CS 740 – Computational Complexity and Algorithm Analysis

Spring Quarter 2012

Slides 2

Pascal Hitzler
Kno.e.sis Center
Wright State University, Dayton, OH
http://www.knoesis.org/pascal/
1. SAT is in NP
2. SAT is NP-hard
SAT is in NP

\[ F = \bigg( \bigwedge_{i=1}^{n} \bigg( \bigvee_{j=1}^{m} L_{i,j} \bigg) \bigg) \]

- Non-deterministically pick a truth assignment. Represent this in a look-up table. [linear in number of literals]
- Check if truth assignment satisfies F. [quadratic – because of comparison of input with table entries]

- Formally, we need to do this on a TM – the encoding is a bit unwieldy, but straightforward.
1. SAT is in NP
2. SAT is NP-hard
Idea

- Give a logical formula which transforms computations of a TM $M$ with input string $u$ into a formula $f(u)$ s.t.

  $$u \text{ is accepted } \iff f(u) \text{ is satisfiable.}$$

- Show that transformation is polynomial.

- $[f(u)$ doesn’t have to be in CNF because of Exercise 30$]$
Encoding

ND TM $M$:
- states: $q_0, \ldots, q_m$
- alphabet: $B = a_0, \ldots, a_t$
- accepting state: $q_m$
- rejecting state: $q_{m-1}$ (only one)

$p(n)$ polynomial which is upper bound to number of computations

Boolean variables:
- $Q_{i,k}$ M is in state $q_i$ at time $k$
- $P_{j,k}$ Tape head is in position $j$ at time $k$
- $S_{j,r,k}$ Tape position $j$ contains symbol $a_r$ at time $k$
SAT is NP-hard: Clauses \textit{i} & \textit{ii}

<table>
<thead>
<tr>
<th>Clause</th>
<th>Conditions</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{i})</td>
<td><strong>State</strong></td>
<td>$0 \leq k \leq p(n)$</td>
</tr>
<tr>
<td></td>
<td>$\bigvee_{i=0}^{m} Q_{i,k}$</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>$\neg Q_{i,k} \lor \neg Q_{i',k}$</td>
<td>$0 \leq i &lt; i' \leq m$</td>
</tr>
<tr>
<td></td>
<td>$0 \leq k \leq p(n)$</td>
<td>$</td>
</tr>
<tr>
<td>\textit{ii})</td>
<td><strong>Tape head</strong></td>
<td>$0 \leq k \leq p(n)$</td>
</tr>
<tr>
<td></td>
<td>$\bigvee_{j=0}^{p(n)} P_{j,k}$</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>$\neg P_{j,k} \lor \neg P_{j',k}$</td>
<td>$0 \leq j &lt; j' \leq p(n)$</td>
</tr>
<tr>
<td></td>
<td>$0 \leq k \leq p(n)$</td>
<td>$</td>
</tr>
</tbody>
</table>
## SAT is NP-hard: Clause iii

<table>
<thead>
<tr>
<th>Clause</th>
<th>Conditions</th>
<th>Interpretation</th>
</tr>
</thead>
</table>
| iii) Symbols | $\forall_{r=0}^{t} S_{j,r,k}$  
$0 \leq j \leq p(n)$  
$0 \leq k \leq p(n)$ | For each time $k$ and position $j$, position $j$ contains at least one symbol  
[p($n$)$^2$ clauses, $t$ literals each] |
| $\neg S_{j,r,k} \lor \neg S_{j,r',k}$ | $0 \leq j \leq p(n)$  
$0 \leq r < r' \leq t$  
$0 \leq k \leq p(n)$ | … and at most one symbol  
[O($t^2$) $\times$ p($n$)$^2$ clauses] |
### Clause Interpretation

<table>
<thead>
<tr>
<th>Clause</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>iv) <strong>Initialization</strong></td>
<td></td>
</tr>
<tr>
<td>( Q_{0,0} )</td>
<td>Begin in state 0</td>
</tr>
<tr>
<td>( P_{0,0} )</td>
<td>…reading leftmost tape cell (position 0)</td>
</tr>
<tr>
<td>( S_{0,0,0} )</td>
<td>…which contains a blank (symbol 0)</td>
</tr>
<tr>
<td>( S_{1,r1,0} )</td>
<td>The next n symbols contain the input string, ( a_{r1} ), ( a_{r2} ), … ( a_{rn} )</td>
</tr>
<tr>
<td>( S_{2,r2,0} )</td>
<td>( S_{n,rn,0} )</td>
</tr>
<tr>
<td>( S_{n+1,0,0} )</td>
<td>And the rest of the tape contains blanks…</td>
</tr>
<tr>
<td>( S_{p(n),0,0} )</td>
<td>… for the entire accessible portion</td>
</tr>
<tr>
<td>v) <strong>Final state</strong></td>
<td></td>
</tr>
<tr>
<td>( Q_{m,p(n)} )</td>
<td>The computation ends in ( q_m ) – the accepting state</td>
</tr>
</tbody>
</table>
A computation that satisfies all of these clauses still doesn’t necessarily follow the rules of the machine, $M$.

Each state/symbol/position after time 0 must be obtained from the transition rules of $M$. 
### Tape Consistency

<table>
<thead>
<tr>
<th>Clause</th>
<th>Conditions</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi) Tape</td>
<td>0 ≤ j ≤ p(n)</td>
<td>Symbols not at the position of the tape head are unchanged</td>
</tr>
<tr>
<td>Changes</td>
<td>0 ≤ r ≤ t</td>
<td>[p(n)^2 × t clauses]</td>
</tr>
<tr>
<td>( \neg S_{j,r,k} \lor P_{j,k} \lor S_{j,r,k+1} )</td>
<td>0 ≤ k ≤ p(n)</td>
<td></td>
</tr>
</tbody>
</table>
Converting rules in $\delta$ to clauses

\[ \neg Q_{i,k} \lor \neg P_{j,k} \lor \neg S_{j,r,k} \lor Q_{i',k+1} \]

If none of these are satisfied, then we are in state $q_i$ and position $j$ scanning symbol $a_r$ at time $k$

In that case, the next state must be $Q_{i'}$ or the clause is not satisfied.

For each $\delta(q_i, a_r) = [q_i', ?, ?]$
Same thing for tape symbols

\[ \neg Q_{i,k} \lor \neg P_{j,k} \lor \neg S_{j,r,k} \lor S_{j,r',k+1} \]

If none of these are satisfied, then we are in state \( Q_i \) and position \( P_j \) scanning symbol \( S_r \) at time \( k \)

In that case, the next symbol at position \( j \) must be \( S_r \) or the clause is not satisfied.

For each \( \delta(q_i, a_r) = [?, a_r, ?] \)
If none of these are satisfied, then we are in state $Q_i$ and position $P_j$ scanning symbol $S_r$ at time $k$. In that case, the tape head will move either one position left or one position right.

Where $n(L) = -1$, and $n(R) = +1$

For each $\delta(q_i, a_r) = [?, ?, L/R]$
The conjunction of these three clause types ensures that if we are in a certain state, reading a particular symbol at a particular time, we must be in the right configuration, according to $\delta$ in the following time step.

These are machine dependent.
Consistency clauses are constructed for every time, state, tape head position and tape symbol.

However, if we are scanning position 0 and attempt to move left, we go directly to the rejecting state.
Hey, wait a minute!

We’ve been talking like there is only one transition for each state/symbol pair, but this is a non-deterministic Turing machine, right?

Let $\text{trans}(i, j, r, k)$ be the disjunction of all the consistency clause sets for $i, j, r, k$. The resulting clause ensures that we are in some valid configuration following each transition.
And now we’re done

<table>
<thead>
<tr>
<th>Clause</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi) Halted</td>
<td></td>
</tr>
</tbody>
</table>
\[-Q_{i,k} \lor \neg P_{j,k} \lor \neg S_{j,r,k} \lor Q_{i,k+1}\] same state

\[-Q_{i,k} \lor \neg P_{j,k} \lor \neg S_{j,r,k} \lor P_{j,k+1}\] same tape head position

\[-Q_{i,k} \lor \neg P_{j,k} \lor \neg S_{j,r,k} \lor S_{j,r,k+1}\] same symbol at position r

For all appropriate \(j, r, k, \) and \(i = q_{m-1}\),

and \(i = q_{m}\)
We’ve defined a set of wff that are satisfiable if (and only if) some computation of ND TM M leads to an accepting final state.
Polynomial transformation?

Can the formula be created from any NDTM M *in polynomial time*?

- The values $m$ and $t$ are independent of the size of the input string. They do not grow with $n$.
- The number of clauses is polynomial in $p(n)$.

qed