "the entry in position (i, j) is c".
- Define ϕ_w as a formula over these $x_{i,j,c}$ variables, satisfiable if and only if there is an accepting computation history for w (in matrix form).
- Moreover, an assignment of values to the $x_{i,j,c}$ variables that satisfies ϕ_w will correspond to an encoding of an accepting computation.
- Specifically, $\phi_w = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$, where:
  - ϕ_{cell} : There is exactly one value in each matrix location.
  - ϕ_{start} : The first row represents the starting configuration.
  - ϕ_{accept} : The last row is an accepting configuration.
  - ϕ_{move} : Successive rows represent allowable moves of M.

ϕ_{cell}

- For each position (i,j), write the conjunction of two formulas:

  $\bigvee_{c \in C} x_{i,j,c}$: Some value appears in position (i,j).

  $\bigwedge_{c, d \in C, c \neq d} (\neg x_{i,j,c} \vee \neg x_{i,j,d})$: Position (i,j) doesn't contain two values.

- ϕ_{cell}: Conjoin formulas for all positions (i,j).

- Easy to construct the entire formula ϕ_{cell} given w input.
- Construct it in polynomial time.
- Sanity check: Length of formula is polynomial in |w|:
  - $O((p(|w|)^2)$ subformulas, one for each (i,j).
  - Length of each subformula depends on C, $O(|C|^2)$.

ϕ_{start}

- The right symbols appear in the first row:

$$\# \quad q_0 \quad w_1 \quad w_2 \quad w_3 \quad \dots \quad w_n \quad -- \quad -- \quad \dots \quad -- \quad \#$$

$$\phi_{start}: \; x_{1,1,\#} \wedge x_{1,2,q0} \wedge x_{1,3,w1} \wedge x_{1,4,w2} \wedge \dots$$
$$\wedge \; x_{1,n+2,wn} \wedge x_{1,n+3,--} \wedge \dots$$
$$\wedge \; x_{1,p(n)+2,--} \wedge x_{1,p(n)+3,\#}$$

ϕ_{accept}

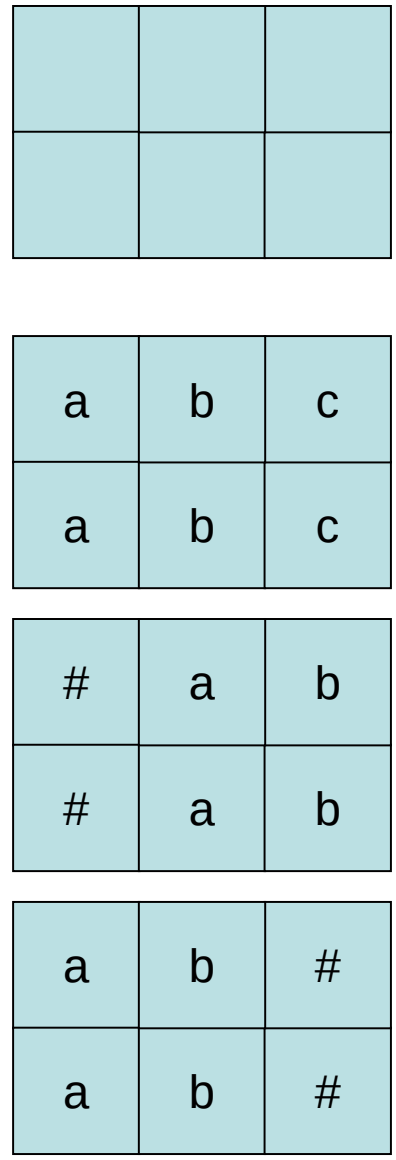
- For each j, $2 \leq j \leq p(|w|) + 2$, write the formula:

$$x_{p(|w|)+1,j,qacc}$$

- q_{acc} appears in position j of the last row.
- ϕ_{accept}: Take disjunction (or) of all formulas for all j.
- That is, q_{acc} appears in some position of the last row.

ϕ_{move}

- As for PCP, correct moves depend on correct changes to local portions of configurations.
- It's enough to consider 2×3 rectangles:
- If every 2×3 rectangle is "good", i.e., consistent with the transitions, then the entire matrix represents an accepting CH.
- For each position (i,j), $1 \leq i \leq p(|w|)$, $1 \leq j \leq p(|w|)+1$, write a formula saying that the rectangle with upper left at (i,j) is "good".
- Then conjoin all of these, $O(p(|w|)^2)$ clauses.
- Good tiles for (i,j), for a, b, c in Γ:



ϕ_{move}

- Other good tiles are defined in terms of the nondeterministic transition function δ.
- E.g., if $\delta(q_1, a)$ includes tuple (q_2, b, L), then the following are good:
  - Represents the move directly; for any c:
  - Head moves left out of the rectangle; for any c, d:
  - Head is just to the left of the rectangle; for any c, d:
  - Head at right; for any c, d, e:
  - And more, for #, etc.
- Analogously if $\delta(q_1, a)$ includes (q_2, b, R).
- Since M is nondeterministic, $\delta(q_1, a)$ may contain several moves, so include all the tiles.

c	q_1	a
q_2	c	b

q_1	a	c
d	b	c

a	c	d
b	c	d

d	c	q_1
d	q_2	c

e	d	c
e	d	q_2

ϕ_{move}

- The good tiles give partial constraints on the computation.
- When taken together, they give enough constraints so that only a correct CH can satisfy them all.
- The part (conjunct) of ϕ_{move} for (i,j) should say that the rectangle with upper left at (i,j) is good:
- It is simply the disjunction (or), over all allowable tiles, of the subformula:

a1	a2	a3
b1	b2	b3

$$x_{i,j,a1} \wedge x_{i,j+1,a2} \wedge x_{i,j+2,a3} \wedge x_{i+1,j,b1} \wedge x_{i+1,j+1,b2} \wedge x_{i+1,j+2,b3}$$

- Thus, ϕ_{move} is the conjunction over all (i,j), of the disjunction over all good tiles, of the formula just above.

ϕ_{move}

- ϕ_{move} is the conjunction over all (i,j), of the disjunction over all good tiles, of the given six-term conjunctive formula.
- Q: How big is the formula ϕ_{move}?
- $O(p(|w|)^2)$ clauses, one for each (i,j) pair.
- Each clause is only constant length, O(1).
  - Because machine M yields only a constant number of good tiles.
  - And there are only 6 terms for each tile.
- Thus, length of ϕ_{move} is polynomial in |w|.
- $\phi_w = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$, length also poly in |w|.

ϕ_{move}

- $\phi_w = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$, length poly in |w|.
- More importantly, can produce ϕ_w from w in time that is polynomial in |w|.
- $w \in A$ if and only if M has an accepting CH for w if and only if ϕ_w is satisfiable.
- Thus, $A \leq_p SAT$.
- Since A was any language in NP, this proves that SAT is NP-hard.
- Since SAT is in NP and is NP-hard, SAT is NP-complete.

Clay Math Institute Millenium Problems: $1,000,000 each

1. Birch and Swinnerton-Dyer Conjecture
2. Hodge Conjecture
3. Navier-Stokes Equations
4. P = NP?
5. Poincaré Conjecture **(Solved)**[1]
6. Riemann Hypothesis
7. Yang-Mills Theory

[1] Solved by Grigori Perelman 2003: Prize unclaimed

How important is the P vs NP question

Lance Fortnow has an article on P and NP in the September 2009 Communications of the ACM, in which he says

> *"The P versus NP problem has gone from an interesting problem related to logic to perhaps the most fundamental and important mathematical question of our time, whose importance only grows as computers become more powerful and widespread."*

Claim: It is worth well over the $1,000,000