Thu Oct 31 16:32:03 PDT 2019

Abstract

The Cook-Levin theorem, also known as Cook's theorem, states that Boolean satisfiability (SAT) is $\mathcal{NP}$-complete, meaning that any $\mathcal{NP}$ problem, *i.e.,* any problem solved in polynomial time on a non-deterministic machine, has a [deterministic] polynomial time reduction to SAT.

Boolean Satisfiability

There are many alternative ways to define a Boolean expression, but for our discussion, we must fix one of them. We define a string to be a *Boolean expression* if it is generated by the following context-free grammar G, with start symbol S: Let BOOL be the language of all strings generated by G.

$S \to\, !S$ (logical not)
$S \to S \Rightarrow S$ (implies)
$S \to S \equiv S$ (logical equal)
$S \to S \not\equiv S$ (logical not equal)
$S \to S * S$ (logical and)
$S \to S + S$ (logical or)
$S \to (S)$
$S \to 0$ (false)
$S \to 1$ (true)
$S \to I$
$I \to$ identifier

We assume an infinite set of strings we call *identifiers.* An *assignment* of a Boolean expression E is an assignment of each identifier in E to a logical value, either 0 (false) or 1 (true). We say that an assignment *satisfies* E the if evaluation of E yields 1, after repacing each identifier by its assigned value. Otherwise, E is not satisfiable, *i.e.,* a *contradiction.* Evaluation uses the rules of precedence of C++. An instance of the *Boolean satisfiability problem* is a Boolean expression $E \in$ BOOL, where $E \in$ SAT means E is satisfiable.

Non-Deterministic Turing Machines

Definition 1 *A language L is in the class $\mathcal{NP}$–*TIME *if there is some non-deterministic machine M which accepts L in polynomial time.*

Every $\mathcal{NP}$ language can be reduced to a $\mathcal{NP}$ language over any alphabet Σ of size two in polynomial time, by using a Huffman code. Since we use numerals in the reduction proof of Theorem 1 below, we will let $\Sigma = \{a, b\}$.

We assume that L is accepted in polynomial time by a non-deterministic machine M. By the Church-Turing thesis, we can assume that M is a non-deterministic Turing machine (NTM).

We give the definition of an NTM below. This definition differs somewhat from the definition given on page 273 of our textbook, the sixth edition of Linz. In particular, the tape is semi-infinite *i.e.,* has a left-most cell and extends infinitely to the right, as opposed to being infinite in both directions as in our text. The tape is a sequence of *cells* $\text{cell}[1], \text{cell}[2] \ldots$, where $\text{cell}[1]$ is the leftmost cell. Each cell can hold one *tape symbol.* A *read–write* head is positioned at one cell, the current cell. It can read and write the current cell, and can move left or right one cell, or stay at the same cell.

An NTM M is defined to be an ordered tuple $(Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ where Q is the finite set of internal states, Σ is the input alphabet, Γ is the tape alphabet, $q_0 \in Q$ is the start state,

$\Box \in \Gamma$ is the blank symbol, $F \subseteq Q$ is the set of final states, and δ is the transition function. We write $\Delta = \{L, N, R\}$, the set of movement directions. (R is for right, L is for left, and N is for stay.) For any $(q, x) \in Q \times \Gamma$, $\delta(q, x)$ is some subset of $Q \times \Gamma \times \Delta$.[1] Without loss of generality, there are never more than two choices at any step, the tape alphabet has only three symbols, and there is just one final state h which is a trapping state, and which we call the *halt state.* We summarize our requirements for an NTM below.

$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \Box\}$
$\delta(q, \gamma)$ has at most two members for any $(q, \gamma) \in Q \times \Gamma$
$\delta(h, \gamma) = \{(h, \gamma, N)\}$ for any $\gamma \in \Gamma$.
$F = \{h\}$
All but finitely many cells are blank (*i.e.,* contain $\Box$).

We say that M has *halted* if the state is h. Note that M simply marks time when its state is h. Once M has halted, it will remain in the halt state forever.

Instantaneous Descriptions

At each step of a computation, M has a current *instantaneous description*, or **id**, which is a string of the form uqv, where $u, v \in \Gamma^*$ and $q \in Q$. This **id** encodes the following configuration of M:

1. The internal state of M is q.
2. The tape contents consist of the string uv followed by infinitely many blanks, and the read/write head is positioned at cell$[k+1]$ where $k = |u|$. That is, the i^{th} symbol of u is written in cell$[i]$, and the i^{th} symbol of v is written in cell$[k+i]$.

Action of M

We describe one step of M. Let q_i be the current internal state. If $q_i = h$, the computation has halted, and the **id.** does not change. and let x be the symbol at the current cell. $(q_j, y, D) \in \delta(q_i, x)$ is arbitrarily chosen. If $\delta(q_i, x)$ is empty, M *hangs*, *i.e.,* stops abnormally. Hanging does not count as halting. The symbol y is written to the current cell, and the read-write head moves in direction $D \in \{L, N, R\}$. If the current cell is cell[1], the leftmost cell, and if the direction is L, then M hangs.

Definition 2 *We say that an NTM M* accepts *a string $w \in \Sigma^*$ if there is a computation of M which begins at the* **id** *$q_0 w$ and halts,* i.e., *ends in state h. We say that M* accepts *a language L if M accepts every $w \in L$ and does not accept any string which is not a member of L.*

Definition 3 *A language L is $\mathcal{NP}$*–COMPLETE *if there is a $\mathcal{P}$*–TIME *reduction of any given $\mathcal{NP}$*–TIME *language to L.*

Theorem 1 (Cook-Levin) SAT *is $\mathcal{NP}$*–COMPLETE*.*

Proof of Theorem 1

Let L be a language over $\Sigma = \{a, b\}$ which is accepted the NTM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F)$ in time T, where T is a polynomial function. Let N be the number of internal states of M. We can assume $T(n) \geq n$ for all n. We construct a reduction $R_M : \Sigma^* \to$ BOOL such that $w \in L \Leftrightarrow R_M(w) \in$ SAT.

[1]If M is deterministic, $\delta(q, \gamma)$ has at most one member.

Let $w \in \Sigma^*$, and let $n = |w|$, and let $T = T(n)$. $R_M(w)$ uses $(T+1)^2(1+|Q|)$ variables, and is the conjunction of $(T+1)^2|Q|$ clauses. $R_M(w)$ is satisfiable if and only if $w \in L$.

Write $E = R_M(w)$. Let $\mathcal{X}$ be the set of all computations of M with input w consisting of exactly T steps.

Overview. E contains MMMM variables, and is the conjunction of NNNN clauses. If $X \in \mathcal{X}$, each variable of E is a statement about X. Typically, a variable is true for some $X \in \mathcal{X}$ and false for others. After substituting for each variable its value given by X, each clause of E is true if and only if X halts within T steps. Thus, there is a halting computation of M with input w of length at most T if and only if there is one choice of X for which E is true, meaning that E is satisfiable.

Variables. We will introduce each variable using parameters. We could, with a little effort, choose identifiers for those variables which follow the rules for C++ identifiers, for instance. We leave that task as an exercise for the reader.

Since the head cannot move more than one cell at each step, after T steps it will be at cell$[i]$ for some $0 \le i \le T$; the remaining cells will remain blank during the computation.

We give an informal definition of each variable, in terms M and an arbitrary computation $X \in \mathcal{X}$. We say "time t" to mean after t steps of X.

1. For each $t \in [0, T]$ and $q \in Q$, let $sta(t, q)$ be the statement that the internal state of M is q at time t.
2. For each $t, i \in [0, T]$, let $head(t, i)$ be the statement that the head is at cell$[i]$ at time t.
3. For each $t, i \in \{0, T\}$ and $\gamma \in \Gamma$, let $symb[t, i, \gamma]$ be the statement that at time t the symbol at cell$[i]$ is γ.
4. For each $t \in [[]1, T]$ and each $D \in \{L, N, R\}$, let $move[t, D]$ be the statement that the head moves in direction D at the t^{th} step.

Thus, there are exactly $4T^2 - 11T + 4$ variables.

Clauses. Each clause is either a single variable, or a coumpound statement involving the variables, which states that M behaves properly. We give a brief explanation of each clause. For an assignment to satisfy E, all these clauses must be true. of each.

1. $head[0, 0]$, the head is at cell[0] initially.
2. $sta[0, q_0]$, the initial internal state is q_0.
3. $symb[0, i, w_i]$ for $i \le n$; cells $1 \ldots n$ initially contain the string w.
4. $symb[9, j, \Box]$ for all $n < j \le T$; the rest of the tape is initially blank.
5. $sta[T, h]$, the final internal state is the halt state.
6. $!sta[t, q_i] + !sta[t, q_j]$ for $t \in [[]9, T]$ and distinct states $q_i \neq q_j \in Q$; M cannot have two internal states at once.
7. $(!symb[t, i, a] + !symb[t, i, b]) * (!symb[t, i, a] + !symb[t, i, \Box]) * (!symb[t, i, b] + !symb[t, i, \Box])$; cell$[i]$ holds only one symbol at a time.
8. $!head[t, i] + !head[t, j]$ for $i \neq j$; the head cannot be in more than one place at a time.
9. $move[t, i, L] * head[t, i] \Rightarrow head[t+1, i-1]$
10. $move[t, i, N] * head[t, i] \Rightarrow head[t+1, i]$
11. $move[t, i, R] * head[t, i] \Rightarrow head[t+1, i+1]$

The last three clauses state that the change of the head position is consistent with the movement direction.

The above clauses are necessary, but not sufficient. Description of the remaining clauses is much more complex.

12. For each $q \in Q$ and $\gamma \in \Gamma$, and for each $1 \le t \le T$ and $0 \le i \le T$, We will define a clause $\mathcal{C}[q, \gamma, t, i]$, which is a statement that M behaves properly at the t^{th} step of the computation, if the head is at cell$[i]$, the current internal state is q, and the current symbol at cell$[i]$ is γ. Recall that $\delta(q, \gamma)$ is a set consisting of at most two triples.
    $\mathcal{C}[t, i, q, \gamma]$ is the statement

    $$\mathit{sta}[t-1, q] * \mathit{symb}[t-1, i, \gamma] \Rightarrow \mathcal{R}[q, \gamma, t, i]$$

    where $\mathcal{R}[q, \gamma, t, i]$ is described below.

    (a) If $\delta(q, \gamma) = \emptyset$, then $\mathcal{R}[q, \gamma, t, i] = 0$, meaning that the internal state q together with the tape symbol γ may not occur at any step of a halting computation.

    (b) If $\delta(q, \gamma) = \{(q', \gamma', D)\}$ then $\mathcal{R}[q, \gamma, t, i]$ is

    $$\mathit{sta}[t, i, q'] * \mathit{symb}[t, i, \gamma'] * \mathit{move}[t, i, D]$$

    (c) If $\delta(q, \gamma) = \{(q', \gamma', D'), (q'', \gamma'', D''\}$ then $\mathcal{R}[q, \gamma, t, i]$ is

    $$\mathit{sta}[t, i, q'] * \mathit{symb}[t, i, \gamma'] * \mathit{move}[t, i, D'] + \mathit{sta}[t, i, q''] * \mathit{symb}[t, i, \gamma''] * \mathit{move}[t, i, D'']$$

In summary, E is the conjunction of all the clauses listed above, and can be constructed in polynomial time. Each computation of M with input w assigns values to all the variables. That computation halts if and only if all clauses evaluate to true. Thus, there is a halting computation of M with input w if and only if there is a satisfying assignment of E. This completes the proof of the Cook-Levin theorem.