Vertex Cover

A vertex cover (v.c.) C of an undirected graph G is a set of vertices such that each edge of G has (at least) one endpoint in C.

$VC = \{\langle G \rangle \,\$\, \langle k \rangle \mid G \text{ has a v.c. of size } k\}$

Thm. 1 VC is p.-t. red. to Ex-3-SAT.

Proof. Let $E = C_1 \wedge \ldots \wedge C_n$ be a B.E. in exact 3NF. Construct a graph G s.t. E is satisfiable iff G has a v.c. of size $2n$.

For each clause $C_r = (\ell_1^r \vee \ell_2^r \vee \ell_3^r)$, create three vertices v_1^r, v_2^r and v_3^r in G. Connect v_i^r and v_j^s by an edge iff

1. $r = s$ and $i \neq j$ OR

2. ℓ_i^r is the negation of ℓ_j^s.

The reduction can be computed in p.t. $\square$

The correctness of the construction:

Observation: Any v.c. of some constructed graph has at least $2n$ vertices, since two out of the three vertices forming a "triangle" according to condition 1 must be in any cover.

Let α be a satisfying assignment of E. Each clause must contain some literal ℓ_i^r evaluating to 1. Pick the corresponding vertex v_i^r. The vertices *not* picked this way form a v.c. of size $2n$ (which is minimal according to the observation).

Now assume that C is a v.c. in G of size $2n$. Consider some "triangle" (i.e., some clause). According to the observation, one vertex does *not* belong to the cover. Assign 1 to the literal corresponding to this vertex. This yields no contradiction by the second condition and gives a (partial) assignment which can be extended arbitrarily to give a satisfying assignment of E.

C, VC, and IS

The reduction proofs for C and VC were *very* similar. This is no coincidence.

A set I of vertices of a given graph G is called an *independent set* if no edge of G joins any two vertices in I.

$IS = \{\langle G \rangle \,\$\, \langle k \rangle \mid G$ has an independent set of size $k\}$

$G^c = (V, E^c)$ is the *(loopless) graph complement* of the undirected graph $G = (V, E)$ if $E \cap E^c = \emptyset$ and

$$\forall v, v' \in V : v \neq v' \Rightarrow \{v, v'\} \in E \cup E^c$$

Thm. 2 An n-vertex graph G has a v.c. of size k iff
G has an independent set of size $n - k$ iff
G^c has a clique of size $n - k$.

Cor. 3 C, VC, and IS are p.-t. reducible to each other.

Hardness and Completeness

Let $\mathcal{C}$ be a language class.

A language L is *hard for* $\mathcal{C}$ (with respect to p.-t. reductions) if every language in $\mathcal{C}$ is p.-t. reducible to L.

A language L is *complete for* $\mathcal{C}$ (with respect to p.-t. reductions) L is hard for $\mathcal{C}$ and $L \in \mathcal{C}$.

Important special case: $\mathcal{NP}$-completeness.

The n-step Accepting Problem I

$\mathsf{ACC}_{\leq} = \{\langle T \rangle \, \$ \, \langle w \rangle \, \$^n \mid \text{NTM } T \text{ accepts } w \text{ in at most } n \text{ steps } \}$

Thm. 4 $\mathsf{ACC}_{\leq}$ is $\mathcal{NP}$-hard.

Proof. If $L \in \mathcal{NP}$, then there is an NTM T deciding $w \in L$ in time $p(|w|)$. So, $w \in L$ iff $\langle T \rangle \, \$ \, \langle w \rangle \, \$^{p(|w|)} \in \mathsf{ACC}_{\leq}$. $\square$

The n-step Accepting Problem II

Thm. 5 $\mathrm{ACC}_{\leq}$ is $\mathcal{NP}$-complete.

Proof. Hardness due to last theorem.
$\mathrm{ACC}_{\leq} \in \mathcal{NP}$, because dovetailing a *nondeterministic* universal TM (working on input $\langle T \rangle \,\$\, \langle w \rangle$) and a step counter (counting up to n) gives an NTM deciding $\mathrm{ACC}_{\leq}$ in p.-t.
□

Remark: The NTM for $\mathrm{ACC}_{\leq}$ seemingly intrinsically makes nondeterministic guesses along its whole computation.

"Guess and Check" Revisited

Our "standard" way designing p.-t. *nondeterministic* algorithm was to implement two phases:

1. Guess a so-called *certificate* of polynomial length, this way making a polynomial number of nondeterministic steps.

2. *Verify* the certificate by a deterministic p.-t. algorithm.

According to Kinber/Smith definition, a language L is in $\mathcal{NP}$ iff there is a "Guess and Check" p.-t. algorithm for deciding L.

We will exhibit a "Guess and Check" p.-t. algorithm for deciding $\mathrm{ACC}_{\leq}$ on the next slide. Hence, we can conclude:

Thm. 6 $L \in \mathcal{NP}$ (according to "our" definition) iff L lies in the class $\mathcal{NP}$ as defined by Kinber/Smith.

"Guess and Check" for $\mathsf{ACC}_{\leq}$

A certificate of $x = \langle T \rangle \, \$ \, \langle w \rangle \, \$^n \in \mathsf{ACC}_{\leq}$ is a (codified) sequence

$$(q_1, w_1), (q_2, w_2), \ldots, (q_m, w_m)$$

of configurations with $m \leq n$. Such a sequence is of length polynomial in $|x|$.

In the "checking phase", the algorithm tests the following:

- Is $w = w_1$?
- Is q_m accepting?
- For each $1 \leq k < m$, test if $(q_k, w_k) \vdash_T (q_{k+1}, w_{k+1})$ (by using a universal TM in a deterministic fashion)?

The certificate is verified if each of this questions is answered with YES.

ACC Variants I

$\text{ACC}_{=} = \{\langle T \rangle \, \$ \, \langle w \rangle \, \$^n \mid \text{NTM } T \text{ accepts } w \text{ in exactly } n \text{ steps } \}$

Thm. 7 $\text{ACC}_{=}$ is $\mathcal{NP}$-complete.

Proof. Membership in $\mathcal{NP}$ similar as before.

We reduce $\text{ACC}_{\leq}$ to $\text{ACC}_{=}$:

Given T, modify it by adding rules $((q, a), (q, a))$ for all states q and all tape symbols a in order to obtain TM T'. This allows T' to "wait" if T happens to accept "too fast". Hence,

$$\langle T \rangle \, \$ \, \langle w \rangle \, \$^n \in \text{ACC}_{\leq} \iff \left\langle T' \right\rangle \, \$ \, \langle w \rangle \, \$^n \in \text{ACC}_{=}$$

□

ACC Variants II

$\text{ACC}'_{=} = \{\langle T\rangle \,\$\, \langle w\rangle \mid \text{NTM } T \text{ accepts } w \text{ in exactly } |w| \text{ steps } \}$

Thm. 8 $\text{ACC}'_{=}$ is $\mathcal{NP}$-complete.

Proof. Membership in $\mathcal{NP}$ similar as before.

We reduce $\text{ACC}_{=}$ to $\text{ACC}'_{=}$:

If $|w| \geq n$, translate the $\text{ACC}_{=}$-instance $\langle T\rangle \,\$\, \langle w\rangle \,\$^n$ into an equivalent $\text{ACC}'_{=}$-instance $\langle T\rangle \,\$\, \langle w\rangle$ by adding wait loops as before.

Otherwise, proceed as follows:
Given w, define $w' = w\sqcup^{|w|-n}$. Hence,

$$\langle T\rangle \,\$\, \langle w\rangle \,\$^n \in \text{ACC}_{=} \iff \langle T\rangle \,\$\, \left\langle w'\right\rangle \in \text{ACC}'_{=}$$

□

Rem.: works also for $w = e$!

Computation Carpet Certificates

Convention: We now assume (w.l.o.g.) that the tape alphabet and the state set of a Turing machine are disjoint. A configuration $(q, \alpha\underline{a}\omega)$ can then be written as $\alpha q a \omega$.

Given an instance $\langle T \rangle \, \$ \, \langle w \rangle$ of $\mathsf{ACC}'_{=}$, with $\triangleright sw = a_{1,1} \ldots a_{1,n+2}$, a certificate (for some Guess and Check algorithm) can be viewed as a 2-dimensional "carpet" (of size polynomial in $|w|$ and $\langle T \rangle$:

$$CC = \begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1,n+2} \\ a_{21} & a_{22} & \cdots & a_{2,n+2} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n+1,1} & a_{n+1,2} & \cdots & a_{n+1,n+2} \end{array}$$

Verifying Computation Carpet Certificates

A computation carpet certificate CC will be verified if

1. $\triangleright sw = a_{1,1} \ldots a_{1,n+2}$,

2. $\triangleright h1\sqcup^{n-1} = a_{n+1,1} \ldots a_{n+1,n+2}$, and

3. each jth row in the carpet is a successor configuration of the $j-1$th row ($1 < j \leq n+1$).

We are going to employ computation carpets in order to prove

Cook's Theorem

Thm. 9 SAT is $\mathcal{NP}$-complete.

Proof. We already know: SAT $\in \mathcal{NP}$.

On the following slides, we show how to describe computation carpets by a Boolean expression (having a size polynomially related to the size of the carpet).

This way, we will reduce $\mathrm{ACC}'_{=}$ to SAT.

To an instance $\langle T \rangle \,\$\, \langle w \rangle$ of $\mathrm{ACC}'_{=}$, we associate an expression

$$E(T,w) = E_{S_1} \wedge E_{S_2} \wedge E_{S_3}(w) \wedge E_{S_4} \wedge E_T$$

which is satisfiable iff T accepts w in exactly $|w|$ steps. □

Describing Carpet Patterns

We introduce the variables $C_{i,j,X}$ which are "true" iff there is an X at position (i, j) of a computation carpet.

There are some immediate "syntactical conditions" to be posed:

- At each position, a symbol must be specified:

$$E_{S_1} = \bigwedge_{i,j} \bigvee_{X} C_{i,j,X}$$

- At each position, not more than one symbol may be specified:

$$E_{S_2} = \bigwedge_{i,j} \bigwedge_{X} \bigwedge_{Y, X \neq Y} \overline{C_{i,j,X}} \vee \overline{C_{i,j,Y}}$$

Describing The Start And The End

- The initial configuration, given by the start state s and the input word $w = a_1 \ldots a_n$, is correctly described:

$$E_{S_3}(w) = C_{1,1,\triangleright} \wedge C_{1,2,s} \wedge \bigwedge_j C_{1,j+2,a_j}$$

- The final configuration is correctly described:

$$E_{S_4} = C_{n+1,1,\triangleright} \wedge C_{n+1,2,h} \wedge C_{n+1,3,1} \wedge \bigwedge_{4 \leq j \leq n+1} C_{n+1,j,\sqcup}$$

Describing Transitions

Depending on T, we can check whether or not a symbol U assumed to be at position (i, j) and a symbol V at position $(i, j + 1)$ (one of them holding a state) can possibly be there by looking at the symbols W, X, Y, Z at positions $(i-1, j-1)$, $(i-1, j)$, $(i-1, j+1)$ and $(i - 1, j + 2)$, respectively. This describes a predicate $P_T(U, V, W, X, Y, Z)$. All other symbols stay the same from row $i - 1$ to row i.

$E_T = \bigwedge_{i>1} \bigvee_j (\bigvee_{U,V,W,X,Y,Z:P_T(U,V,W,X,Y,Z)} C_{i-1,j,U} \wedge C_{i-1,j+1,V} \wedge C_{i,j-1,W} \wedge C_{i,j,X} \wedge C_{i,j+1,Y} \wedge C_{i,j+2,Z}) \wedge \bigwedge_{k,k\neq j} \bigwedge_X C_{i,k,X} \Leftrightarrow C_{i-1,k,X}$

Consequences From Cook's Theorem

Thm. 10 The following problems are $\mathcal{NP}$-complete:

- Clique,
- Vertex Cover,
- Independent Set,
- Lit-SAT,
- CSAT,
- 3-SAT,
- EX-3-SAT

Three problems we considered have yet no $\mathcal{NP}$-completeness proof:

The directed Hamilt. Cycle Problem

HCD $= \{\langle G \rangle \mid G$ has a Hamitonian cycle $\}$

Given a *directed* graph G, does G have a directed tour (which is called *Hamiltonian cycle* in this case)?

Thm. 11 Ex-3-SAT is p.-t. red. to HCD.

Cor. 12 The following problems are $\mathcal{NP}$-complete: HCD, HC, and TS.

Ladders. . .

Consider a directed graph $L(n)$ with $2n + 4$ vertices:

$$a, d, b_0, \ldots, b_n, c_0, \ldots, c_n$$

and edges:

(a, b_0), (a, c_0), (b_n, d), (c_n, d),
(b_i, c_i) for $i = 0, \ldots, n$
(c_i, b_i) for $i = 0, \ldots, n$
(b_i, c_{i+1}) for $i = 0, \ldots, n-1$
(c_i, b_{i+1}) for $i = 0, \ldots, n-1$

Lem. 13 A ladder $L(n)$ has two paths from a to d visiting each vertex exactly once:

$$a, b_0, c_0, b_1, c_1, \ldots, b_n, c_n, d$$

$$a, c_0, b_0, c_1, b_1, \ldots, c_n, b_n, d$$

In the reduction, ladders will be used to set variables consistently, with level i connected to clause i.

. . . and Snakes

Consider a directed graph S with 6 vertices:

$$u_0, u_1, u_2, v_0, v_1, v_2$$

and edges

$(u_i, u_{(i+1) \bmod 3})$ for $i = 0, 1, 2$
$(v_i, v_{(i-1) \bmod 3})$ for $i = 0, 1, 2$
(u_i, v_i) for $i = 0, 1, 2$

Lem. 14 Any path P in S which

- visits each vertex at most once,
- starts with a u-vertex and ends with a v-vertex, and
- leaves a graph $S - P$ which is similarly traversible by one or two other paths from u- to v-vertices

will "enter" the graph at u_i and "leave" it at the corresponding v_i.

In the reduction, snakes will be used to simulate clauses.

The Reduction for HCD

Let $E = C_1 \wedge \ldots \wedge C_n$ be an instance of Ex-3-SAT, with each clause $C_r = (\ell_0^r \vee \ell_1^r \vee \ell_2^r)$.

Let $x_1, \ldots, x_m$ be the variables of E.

For each variable x_i, we create a copy L^i of a ladder $L(3n)$ with vertices

$$a^i, d^i, b_0^i, \ldots, b_{3n}^i, c_0^i, \ldots, c_{3n}^i.$$

The ladders are cyclically interconnected: add edges (d^i, a^{i+1}), $i = 1, \ldots, n-1$, and (d^m, a^1).

For each clause, we create a copy S^r of snake S with vertices $u_0^r, u_1^r, u_2^r, v_0^r, v_1^r, v_2^r$, where $\{u_j^r, v_j^r\}$ "simulate" ℓ_j^r.

Furthermore, introduce:

- edges (b_r^i, u_k^r) and (v_k^r, c_r^{i+1}) if $\ell_k^r = x_i$, i.e., x_i occurs positively in C_r; and

- edges (c_r^i, u_k^r) and (v_k^r, b_r^{i+1}) if $\ell_k^r = \overline{x_i}$, i.e., x_i occurs negatively in C_r.

Why does the HCD reduction work?

Let α be a satisfying assignment of the B.E. E. Then, the constructed HCD instance has a Hamiltonian cycle: superficially, this cycle can be described as going through the ladders L^1 through L^n, on its way traversing the snakes.

If α assigns 1 to x_i, then the traversal of L^i starts with $a^i, b^i_0, c^i_0, \ldots$.

If α assigns 0 to x_i, then the traversal of L^i starts with $a^i, c^i_0, b^i_0, \ldots$.

Since α is satisfying, there is a witness function

$$\beta : \{1, \ldots, n\} \to \{0, 1, 2\}$$

such that $\alpha(\ell^r_{\beta(r)}) = 1$.

For each clause C_r, we can implement the following "detour" into the cycle sketched so far:

- If $x_i = \ell^r_{\beta(r)}$, then trace L^i from $b^i_r, u^r_{\beta(r)}$, throughout the whole snake S^r and leave it via $v^r_{\beta(r)}, c^{i+1}_r$.

- If $\overline{x_i} = \ell^r_{\beta(r)}$, then trace L^i from $c^i_r, u^r_{\beta(r)}$, throughout the whole snake S^r and leave it via $v^r_{\beta(r)}, b^{i+1}_r$.

The reverse direction is left to the students.