CS121 Section 8: Rice's Theorem, Reduction, P, EXP

Zuzanna Skoczylas
A function $F$ is called nontrivial iff there are two inputs $x$ and $y$ such that $F(x)=0$ and $F(y)=1$. 

$$F(x) = \begin{cases} 
1 & |x| > 10 \\
0 & o/w 
\end{cases}$$
Semantic property

A pair of Turing machines $M$ and $M'$ are *functionally equivalent* if for every $x \in \{0,1\}^*$, $M(x) = M'(x)$. (In particular, $M$ halts on $x$ iff $M'$ halts on $x$ for all $x$.) A function $F:\{0,1\}^* \rightarrow \{0,1\}$ is *semantic* if for every pair of strings $M, M'$ that represent functionally equivalent Turing machines, $F(M) = F(M')$. Example: $\text{ZEROFUNC}$

$$\text{ZEROFUNC}: \{0,1\}^* \rightarrow \{0,1\}$$

$1$ iff $M$ represents TM such that $M$ outputs $0$ for all $x \in \{0,1\}^*$

$$\text{ZEROFUNC}(M) =$ $\text{ZEROFUNC}(M')$$
Rice's Theorem

Let $F: \{0,1\}^* \rightarrow \{0,1\}$. If $F$ is semantic and non-trivial then it is uncomputable.
Rice's Theorem Example

Prove that CONST in uncomputable. CONST: \{0,1\}^* \rightarrow \{0,1\} is a function that on every input M representing Turning Machine returns 1 iff M computes a constant 0 or constant 1 function.

\[
M \, - \, TM \, that \, is \, constant \, 0 \, function
\]

\[
M' \, - \, TM \, that \, returns \, 1 \, iff \, \exists \, a \, path \, in \, the \, graph \, from \, s \, to \, t
\]

\[
\text{CONST (M)} = 1 \quad \text{CONST (M')} = 0
\]
Rice's Theorem Example

\[ Q, Q' \]

\[ \forall x \in \{0, 1\}^* \quad Q(x) = Q'(x) \]

\[ \text{CONST}(Q) = \text{CONST}(Q') \]
Reduction

Steps to prove F is uncomputable:

1. Assume for contradiction that F is computable, so there exists program P that computes it.
2. Create program compHALT or compHALTONZERO, that using P computes HALT or HALTONZERO. Here you usually have to modify the input to P.
3. This implies that HALT or HALTONZERO is computable, which is a contradiction proved in the book.
4. Therefore F cannot be computable.
Prove that ACCEPT is uncomputable. ACCEPT(P) returns 1 if P halts on any \{0,1\}^* string and 0 otherwise.

1. Assume for contradiction that ACCEPT is computable, so there exists program E that computes it.

2. Create the program compHALT, that using E computes HALT.

\[
\text{compHALT}(M,x) : \\
1. \text{Construct NAND-\text{TM} program } M' : \\
\text{runs } M \text{ on } x \\
\text{return 1 if } M' \text{ halted} \\
2. \text{return } E(M')
\]
Prove that\( \text{ACCEPT} \) is uncomputable. \( \text{ACCEPT}(P) \) returns 1 if \( P \) halts on any \( \{0,1\}^* \) string and 0 otherwise.

3. This implies that \( \text{HALT} \) is computable, which is a contradiction proved in the book.

4. Therefore \( \text{ACCEPT} \) cannot be computable.

\[
\begin{align*}
\varepsilon(M') &= 1 \implies \text{HALT}(M,x) = 1. \\
\varepsilon(M') &= 0 \implies \text{HALT}(M_1,x) = 0
\end{align*}
\]
A propositional formula $\varphi$ involves $n$ variables $x_1, \ldots, x_n$ and the logical operators AND ($\land$), OR ($\lor$), and NOT ($\neg$). We say that such a formula is in conjunctive normal form (CNF for short) if it is an AND of ORs of variables or their negations. For example, this is a CNF formula

$$ (x_7 \lor \neg x_{22} \lor x_{15}) \land (x_{37} \lor x_{22}) \land (x_{55} \lor \neg x_7) $$

The satisfiability problem is the task of determining, given a CNF formula $\varphi$, whether or not there exists a satisfying assignment for $\varphi$. A satisfying assignment for $\varphi$ is a string $x \in \{0,1\}^n$ such that $\varphi$ evaluates to True if we assign its variables the values of $x$. 
We say that a formula is a $k$-CNF if it is an AND of ORs where each OR involves exactly $k$ literals. The $k$-SAT problem is the restriction of the satisfiability problem for the case that the input formula is a $k$-CNF.

In particular, the 2SAT problem is to find out, given a 2-CNF formula $\phi$, whether there is an assignment $x \in \{0, 1\}^n$ that satisfies $\phi$, in the sense that it makes it evaluate to 1.

\[
(\chi_1 \lor \chi_2) \land (\chi_1) \iff (\chi_1 \lor \chi_2) \land (\chi_1 \lor \chi_1)
\]
2SAT in EXP

\[(\bar{x} \lor y) \land (\bar{y} \lor z) \land (x \lor \bar{z}) \land (z \lor y)\]
2SAT IN P

\[ (\overline{x} \lor y) \land (\overline{y} \lor z) \land (x \lor \overline{z}) \land (z \lor y) \lor (a \lor \overline{b}) \]

\[
\begin{align*}
\overline{y} \lor 2 \\
y \Rightarrow 2 \\
\overline{2} \Rightarrow \overline{y} \\
x \lor \overline{y} \\
x \Rightarrow \overline{y} \\
\overline{y} \Rightarrow \overline{x}
\end{align*}
\]
2SAT IN P

$x \Rightarrow \overline{x}$

$x = TRUE$

$x = FALSE$

$\overline{x} = \overline{y}$

$\overline{y} \Rightarrow y$

$y = TRUE$

$y = x = 2 = TRUE$

$O(n \cdot m)$
P is a subset of EXP!
Section problems

1. Function \( F : \{0,1\}^* \rightarrow \{0, 1\} \) checks whether the input encodes a TM that, on every input for which it halts, outputs either a string with at most \( n \) 0s or a string with length at least \( n \). Prove that \( F \) is uncomputable using Rice's theorem or state why Rice's theorem does not apply and show polynomial time algorithm.

2. Prove that if \( F, G : \{0,1\}^* \rightarrow \{0, 1\} \) are in P then their composition \( F \circ G \), which is the function \( H \) s.t. \( H(x) = F(G(x)) \), is also in P.

3. Prove or disprove: \( F \) is uncomputable. Let \( F \) be the following function. On input a (string representing a) pair \((M, P)\) where \( M \) is a Turing Machine and \( P \) is a NAND-TM program, \( F \) outputs 1 if and only if \( M \) and \( P \) are functionally equivalent, in the sense that for every \( x \in \{0,1\}^* \), either both \( M \) and \( P \) don’t halt on \( x \), or \( M(x) = P(x) \).