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

Zuzanna Skoczylas

Non-trivial property

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 $F:\{0,1\}^* \to \{0,1\}$ is *semantic* if for every pair of strings *M*,*M'* that represent functionally equivalent Turing machines, $F(M)=F(M')$. Example: *ZEROFUNC*

ZEROFUNC: $\{0,1\}^* \to \{0,1\}$ $\quad \forall M \in \{0,1\}^*$

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

ZEROFUNC(M) = ZEROFUNC(M')

Rice's Theorem

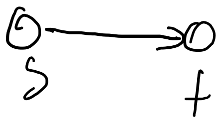
Let F: $\{0,1\}^* \rightarrow \{0,1\}$. If *F* is semantic and non-trivial then it is uncomputable.

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 ~~re~~returns 1 iff $\exists$ a path in the graph from s to t



$CONST(M) = 1$ $\quad$ $CONST(M') = 0$

Rice's Theorem Example

$Q\ ,\ Q'$

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

$CONST(Q) = CONST(Q')$

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.

Reduction Example

Prove that ACCEPT in 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.

compHALT(M, x):

1. Construct NAND-TM program M':
   runs M on x
   returns 1 if M halted
2. return E(M')

Reduction Example

Prove that ACCEPT in uncomputable. ACCEPT(P) returns 1 if P halts on any $\{0,1\}^*$ string and 0 otherwise.

3. This implies that HALT computable, which is a contradiction proved in the book.
4. Therefore ACCEPT cannot be computable.

$E(M') = 1. \Rightarrow HALT(M,x) = 1.$

$E(M') = 0 \Rightarrow HALT(M,x) = 0$

SAT

A *propositional formula φ* involves *n* variables $x_1,\dots, x_n$ and the logical operators AND (∧), OR (∨), and NOT (¬). 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

$$(x_7 \vee \neg x_{22} \vee x_{15}) \overset{\text{AND}}{\wedge} (x_{37} \vee x_{22}) \overset{\text{AND}}{\wedge} (\underbrace{x_{55} \vee \neg x_7}_{\text{or}})$$

The *satisfiability problem* is the task of determining, given a CNF formula *φ*, whether or not there exists a *satisfying assignment* for *φ*. A satisfying assignment for *φ* is a string $x \in \{0,1\}^n$ such that *φ* evaluates to *True* if we assign its variables the values of *x*.

2SAT

We say that a formula is a *k*-CNF 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 φ, whether there is an assignment $x \in \{0,1\}^n$ that *satisfies* φ, in the sense that it makes it evaluate to 1.

$(x_1 \vee x_2) \wedge (x_1) \iff (x_1 \vee x_2) \wedge (x_1 \vee x_1)$

2SAT in EXP

$$(\bar{x} \vee y) \wedge (\bar{y} \vee z) \wedge (x \vee \bar{z}) \wedge (z \vee y)$$

x, y, z

0 1   0 1   0 1

2^3

2^n

2SAT IN P

$(\bar{x} \vee y) \wedge (\bar{y} \vee z) \wedge (x \vee \bar{z}) \wedge (z \vee y)$ ✓ ✓ ✓

$\bar{y} \vee z$

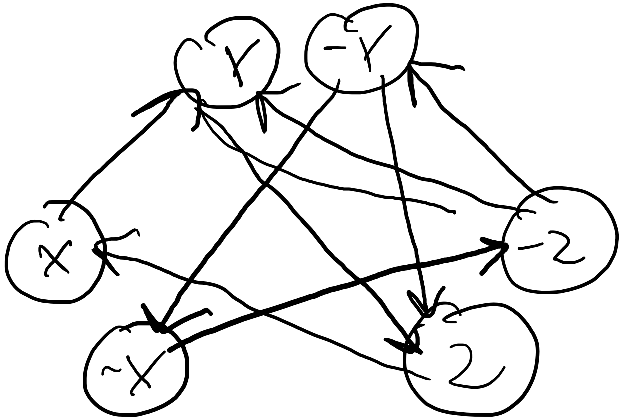
$y \Rightarrow z$

$\bar{z} \Rightarrow \bar{y}$

$\bar{x} \vee y$

$x \Rightarrow y$

$\bar{y} \Rightarrow \bar{x}$



$(a \vee b)$

$\bar{a} \Rightarrow b$

$\bar{b} \Rightarrow a$

$z \vee y$

$\bar{z} \Rightarrow y$

$\bar{y} \Rightarrow z$

$x \vee \bar{z}$

$\bar{x} \Rightarrow \bar{z}$

$z \Rightarrow x$

2SAT IN P

$x \Rightarrow \overline{x}$

$x = \text{FALSE}$

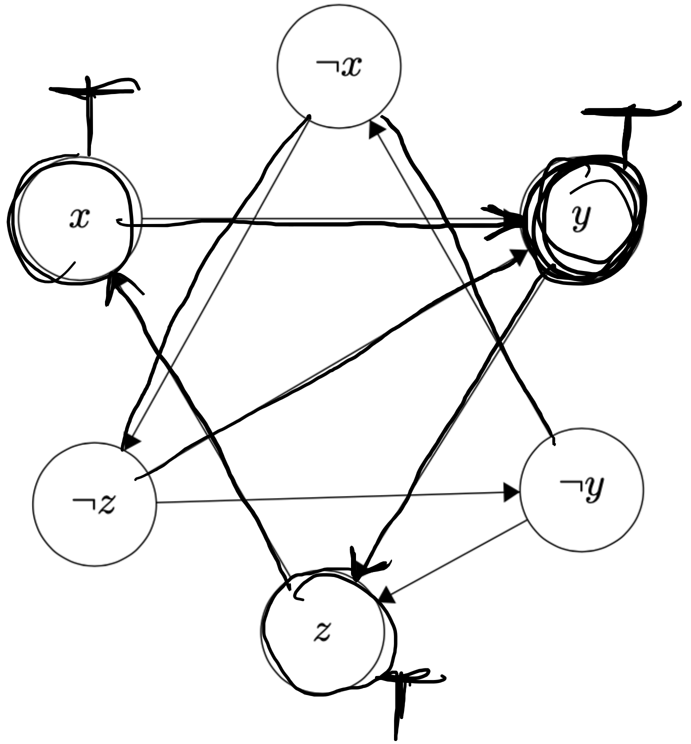
$\overline{x} \Rightarrow x$

$x = \text{TRUE}$

$n \quad 2n$

$m \quad 2m$

$O(n(m+n))$

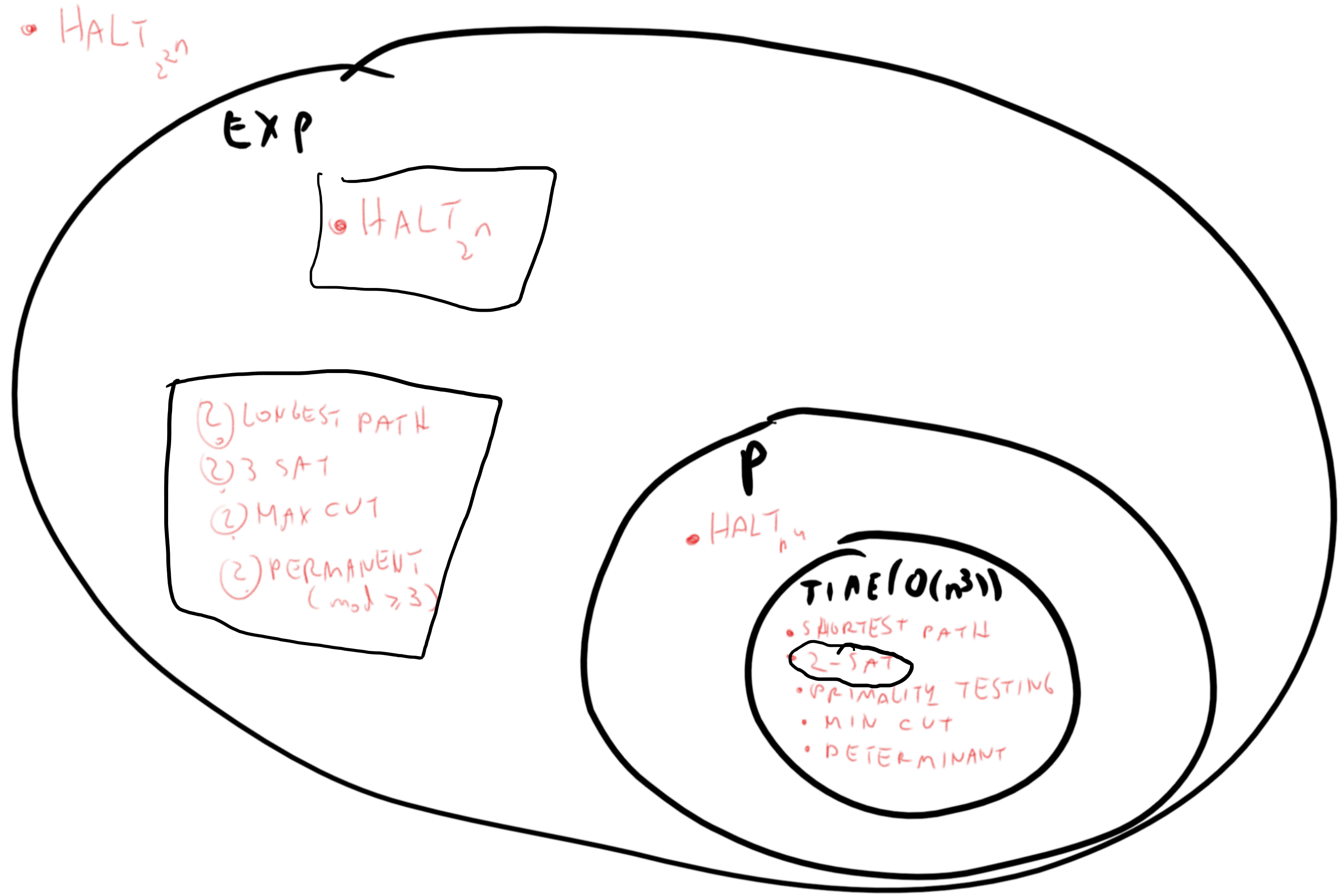


$\overline{y} \Rightarrow y$

$y = \text{TRUE}$

$y = x = z = \text{true}$

P is a subset of EXP!



Section problems

1. Function $F : \{0,1\}^* \to \{0, 1\}$ 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 $F, G : \{0,1\}^* \to \{0, 1\}$ 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 $M(x) = P(x)$.