CIRCUIT-SAT is NP-hard

Lucia Moura

Winter 2010

Theorem

The circuit-satisfiability problem is NP-hard.

Proof. (Cormen, Leiserson and Rivest, Introduction to Algorithms.)
The proof uses Karp transformation. Let X be a problem in NP. We will describe a polynomial time algorithm F computing a transformation function f that maps every binary string x to a circuit $K = f(x)$ such that $x \in X$ if and only if $K \in$ CIRCUIT-SAT.

In the next slides we describe how to build $K = f(x)$, and argue that F runs in polynomial time and does the job of mapping "yes" instances of X to "yes" instances of CIRCUIT-SAT and "no" instances of X to "no" instances of CIRCUIT-SAT.

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:
  - the program for A,

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:
  - the program for A,
  - the program counter, auxiliary machine state,

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:
  - the program for A,
  - the program counter, auxiliary machine state,
  - the input x, the certificate y,

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:
  - the program for A,
  - the program counter, auxiliary machine state,
  - the input x, the certificate y,
  - working storage.

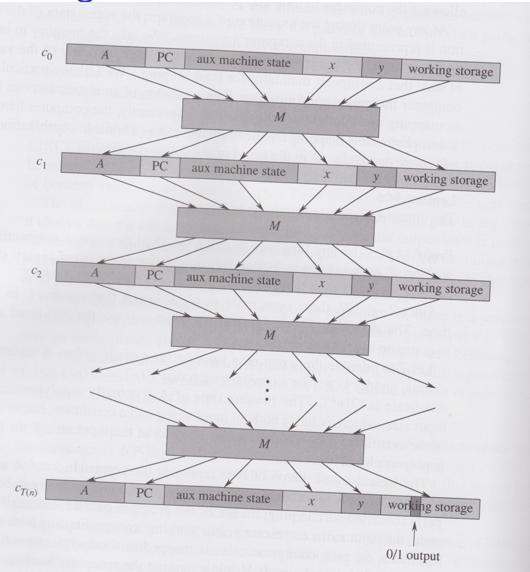
How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:
  - the program for A,
  - the program counter, auxiliary machine state,
  - the input x, the certificate y,
  - working storage.
- Starting with configuration c_o, configuration c_i is mapped to configuration c_{i+1} by a combinatorial circuit M implementing the computer hardware. The configurations reside as values on the wires connecting copies of M.

How to compute $K = f(x)$, for $x \in X$

- Since $X \in$ NP, we have a two input certifier $A(x, y)$ that runs in polynomial time. The algorithm F will use certifier A to build K.
- Let $T(n)$ be the worst-case running time of A on inputs of length $n = |x|$. Since $|y|$ has also a polynomial size $|x|$, we know that there is a k such that $T(n) \in O(n^k)$ and $|y| \in O(n^k)$.
- We represent K as a sequence of configurations as illustrated in the figure. Each configuration, includes:
  - the program for A,
  - the program counter, auxiliary machine state,
  - the input x, the certificate y,
  - working storage.
- Starting with configuration c_o, configuration c_i is mapped to configuration c_{i+1} by a combinatorial circuit M implementing the computer hardware. The configurations reside as values on the wires connecting copies of M.
- The output of algorithm A appears as one of the bits of $c_{T(n)}$.

Configurations and connections used to build K



How $K = f(x)$ is built

Build K as shown in the previous picture.
The following values in c_o must be wired to their known values:

- program A.
- initial counter,
- input x,
- initial state of the memory.

The only remaining **inputs** for the circuit K are the bits of y.
Also, all outputs (values in $c_{T(n)}$) are ignored, and the only **output** of K is the bit that represents $A(x, y)$.
Algorithm F then receives x and outputs K, the circuit described above.

Lemma

*Let $K = f(x)$ computed by Algorithm F. Then,
K is satisfiable if and only if $x \in X$.*

Proof:
($\Leftarrow$)
Suppose $x \in X$. Then, there exists a certificate y such that $A(x, y) = 1$. If we apply the bits of y to the inputs of K the output of the circuit will be $A(x, y) = 1$. So K is satisfiable.
($\Rightarrow$)
Suppose K is satisfiable. Then there exists an input y to K such that $K(y) = 1$. But by construction $K(y) = A(x, y)$ and so $A(x, y) = 1$, and $x \in X$.

Lemma

Algorithm F runs in polynomial time in $n = |x|$.

Proof:

- First we claim the number of bits to represent each configuration c_i is polynomial on n:
  First, the program for A has constant size (independent on $|x|$), $|x| = n$, $|y| \in O(n^k)$. Since A runs in $O(n^k)$ steps the amount of work storage is also polynomial on n.

Lemma

Algorithm F runs in polynomial time in $n = |x|$.

Proof:

- First we claim the number of bits to represent each configuration c_i is polynomial on n:
  First, the program for A has constant size (independent on $|x|$), $|x| = n$, $|y| \in O(n^k)$. Since A runs in $O(n^k)$ steps the amount of work storage is also polynomial on n.
- The combinatorial circuit M implementing the computer hardware has size polynomial in the length of a configuration (which is in $O(n^k)$), and hence is a polynomial in n.

Lemma

Algorithm F runs in polynomial time in $n = |x|$.

Proof:

- First we claim the number of bits to represent each configuration c_i is polynomial on n:
  First, the program for A has constant size (independent on $|x|$), $|x| = n$, $|y| \in O(n^k)$. Since A runs in $O(n^k)$ steps the amount of work storage is also polynomial on n.
- The combinatorial circuit M implementing the computer hardware has size polynomial in the length of a configuration (which is in $O(n^k)$), and hence is a polynomial in n.
- The circuit K contains $T(n) = O(n^k)$ copies of configurations and of M.

Lemma

Algorithm F runs in polynomial time in $n = |x|$.

Proof:

- First we claim the number of bits to represent each configuration c_i is polynomial on n:
  First, the program for A has constant size (independent on $|x|$), $|x| = n$, $|y| \in O(n^k)$. Since A runs in $O(n^k)$ steps the amount of work storage is also polynomial on n.
- The combinatorial circuit M implementing the computer hardware has size polynomial in the length of a configuration (which is in $O(n^k)$), and hence is a polynomial in n.
- The circuit K contains $T(n) = O(n^k)$ copies of configurations and of M.
- Each step of the algorithm F that builds K takes polynomial time.