Advanced Topics in Theoretical Computer Science

Part 5: Complexity (Part 2)

31.01.2013

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Until now

- Big O notation
- Types of complexity: space complexity and time complexity
  - DTIME and NTIME
  - DSPACE and NSPACE
- P, NP, PSPACE, Relationships
- Complete problems; hard problems; reductions
- Closure of complexity classes
- Examples

Until now: DTIME and NTIME

Basic model: k-DTM or k-NTM M (one tape for the input)

If M makes for every input word of length n at most $T(n)$ steps, then M is $T(n)$-time bounded.

In this case, the language accepted by M has time complexity $T(n)$; (more precisely $\max(n+1, T(n))$.

Definition ($NTIME(T(n)), DTIME(T(n))$)

- $DTIME(T(n))$ class of all languages accepted by $T(n)$-time bounded DTMs.
- $NTIME(T(n))$ class of all languages accepted by $T(n)$-time bounded NTMs.

Until now: DSPACE and NSPACE

Basic model: k-DTM or k-NTM M with special tape for the input (is read-only) + k storage tapes (offline DTM) $\mapsto$ needed if $S(n)$ sublinear

If M needs, for every input word of length n, at most $S(n)$ cells on the storage tapes then M is $S(n)$-space bounded.

The language accepted by M has space complexity $S(n)$;
(more precisely $\max(1, S(n))$).

Definition ($NSPACE(T(n)), DSPACE(T(n))$)

- $DSPACE(S(n))$ class of all languages accepted by $S(n)$-space bounded DTMs.
- $NSPACE(S(n))$ class of all languages accepted by $S(n)$-space bounded NTMs.

Until now: Facts

NTM vs. DTM:

- $DTIME(f(n)) \subseteq NTIME(f(n))$
- $DSPACE(f(n)) \subseteq NSPACE(f(n))$
- $NTIME(f(n)) \subseteq DTIME(2^{h(n)})$ where $h \in O(f)$.
  Idea: If we try to simulate an NTM with a DTM we may need exponentially more time.
- $NSPACE(f(n)) \subseteq DSPACE(f^2(n))$
  This is **Savitch's theorem**, proved by Walter Savitch in 1970.
  (The proof is a bit involved and is not given in this lecture.)

Time vs. Space:

- $DTIME(f(n)) \subseteq DSPACE(f(n))$
- $NTIME(f(n)) \subseteq NSPACE(f(n))$

($DSPACE(f(n))$, $NSPACE(f(n))$ are much larger)

Until now: Facts

Theorem.
For every $c \in \mathbb{R}^+$ and every storage function $S(n)$ the following hold:

- $DSPACE(S(n)) = DSPACE(cS(n))$
- $NSPACE(S(n)) = NSPACE(cS(n))$

Theorem For every $c \in \mathbb{R}^+$ and every time function $T(n)$ with $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$ the following hold:

- $DTIME(T(n)) = DTIME(cT(n))$
- $NTIME(T(n)) = NTIME(cT(n))$

Theorem: Let T be a time function with $\lim_{n\to\infty} \frac{T(n)}{n} = \infty$ and S a storage function.

(a) If $f(n) \in O(T(n))$ then $DTIME(f(n)) \subseteq DTIME(T(n))$.

(b) If $g(n) \in O(S(n))$ then $DSPACE(g(n)) \subseteq DSPACE(S(n))$.

Until now: P, NP, PSPACE

Definition

$$\begin{aligned} P &= \bigcup_{i \geq 1} DTIME(n^i) \\ NP &= \bigcup_{i \geq 1} NTIME(n^i) \\ PSPACE &= \bigcup_{i \geq 1} DSPACE(n^i) \end{aligned}$$

Lemma $NP \subseteq \bigcup_{i \geq 1} DTIME(2^{O(n^d)})$

$P \subseteq NP \subseteq PSPACE$

How do we show that a certain problem is in a certain complexity class?

Reduction to a known problem

Until now: Reduction

Definition (Polynomial time reducibility)

Let L_1, L_2 be languages.
L_2 is polynomial time reducible to L_1 (notation: $L_2 \preceq_{\text{pol}} L_1$)
if there exists a polynomial time bounded DTM, which for every input w computes an output $f(w)$ such that

$$w \in L_2 \text{ if and only if } f(w) \in L_1$$

Lemma (Polynomial time reduction)

- Let L_2 be polynomial time reducible to L_1 ($L_2 \preceq_{\text{pol}} L_1$). Then:
  - If $L_1 \in NP$ then $L_2 \in NP$.
  - If $L_1 \in P$ then $L_2 \in P$.
- The composition of two polynomial time reductions is again a polynomial time reduction.

Until now: NP

Theorem (Characterisation of NP)

A language L is in NP if and only if there exists a language L' in P and a $k \geq 0$ such that for all $w \in \Sigma^*$:

$$w \in L \text{ iff} \quad \text{there exists } c : \langle w, c \rangle \in L' \text{ and } |c| < |w|^k$$

c is also called witness or certificate for w in L.
A DTM which accepts the language L' is called verifier.

Important

A decision procedure is in NP iff every "Yes" instance has a short witness

(i.e. its length is polynomial in the length of the input)

which can be verified in polynomial time.

Until now: Complete and hard problems

Definition (NP-complete, NP-hard)
A language L is NP-hard (NP-difficult) if every language L' in NP is reducible in polynomial time to L.

A language L is NP-complete if:
- $L \in NP$
- L is NP-hard

Definition (PSPACE-complete, PSPACE-hard)
A language L is PSPACE-hard (PSPACE-difficult) if every language L' in PSPACE is reducible in polynomial time to L.

A language L is PSPACE-complete if:
- $L \in PSPACE$
- L is PSPACE-hard

Until now: Complete and hard problems

Remarks:

- If we can prove that at least one NP-hard problem is in P then P = NP
- If P $\neq$ NP then no NP complete problem can be solved in polynomial time

Open problem: Is P = NP? (Millenium Problem)

Until now: Complete and hard problems

How to show that a language L is NP-complete?

1. Prove that $L \in NP$
2. Find a language L' known to be NP-complete and reduce it to L

Is this sufficient?

Yes.

If $L' \in NP$ then every language in NP is reducible to L' and therefore also to L.

Often used: the SAT problem (Proved to be NP-complete by S. Cook)

$$L' = L_{\text{sat}} = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$$

Until now: Cook's theorem

Theorem $SAT = \{w \mid w \text{ is a satisfiable formula of propositional logic}\}$ is NP-complete.

Today

- Big O notation
- Types of complexity: space complexity and time complexity
  - DTIME and NTIME
  - DSPACE and NSPACE
- P, NP, PSPACE, Relationships
- Complete problems; hard problems; reductions
- Closure of complexity classes
- Examples

Closure of complexity classes

P, PSPACE are closed under complement

All complexity classes which are defined in terms of deterministic Turing machines are closed under complement.

Proof: If a language L is in such a class then also its complement is

(run the machine for L and revert the output)

Closure of complexity classes

Is NP closed under complement?

Closure of complexity classes

Is NP closed under complement?

Nobody knows!

Closure of complexity classes

Is NP closed under complement?

Nobody knows!

Definition

co-NP is the class of all laguages for which the complement is in NP

$$\text{co-NP} = \{L \mid \overline{L} \in NP\}$$

Relationships between complexity classes

It is not yet known whether the following relationships hold:

$P \stackrel{?}{=} NP$

$NP \stackrel{?}{=} \text{co-NP}$

$P \stackrel{?}{=} PSPACE$

$NP \stackrel{?}{=} PSPACE$

Today

- Big O notation
- Types of complexity: space complexity and time complexity
  - DTIME and NTIME
  - DSPACE and NSPACE
- P, NP, PSPACE, Relationships
- Complete problems; hard problems; reductions
- Closure of complexity classes
- Examples

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Multiprocessor scheduling

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Has a set of integers a subset with sum x?
7. Multiprocessor scheduling

Examples of NP-complete problems

Definition (SAT, k-CNF, k-DNF)

DNF: A formula is in DNF if it has the form

$$(L_1^1 \wedge \cdots \wedge L_{n_1}^1) \vee \cdots \vee (L_1^m \wedge \cdots \wedge L_{n_m}^m)$$

CNF: A formula is in CNF if it has the form

$$(L_1^1 \vee \cdots \vee L_{n_1}^1) \wedge \cdots \wedge (L_1^m \vee \cdots \vee L_{n_m}^m)$$

Examples of NP-complete problems

Definition (DNF, CNF, k-CNF, k-DNF)

DNF: A formula is in DNF if it has the form

$$(L_1^1 \wedge \cdots \wedge L_{n_1}^1) \vee \cdots \vee (L_1^m \wedge \cdots \wedge L_{n_m}^m)$$

CNF: A formula is in CNF if it has the form

$$(L_1^1 \vee \cdots \vee L_{n_1}^1) \wedge \cdots \wedge (L_1^m \vee \cdots \vee L_{n_m}^m)$$

k-DNF: A formula is in k-DNF if it is in DNF and all its conjunctions have k literals

k-CNF: A formula is in k-CNF if it is in CNF and all its disjunctions have k literals

Examples of NP-complete problems

SAT $= \{w \mid w$ is a satisfiable formula of propositional logic$\}$

CNF-SAT $= \{w \mid w$ is a satisfiable formula of propositional logic in CNF$\}$

k-CNF-SAT $= \{w \mid w$ is a satisfiable formula of propositional logic in k-CNF$\}$

Examples of NP-complete problems

Theorem

The following problems are in NP and are NP-complete:

(1) SAT

(2) CNF-SAT

(3) k-CNF-SAT for $k \geq 3$

Examples of NP-complete problems

Theorem

The following problems are in NP and are NP-complete:

(1) SAT

(2) CNF-SAT

(3) k-CNF-SAT for $k \geq 3$

Proof: (1) SAT is NP-complete by Cook's theorem.

CNF and k-CNF are clearly in NP.

(3) We show that 3-CNF is NP-hard. For this, we construct a polynomial reduction of SAT to 3-CNF.

Examples of NP-complete problems

Proof: (ctd.) Polynomial reduction of SAT to 3-CNF.

Let F be a propositional formula of length n

Step 1 Move negation inwards (compute the negation normal form) $\mapsto O(n)$

Step 2 Fully bracket the formula $\mapsto O(n)$

$P \wedge Q \wedge R \mapsto (P \wedge Q) \wedge R$

Step 3 Starting from inside out replace subformula $Q\text{op}R$ with a new propositional variable $P_{Q\text{op}R}$ and add the formula

$P_{Q\text{op}R} \rightarrow (Q\text{op}R)$ and $(Q\text{op}R) \rightarrow P_{Q\text{op}R}$ $\mapsto O(p(n))$

Step 4 Write all formulae above as clauses $\mapsto$ Rename(F) $\mapsto O(n)$

Let $f : \Sigma^* \rightarrow \Sigma^*$ be defined by:

$f(F) = P_F \wedge \text{Rename}(F)$ if F is a well-formed formula and $f(w) = \perp$ otherwise. Then:

$F \in$ SAT iff F is a satisfiable formula in prop. logic iff $P_F \wedge \mathit{Rename}(F)$ is satisfiable iff $f(F) \in$ 3-CNF-SAT

Example

Let F be the following formula:

$$[(Q \wedge \neg P \wedge \neg(\neg(\neg Q \vee \neg R))) \vee (Q \wedge \neg P \wedge \neg(Q \wedge \neg P))] \wedge (P \vee R).$$

Step 1: After moving negations inwards we obtain the formula:

$$F_1 = [(Q \wedge \neg P \wedge (\neg Q \vee \neg R)) \vee (Q \wedge \neg P \wedge (\neg Q \vee P))] \wedge (P \vee R)$$

Step 2: After fully bracketing the formula we obtain:

$$F_2 = [((Q \wedge \neg P) \wedge (\neg Q \vee \neg R)) \vee (Q \wedge (\neg Q \vee P) \wedge \neg P)] \wedge (P \vee R)$$

Step 3: Replace subformulae with new propositional variables (starting inside).

$$[\underbrace{\underbrace{\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2}}_{P_6} \vee \underbrace{\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4}}_{P_7}}_{P_8}] \wedge \underbrace{(P \vee R)}_{P_5}.$$

P_F

Example

Step 3: Replace subformulae with new propositional variables (starting inside).

$$\overbrace{\overbrace{\overbrace{[\overbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2})}^{P_6} \vee \overbrace{(\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4})}^{P_7}]}^{P_8} \wedge \underbrace{(P \vee R)}_{P_5}}^{P_F}.$$

F is satisfiable iff the following formula is satisfiable:

$$\begin{array}{llllll} P_F & \wedge & (P_F \leftrightarrow (P_8 \wedge P_5)) & \wedge & (P_1 \leftrightarrow (Q \wedge \neg P)) \\ & \wedge & (P_8 \leftrightarrow (P_6 \vee P_7)) & \wedge & (P_2 \leftrightarrow (\neg Q \vee \neg R)) \\ & \wedge & (P_6 \leftrightarrow (P_1 \wedge P_2)) & \wedge & (P_4 \leftrightarrow (\neg Q \vee P)) \\ & \wedge & (P_7 \leftrightarrow (P_1 \wedge P_4)) & \wedge & (P_5 \leftrightarrow (P \vee R)) \end{array}$$

can further exploit polarity

Example

Step 3: Replace subformulae with new propositional variables (starting inside).

$$\underbrace{\Big[\underbrace{\underbrace{\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee \neg R)}_{P_2}}_{P_6} \vee \underbrace{\underbrace{(Q \wedge \neg P)}_{P_1} \wedge \underbrace{(\neg Q \vee P)}_{P_4}}_{P_7}\Big]}_{P_8} \wedge \underbrace{(P \vee R)}_{P_5}}_{P_F}.$$

F is satisfiable iff the following formula is satisfiable:

$$\begin{array}{llllll} P_F & \wedge & (P_F \rightarrow (P_8 \wedge P_5)) & \wedge & (P_1 \rightarrow (Q \wedge \neg P)) \\ & \wedge & (P_8 \rightarrow (P_6 \vee P_7)) & \wedge & (P_2 \rightarrow (\neg Q \vee \neg R)) \\ & \wedge & (P_6 \rightarrow (P_1 \wedge P_2)) & \wedge & (P_4 \rightarrow (\neg Q \vee P)) \\ & \wedge & (P_7 \rightarrow (P_1 \wedge P_4)) & \wedge & (P_5 \rightarrow (P \vee R)) \end{array}$$

Example

F is satisfiable iff the following formula is satisfiable:

$$
\begin{array}{llll}
P_F & \wedge & (P_F \rightarrow (P_8 \wedge P_5)) & \wedge \quad (P_1 \rightarrow (Q \wedge \neg P)) \\
 & \wedge & (P_8 \rightarrow (P_6 \vee P_7)) & \wedge \quad (P_2 \rightarrow (\neg Q \vee \neg R)) \\
 & \wedge & (P_6 \rightarrow (P_1 \wedge P_2)) & \wedge \quad (P_4 \rightarrow (\neg Q \vee P)) \\
 & \wedge & (P_7 \rightarrow (P_1 \wedge P_4)) & \wedge \quad (P_5 \rightarrow (P \vee R))
\end{array}
$$

Step 4: Compute the CNF (at most 3 literals per clause)

$$
\begin{array}{llll}
P_F & \wedge & (\neg P_F \vee P_8) \wedge (\neg P_F \vee P_5) & \wedge \quad (\neg P_1 \vee Q) \wedge (\neg P_1 \vee \neg P) \\
 & \wedge & (\neg P_8 \vee P_6 \vee P_7) & \wedge \quad (\neg P_2 \vee \neg Q \vee \neg R) \\
 & \wedge & (\neg P_6 \vee P_1) \wedge (\neg P_6 \vee P_2) & \wedge \quad (\neg P_4 \vee \neg Q \vee P) \\
 & \wedge & (\neg P_7 \vee P_1) \wedge (\neg P_7 \vee P_4) & \wedge \quad (\neg P_5 \vee P \vee R)
\end{array}
$$

Examples of NP-complete problems

Proof: (ctd.) It immediately follows that CNF and k-CNF are NP-complete

Polynomial reduction from 3-CNF-SAT to CNF:

$f(F) = F$ for every formula in 3-CNF-SAT and $\perp$ otherwise.

$F \in$ 3-CNF-SAT iff $f(F) = F \in$ CNF-SAT.

Polynomial reduction from 3-CNF-SAT to k-CNF, $k > 3$

For every formula in 3-CNF-SAT:

$f(F) = F'$ (where F' is obtained from F by replacing a literal L with $\underbrace{L \vee \cdots \vee L}_{k-2 \text{ times}}$).

$f(w) = \perp$ otherwise.

$F \in$ 3-CNF-SAT iff $f(F) = F \in k$-CNF-SAT.

Examples of problems in P

Theorem

The following problems are in P:

(1) DNF

(2) k-DNF for all k

(3) 2-CNF

(1) Let $F = (L_1^1 \wedge \cdots \wedge L_{n_1}^1) \vee \cdots \vee (L_1^m \wedge \cdots \wedge L_{n_m}^m)$ be a formula in DNF.

F is satisfiable iff for some i: $(L_1^i \wedge \cdots \wedge L_{n_1}^i)$ is satisfiable. A conjunction of literals is satisfiable iff it does not contain complementary literals.

(2) follows from (1)

(3) Finite set of 2-CNF formulae over a finite set of propositional variables. Resolution $\mapsto$ at most quadratically many inferences needed.

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Multiprocessor scheduling

Examples of NP-complete problems

Definition

A clique in a graph G is a complete subgraph of G.

Clique $= \{(G, k) \mid G$ is an undirected graph which has a clique of size $k\}$

Examples of NP-complete problems

Theorem Clique is NP-complete.

Proof: (1) We show that Clique is in NP:

We can construct for instance an NTM which accepts Clique.

- M builds a set V' of nodes (subset of the nodes of G) by choosing k nodes of G (we say that M "guesses" V').
- M checks for all nodes in V' if there are nodes to all other nodes. (this can be done in polynomial time)

Examples of NP-complete problems

Theorem Clique is NP-complete.

Proof: (2) We show that Clique is NP-hard by showing that 3-CNF-SAT $\prec_{\text{pol}}$ Clique.

Let $\mathcal{G}$ be the set of all undirected graphs. We want to construct a map f (DTM computable in polynomial time) which associates with every formula F a pair $(G_F, k_F) \in \mathcal{G} \times \mathbb{N}$ such that

$$F \in \text{3-CNF-SAT} \quad \text{iff} \quad G_F \text{ has a clique of size } k_F.$$

$$F \in \text{3-CNF} \Rightarrow F = (L_1^1 \vee L_2^1 \vee L_3^1) \wedge \cdots \wedge (L_1^m \vee L_2^m \vee L_3^m)$$

F satisfiable iff there exists an assignment $\mathcal{A}$ such that in every clause in F at least one literal is true and it is impossible that P and $\neg P$ are true at the same time.

Examples of NP-complete problems

Theorem Clique is NP-complete.

Proof: (ctd.) Let $k_F := m$ (the number of clauses). We construct G_F as follows:

- **Vertices:** all literals in F.
- **Edges:** We have an edge between two literals if they (i) can become true in the same assignment and (ii) belong to different clauses.

Then:
(1) $f(F)$ is computable in polynomial time.
(2) The following are equivalent:

(a) G_F has a clique of size k_F.

(b) There exists a set of nodes $\{L^1_{i_1}, \ldots, L^m_{i_m}\}$ in G_F which does not contain complementary literals.

(c) There exists an assignment which makes F true.

(d) F is satisfiable.

Examples of NP-complete problems

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Multiprocessor scheduling

Examples of NP-complete problems

Definition (Rucksack problem)

A rucksack problem consists of:

- n objects with weights $a_1, \ldots, a_n$
- a maximum weight b

The rucksack problem is solvable if there exists a subset of the given objects with total weight b.

$$\text{Rucksack} = \{(b, a_1, \ldots, a_n) \in \mathbb{N}^{n+1} \mid \exists I \subseteq \{1, \ldots, n\} \; s.t. \; \sum_{i \in I} a_i = b\}$$

Examples of NP-complete problems

Theorem Rucksack is NP-complete.

Proof: (1) Rucksack is in NP: We guess I and check whether $\sum_{i \in I} a_i = b$

(2) Rucksack is NP-hard: We show that 3-CNF-SAT $\prec_{\text{pol}}$ Rucksack.

Construct f : 3-CNF $\to \mathbb{N}^*$ as follows.

Consider a 3-CNF formula $F = (L_1^1 \vee L_2^1 \vee L_3^1) \wedge \cdots \wedge (L_1^m \vee L_2^m \vee L_3^m)$

$f(F) = (b, a_1, \ldots, a_n)$ where:

(i) a_i encodes which atom occurs in which clause as follows:
p_i positive occurrences; n_i negative occurrences (numbers with $n + m$ positions)

- first m digits of p_i: p_{i_j} how often i-th atom of j-th clause occurs positively
- first m digits of n_i: n_{i_j} how often i-th atom of j-th clause occurs negatively
- last n digits of p_i, n_i: p_{i_j}, n_{i_j} which atom is referred by p_i
  p_i, n_i contain 1 at position $m + i$ and 0 otherwise.

Example

Let the set Prop of propositional variables consist of $\{x_1.x_2.x_3\}$.

$F: \quad (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_2 \vee \neg x_5) \wedge (\neg x_3 \vee \neg x_1 \vee x_4)$

$p_1 = 10010000$ $\qquad$ $n_1 = 00110000$

$p_2 = 02001000$ $\qquad$ $n_2 = 10001000$

$p_3 = 00000100$ $\qquad$ $n_3 = 00100100$

$p_4 = 10100010$ $\qquad$ $n_4 = 00000010$

$p_5 = 00000001$ $\qquad$ $n_5 = 01000001$

Satisfying assignment: $\mathcal{A}(x_1) = \mathcal{A}(x_2) = \mathcal{A}(x_5)$ and $\mathcal{A}(x_3) = \mathcal{A}(x_4) = 0$.

$$p_1 + p_2 + p_5 + n_3 + n_4 = \underbrace{121}_{\substack{\text{all digits} \leq 3 \\ \text{because 3 lit./clause}}} \quad \underbrace{11111}_{\substack{\text{all 1} \\ \text{all atoms considered}}}$$

Examples of NP-complete problems

Proof: (ctd.) If we have a satisfying assignment $\mathcal{A}$, we take for every propositional variable x_i mapped to 0 the number n_i and for every propositional variable x_i mapped to 1 the number p_i.

The sum of these numbers is $b_1 \dots b_m \underbrace{1 \dots 1}_{n \text{ times}}$ with $b_i \leq 3$,

so $b_1 \dots b_m \underbrace{1 \dots 1}_{n} < \underbrace{4 \dots 4}_{m} \underbrace{1 \dots 1}_{n}$

Let $b := \underbrace{4 \dots 4}_{m} \underbrace{1 \dots 1}_{n}$. We choose $\{a_1, \dots, a_k\} = \{p_1, \dots, p_n\} \cup \{n_1, \dots, n_n\} \cup C$.

The role of the numbers in $C = \{c_1, \dots, c_m, d_1, \dots, d_m\}$ is to make the sum of the a_is equal to b: $c_{i_j} = 1$ iff $i = j$; $d_{i_j} = 2$ iff $i = j$ (they are zero otherwise).

$f(F) \in$ Rucksack iff a subset I of $\{a_1, \dots, a_k\}$ adds up to b

iff a subset I of $\{p_1, \dots, p_n\} \cup \{n_1, \dots, n_n\}$ adds up to $b_1 \dots b_m 1 \dots 1$

iff for a subset I of $\{p_1, \dots, p_n\} \cup \{n_1, \dots, n_n\}$ there exists an assignment

iff $\mathcal{A}$ with $\mathcal{A}(P_i) = 1$(*resp.* 0) iff p_i(*resp.* n_i) occurs in I iff F satisfiable

Summary

Examples of NP-complete problems:

1. Is a logical formula satisfiable? (SAT)
2. Does a graph contain a clique of size k?
3. Rucksack problem
4. Is a (un)directed graph hamiltonian?
5. Can a graph be colored with three colors?
6. Multiprocessor scheduling