Complexity

Serge Haddad

LSV, ENS Cachan & CNRS & INRIA, Université Paris-Saclay

L3, ENS Cachan

Plan

Problems and Instances

- A *problem* P is a mapping from A to B.
- An *instance* is an item of A.
- When $B = \{\mathbf{true}, \mathbf{false}\}$, P is a *decision problem*.

Example: a graph $G = (V, E)$ and two vertices $s, t \in V$.

- What is the length of a shortest path from s to t (possibly ∞)?
- Does there exist a path from s to t?

Here we focus on decision problems.

Data Representation

A problem also defines the representation of instances and results.

Usually reasonnable representations should lead to same complexity measures.

Let $G = (V, E)$ be a graph with $n = |V|$ and $m = |E|$.
Two possible representations:

- first the number n and then the pairs of vertices in E.
  A size $\approx (2m + 1) \log(n + 1)$ bits;
- first the number n and then the adjacency matrix.
  A size $\approx n^2 + \log(n + 1)$ bits.

The *special* case of integers:

- n in binary with size $\approx \log(n + 1)$;
- n in unary with size $\approx n$.

Unary representation is used to determine whether the complexity of the problem comes from the numbers.

Complexity Measures

Time and **Space**.

Should be insensitive to scale change (so $O(), \Omega(), \Theta()$).

For space complexity, the size of the instance is not taken into account.

The choice of the computing device is relevant (contrary to the Church's thesis).

Example: A search in a sorted array of size n.

- in $O(log(n))$ with direct access to memory;
- in $O(n)$ with sequential access (like Turing machine);

Establishing Complexity Bounds

Upper Bounds: Provide an *algorithm* and analyze its complexity.

Lower Bounds: *Reduce* an already studied problem to the current problem.

Bootstrapping in complexity: How to establish a lower bound for the first problem?

Two Kinds of Turing Machines

Deterministic Turing machines, DTM (as seen in Computability lectures)

- An input tape (i.e. no write);
- Some working tapes;
- For general problems, a *stopping* state and an output tape (i.e. no read);
- For decision problems, two *absorbing* states: *accept* and *reject*;
- Time complexity is the length of the run and space complexity is the maximal position of a head of the working tapes along the run.

Non deterministic Turing machines, NTM (only for decision problems)

- A non deterministic transition function $\delta : Q \times \Sigma \to 2^{Q \times \Sigma \times Move}$;
- Several runs for a word;
- An *existential* semantic: the machine accepts if there is an accepting run;
- Time and space complexities are maximum values over the runs.

Measure Functions

f a non decreasing function from $\mathbb{N}$ to $\mathbb{N}$ is a *measure* function if:

- There is a Turing machine computing $f(n)$,
- operating in time $O(f(n)+n)$,
- and space $O(f(n))$.

A problem P belongs to:

- TIME(f) (resp. NTIME(f)) if there exists n_0 and a DTM (resp. NTM) deciding P and operating in time at most $f(n)$ for all instance of size $n \geq n_0$.
- SPACE(f) (resp. NSPACE(f)) if there exists n_0 and a DTM (resp. NTM) deciding P and operating in time at most $f(n)$ for all instance of size $n \geq n_0$.

So it is a *worst case* and *asymptotic* complexity.

Some Inclusions (1)

Determinism versus Non Determinism:

- $\mathrm{TIME}(f) \subseteq \mathrm{NTIME}(f)$
- $\mathrm{SPACE}(f) \subseteq \mathrm{NSPACE}(f)$

Time versus Space:

- $\mathrm{TIME}(f) \subseteq \mathrm{SPACE}(f)$
- $\mathrm{NTIME}(f) \subseteq \mathrm{NSPACE}(f)$

Accelerations: For all $k > 0$,

- $\mathrm{SPACE}(kf) \subseteq \mathrm{SPACE}(f)$
- $\mathrm{NSPACE}(kf) \subseteq \mathrm{NSPACE}(f)$

In exercises, similar results for time.

Some Inclusions (2)

A first simulation: $\mathsf{NTIME}(f) \subseteq \mathsf{SPACE}(f)$

- Let n be the size of the instance, compute $f(n)$ (in space $O(f(n))$;
- Check the acceptance of all the possible runs using an array of $f(n)$ choices (in space $O(f(n))$ but in time $O(k^{f(n)})$ for some k).

A second simulation: $\mathsf{NSPACE}(f) \subseteq \bigcup_{i \in \mathbb{N}} \mathsf{TIME}(i^{f+\log})$

Given an instance w of size n and an NTM $\mathcal{M}$,
the *configuration graph* $G_{\mathcal{M},w}$ is defined by:

- Vertices are configurations with working tapes of size $f(n)$;
- There is an edge from $\mathbf{c}$ to $\mathbf{c}'$ if there is a step of $\mathcal{M}$ from $\mathbf{c}$ to $\mathbf{c}'$.

The deterministic machine:

- computes the configuration graph in time $O(i^{f+\log})$ for some i;
- checks the reachability of an accepting configuration from the initial configuration in linear time w.r.t. the size of the graph.

Standard Classes

Let n be the size of the instance.

LOGSPACE $\equiv$ SPACE$(\log(n))$

NLOGSPACE $\equiv$ NSPACE$(\log(n))$

P $\equiv$ PTIME $\equiv \bigcup_{k\in\mathbb{N}}$ TIME(n^k)

NP $\equiv$ NPTIME $\equiv \bigcup_{k\in\mathbb{N}}$ NTIME(n^k)

PSPACE $\equiv \bigcup_{k\in\mathbb{N}}$ SPACE(n^k)

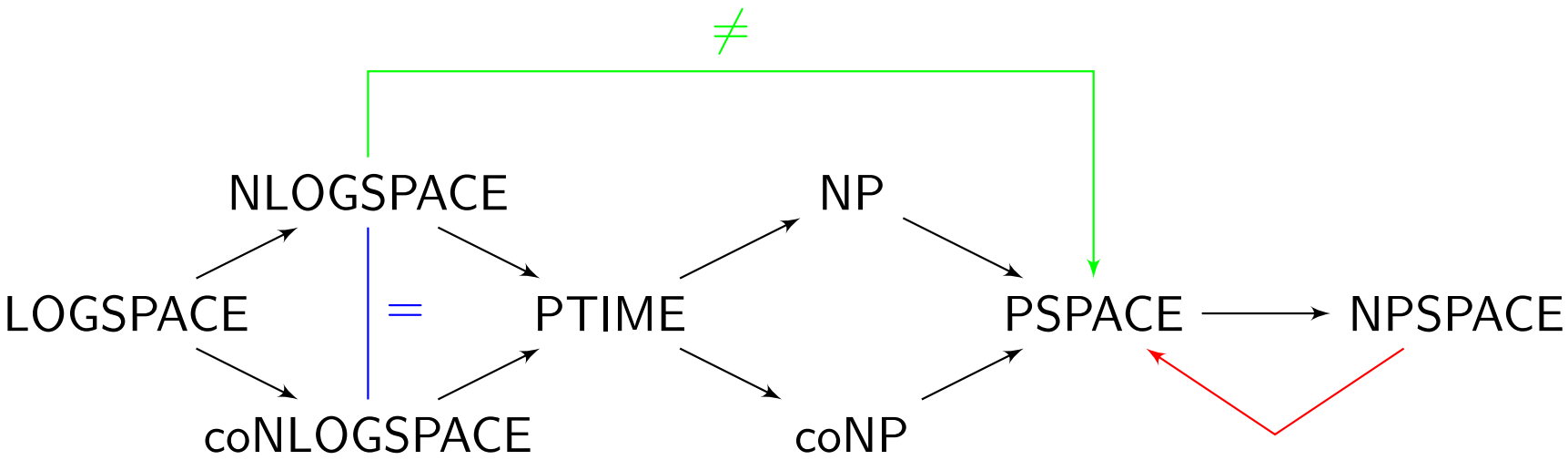
NPSPACE $\equiv \bigcup_{k\in\mathbb{N}}$ NSPACE(n^k)

and beyond (see the M1 course on Advanced Complexity) ...

Let P be a decision problem. Then $\neg P$ is the *complementary problem*.

Given a class C, coC is defined by coC $\equiv \{\neg P \mid P \in \mathsf{C}\}$.

Relation between Classes



Savitch theorem

Immerman-Szelepscényi theorem

A strict inclusion theorem

Reductions

Let P, P' be decision problems ranging over A (resp. A').

A PTIME (resp. LOGSPACE) reduction r is a mapping from A to A' such that:

- For all $a \in P$, $P'(r(a)) = P(a)$;
- There exists an algorithm implementing r and operating in PTIME (resp. LOGSPACE).

Composition of reductions: Let P'' ranging over A''.

- If r (resp. r') is PTIME reduction from P (resp. P') to P' (resp. P'') then $r' \circ r$ is a PTIME reduction from P to P'';
  Execute the second algorithm on the output of the first algorithm.
- If r (resp. r') is LOGSPACE reduction from P (resp. P') to P' (resp. P'') then $r' \circ r$ is a LOGSPACE reduction from P to P'';
  See the exercices.

Hard and Complete Problems

Let C be a complexity class. Then:

- P is a C-*hard problem* if for all $P' \in$ C there is a D-reduction with D $\prec$ C. *($\prec$ means that* D *is supposed to be strictly included in* C*)*
- P is a C-*complete problem* if P belongs to C and is C-hard.

Example: Use PTIME-reduction for NP and LOGSPACE-reduction for PTIME.

How to establish C-hardness for P?

- Either reduce P', a C-complete problem, to P.
- Or pick an accepting TM $\mathcal{M}$ belonging to C and exhibit an algorithm that:
  1. depends on $\mathcal{M}$ and C.
  2. takes a word w as input.
  3. outputs $a_w \in P$ such that $P(a_w)$ iff $\mathcal{M}$ accepts w.
  4. operates in the appropriate class D.

Plan

Propositional Logic

A *formula* of propositional logic φ is inductively defined by:

- φ may be **true**, **false** or an atomic proposition in a countable set $Prop$;
- $\varphi = \neg\varphi_1$, $\varphi = \varphi_1 \vee \varphi_2$, $\varphi = \varphi_1 \wedge \varphi_2$ where φ_1, φ_2 are formulas.

An *interpretation* ν is a mapping from $Prop$ to $\{\mathbf{true}, \mathbf{false}\}$.

Let ν be an interpretation. Then ν may be extended to formulas as follows.

- $\nu(\neg\varphi_1) = \overline{\neg}(\nu(\varphi_1))$;
- $\nu(\varphi_1 \vee \varphi_2) = \overline{\vee}(\nu(\varphi_1), \nu(\varphi_2))$;
- $\nu(\varphi_1 \wedge \varphi_2) = \overline{\wedge}(\nu(\varphi_1), \nu(\varphi_2))$.

Here $\overline{\neg}$ (resp. $\overline{\vee}$, $\overline{\wedge}$) is the function from $\{\mathbf{true}, \mathbf{false}\}$ (resp. $\{\mathbf{true}, \mathbf{false}\}^2$) to $\{\mathbf{true}, \mathbf{false}\}$ corresponding to the operator $\neg$ (resp. $\vee$, $\wedge$).

Other operators are abbreviations:

- $\varphi \Rightarrow \psi \equiv (\neg\varphi) \vee \psi$;
- $\varphi \Leftrightarrow \psi \equiv (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$;
- etc.

The Satisfiability Problem

Let φ be a formula and ν be an interpretation.
If $\nu(\varphi) = \mathbf{true}$, one says that ν *satisfies* φ, denoted by $\nu \models \varphi$.

> Given a formula φ as input, the *satisfiability problem* asks whether there exists an interpretation ν such that $\nu \models \varphi$.

The satisfiability problem belongs to NP:

- Let $p_1, \ldots, p_n$ be the propositions occurring in φ, guess in linear time $\nu(p_1), \ldots, \nu(p_n)$;
- Evaluate in linear time $\nu(\varphi)$ by a recursive function implementing the definition of $\nu(\varphi)$.

Satisfiability is NP-hard (1)

Let $\mathcal{M}$ be a NTM operating in polynomial time $p(n)$ where n is the size of the input w.

We code a configuration at time $j \leq p(n)$ by the following propositions:

- For all $q \in Q$, q^j is **true** if the state is q;
- For all $0 \leq i \leq p(n)$, i^j is **true** if the position of the head is i;
- For all $0 \leq i \leq p(n), a \in \Sigma$, a_i^j is **true** if the i^{th} cell contains a.

We denote $\mathbf{s}^j$ the set of these propositions.

Given a configuration $\mathbf{c}$ of $\mathcal{M}$ operating on w, $\nu_{\mathbf{c}}^j$ is the interpretation of $\mathbf{s}^j$ corresponding to $\mathbf{c}$.

The formula $\varphi_{\mathcal{M},w}$ is a conjunction of subformulas among them for all $j \leq p(n)$:

- $\bigvee_{q \in Q} q^j$ and for all $q \neq q' \in Q$, $\neg q^j \vee \neg (q')^j$;
- $\bigvee_{i \leq p(n)} i^j$ and for all $i < i' \leq p(n)$, $\neg i^j \vee \neg (i')^j$;
- for all $i \leq p(n)$, $\bigvee_{a \in \Sigma} a_i^j$ and for all $a \neq a' \in \Sigma$, $\neg a_i^j \vee \neg (a')_i^j$.

These subformulas ensure that, given an interpretation $\nu \models \varphi_{\mathcal{M},w}$,
for all j, there is a single configuration $\mathbf{c}_\nu^j$ such that $\nu(\mathbf{s}^j) = \nu_{\mathbf{c}_\nu^j}(\mathbf{s}^j)$.

Satisfiability is NP-hard (2)

The following subformulas are related to initial and final configurations:

$$qinit^0,\ 1^0,\ \$^0,\ w[1]_1^0, \ldots, w[n]_n^0,\ \flat_{n+1}^0, \ldots, \flat_{p(n)}^0,\ qacc^{p(n)}.$$

These subformulas ensure that, given an interpretation $\nu \models \varphi_{\mathcal{M},w}$,
$\mathbf{c}_\nu^0$ is the initial configuration and $\mathbf{c}_\nu^{p(n)}$ is an accepting configuration.

Let $\delta(q,a) = \{(nq_1(q,a), na_1(q,a), dp_1(q,a)), \ldots, (nq_K(q,a), na_K(q,a), dp_K(q,a))\}$.

The following subformulas are related to the steps of $\mathcal{M}$

for all $j < p(n)$, $i \leq p(n)$, $a \in \Sigma$:

- $(\neg i^j) \Rightarrow (a_i^j \Leftrightarrow a_i^{j+1})$;
- for all $q \in Q$, $(q^j \wedge i^j \wedge a_i^j) \Rightarrow \bigvee_{k \leq K} nq_k(q,a)^{j+1} \wedge (i + dp_k(q,a))^{j+1} \wedge na_k(q,a)_i^{j+1}$ with a conjunctive clause of the conclusion substituted by **false** when $i + dp_k(q,a) \notin [0, p(n)]$.

These subformulas ensure that, given an interpretation $\nu \models \varphi_{\mathcal{M},w}$,
for all $j < p(n)$, $\mathbf{c}_\nu^j \rightarrow_{\mathcal{M}} \mathbf{c}_\nu^{j+1}$.

- Thus if $\nu \models \varphi_{\mathcal{M},w}$ then $\mathbf{c}_\nu^0, \ldots, \mathbf{c}_\nu^{p(n)}$ is an accepting run for w.
- Conversely assume $\mathbf{c}^0, \ldots, \mathbf{c}^{p(n)}$ is an accepting run.

Then ν, defined by for all j $\nu(\mathbf{s}^j) = \nu_{\mathbf{c}^j}(\mathbf{s}^j)$, satisfies φ.

Formulas in CNF

A formula φ is in *conjunctive normal form* (CNF) if:

- $\varphi = \bigwedge_{i \leq m} \psi_i$ where every ψ_i is a *clause*;
- A clause $\psi_i = \bigvee_{j \leq n_i} \theta_{i,j}$ where every $\theta_{i,j}$ is a *literal*;
- A literal is either some p or $\neg p$ where p is a proposition.

φ and ψ are *equivalent* if for all ν, $\nu(\varphi) = \nu(\psi)$.

For all formula, one can build an equivalent CNF formula as follows.

- Push the negations in front of the propositions:
  - $\neg\neg\varphi \equiv \varphi$;
  - $\neg(\varphi_1 \vee \varphi_2) \equiv (\neg\varphi_1) \wedge (\neg\varphi_2)$;
  - $\neg(\varphi_1 \wedge \varphi_2) \equiv (\neg\varphi_1) \vee (\neg\varphi_2)$.
- Push the disjunctions below the conjunctions:
  $(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \equiv (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3)$.

However this is an exponential time procedure and this blowup is unavoidable.

3SAT is NP-complete (1)

Let φ be CNF formula where all clause has at most 3 literals. Then 3SAT asks whether φ is satisfiable.

Let φ be an arbitrary formula, one builds in PTIME a 3CNF formula ψ as follows.

- For all occurrence of an operator, add a new proposition and label the node of the syntactical tree by this proposition;
- The clauses of ψ are defined as follows. The proposition labelling the root is a clause and for all inner node of the tree:
  - If it is a negation labelled by x and y labels its son, then $\neg x \vee \neg y$ and $x \vee y$ are clauses;
  - If it is a conjunction labelled by x and y, z label its sons, then $x \vee \neg y \vee \neg z$ and $\neg x \vee y$, $\neg x \vee z$ are clauses;
  - If it is a disjunction labelled by x and y, z label its sons, then $\neg x \vee y \vee z$ and $x \vee \neg y$, $x \vee \neg z$ are clauses.

$$\varphi = (\neg(p \wedge q)) \vee r$$

$$\psi = xv \wedge (\neg xv \vee xn \vee r) \wedge (xv \vee \neg xn) \wedge (xv \vee \neg r)$$
$$\wedge\ (\neg xn \vee \neg xw) \wedge (xn \vee xw)\ \wedge$$
$$(xw \vee \neg p \vee \neg q) \wedge (\neg xw \vee p) \wedge (\neg xw \vee q)$$

3SAT is NP-complete (2)

φ is satisfiable if and only if ψ is satisfiable.

Proof.

• Assume $\nu \models \varphi$.

Let ν' extending ν on the new propositions as follows.

Let x be a proposition corresponding to an inner node with subformula φ_x.

Then choose $\nu'(x) = \nu(\varphi_x)$.

It is routine to check that $\nu' \models \psi$.

• Assume $\nu \models \psi$.

By induction on the size of subformulas, one proves that $\nu(x) = \nu(\varphi_x)$.

Considering the clause x where x labels the root, one gets $\nu(\varphi) = \mathbf{true}$.

The Hamiltonian Circuit Problem (1)

Let $G = (V, E)$ be a directed graph with $V = \{v_0, \ldots, v_{n-1}\}$.

A Hamiltonian circuit is a permutation σ of $\{0, \ldots, n-1\}$ such that:

$$\text{for all } 0 \leq i < n,\ (v_{\sigma(i)}, v_{\sigma(i+1\%n)}) \in E$$

The *Hamiltonian circuit problem* asks whether a Hamiltonian circuit exists.

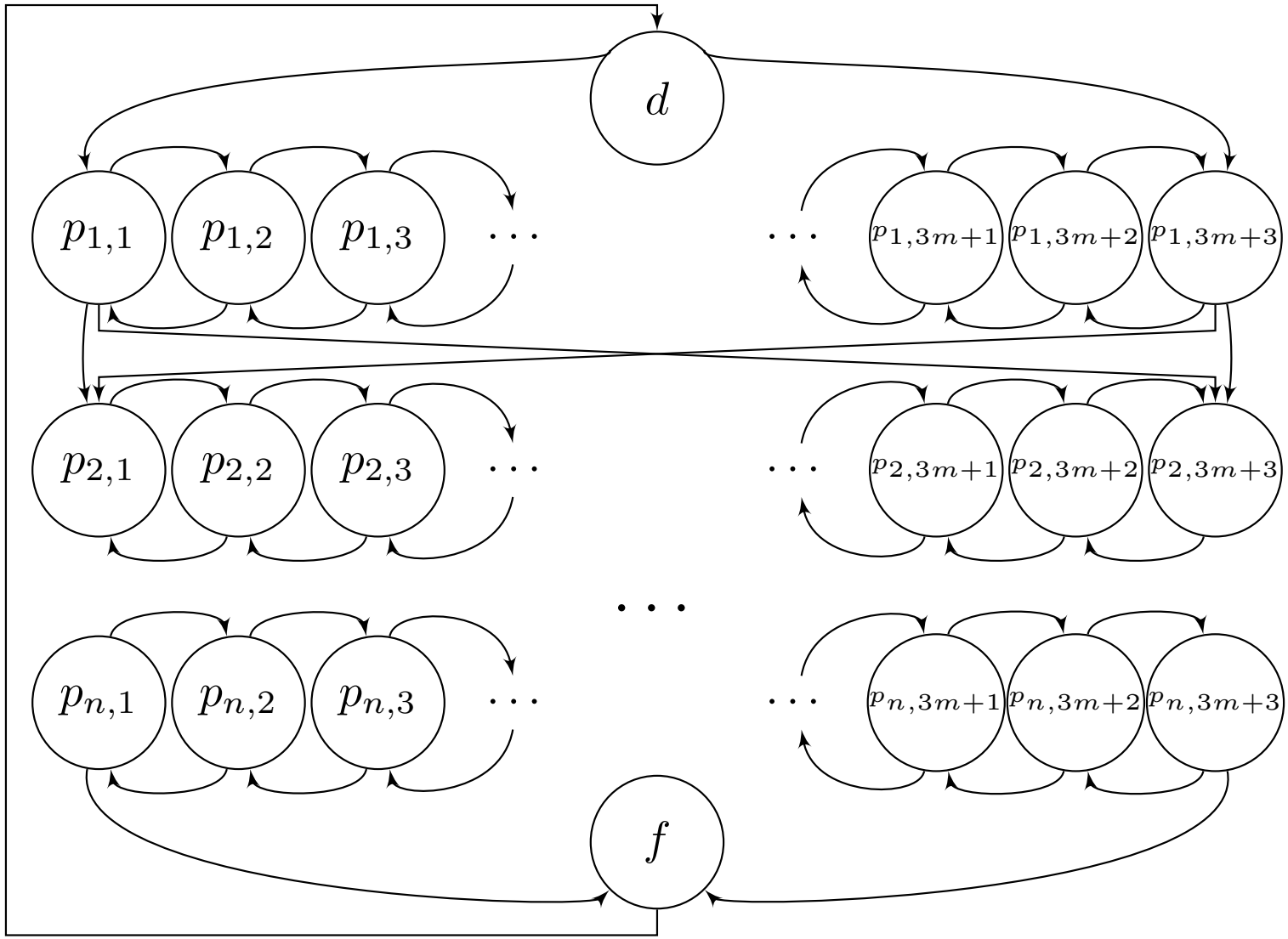
The Hamiltonian circuit problem belongs to NP.

- Guess in linear time a mapping σ from $\{0, \ldots, n-1\}$ to itself.
- Check in polynomial time that σ is a permutation.
- Check in linear time that σ defines a circuit.

The Hamiltonian Circuit Problem (2)

Let φ be a CNF formula with atomic propositions $p_1, \ldots, p_n$ and m clauses $\{c_j\}_{1 \leq j \leq m}$ where no clause includes a proposition and its negation.

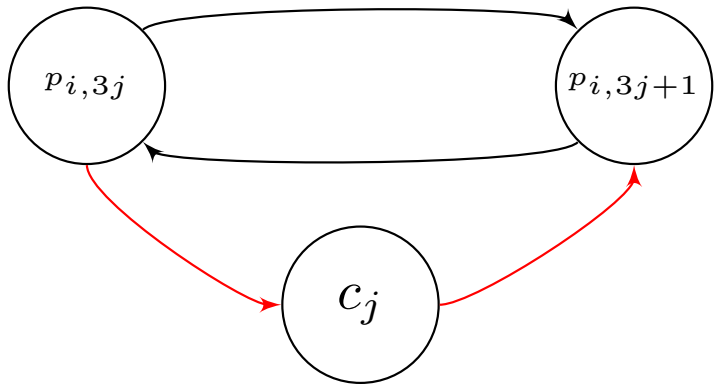
One builds G_φ in two steps.



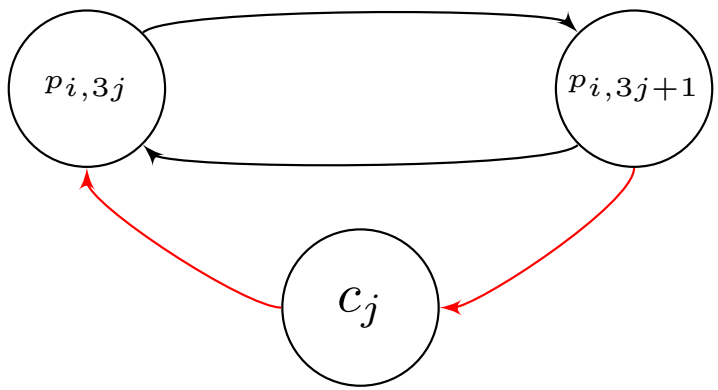
The Hamiltonian Circuit Problem (3)

One also adds vertices $\{c_j\}_{1 \leq j \leq m}$.

Let p_i occurring in c_j. Then one adds the "detour":



Let $\neg p_i$ occurring in c_j. Then one adds the "detour":



The Hamiltonian Circuit Problem (4)

Assume ν fulfills φ.

Then one builds a (non Hamiltonian) circuit as follows.

- Start from d.
- If $\nu(p_1) = \mathbf{true}$ (resp. **false**) then go to $p_{1,1}$ (resp. $p_{1,3m+3}$) and continue on the right (resp. left) until $p_{1,3m+3}$ (resp. $p_{1,1}$);
- Iterate this process for $i = 2, \ldots, n$;
- go to f and back to d.

For all clause c_j, pick a literal x occurring in c_j such that $\nu(x) = \mathbf{true}$.

- If $x = p_i$ then change $p_{i,3j} \to p_{i,3j+1}$ by $p_{i,3j} \to c_j \to p_{i,3j+1}$;
- If $x = \neg p_i$ then change $p_{i,3j+1} \to p_{i,3j}$ by $p_{i,3j+1} \to c_j \to p_{i,3j}$.

The Hamiltonian Circuit Problem (5)

Assume there exists a Hamiltonian circuit. We claim that:

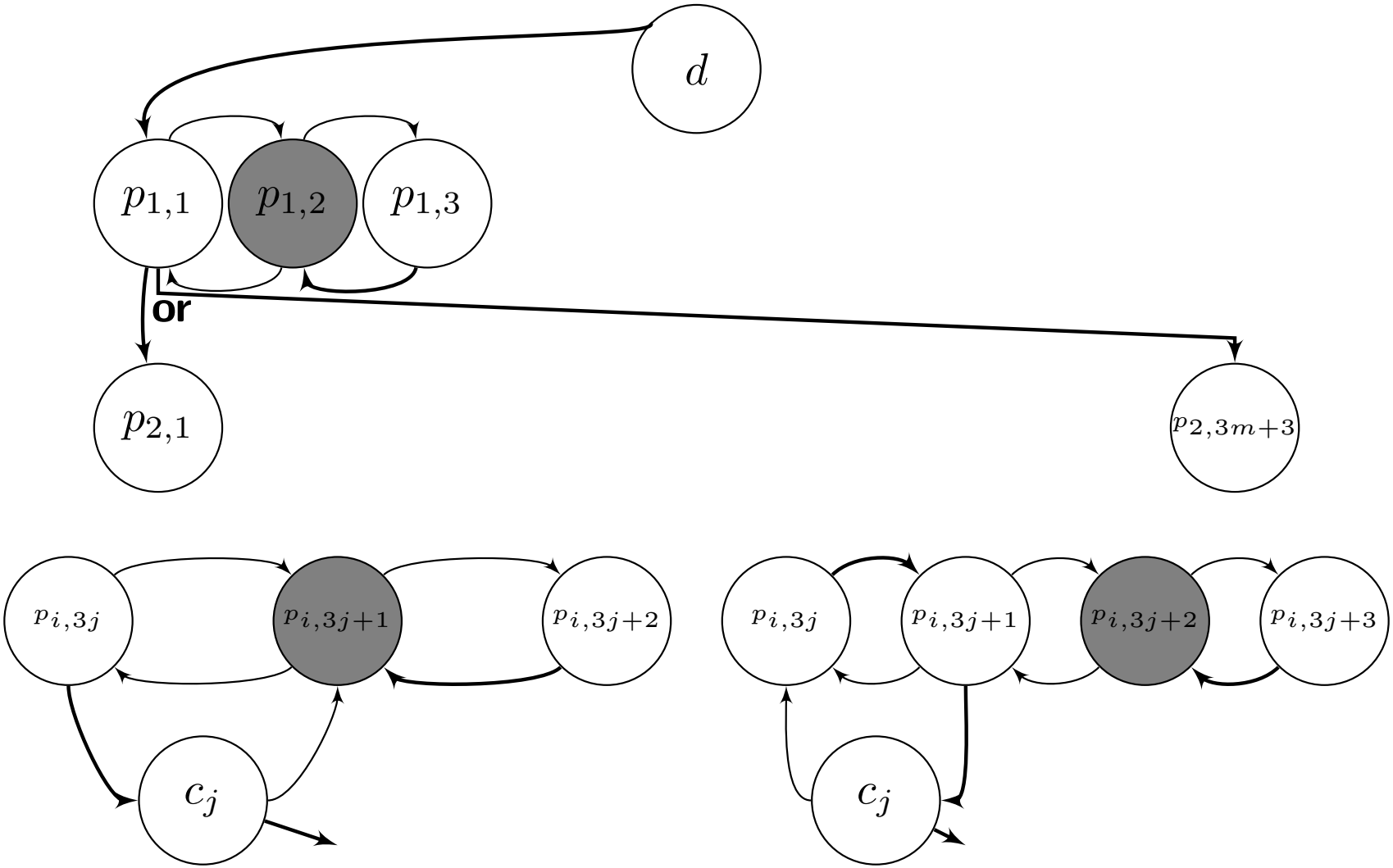
- starting from d, the circuit goes (in descending order) through the "rows" from left to right or right to left;
- with exactly one detour per clause.

If the claim is valid then:

- choosing $\nu(p_i) = \mathbf{true}$ if the row i is crossed from left to right;
- by examination of the detours $\nu \models \varphi$.

The Hamiltonian Circuit Problem (6)

Some impossible Hamiltonian circuits.



The Hamiltonian Cycle Problem (1)

Let $G = (V, E)$ be a *non* directed graph with $V = \{v_0, \ldots, v_{n-1}\}$.

A Hamiltonian cycle is a permutation σ of $\{0, \ldots, n-1\}$ such that:

$$\text{for all } 0 \leq i < n,\ \{v_{\sigma(i)}, v_{\sigma(i+1\%n)}\} \in E$$

The *Hamiltonian cycle problem* asks whether a Hamiltonian cycle exists.

The Hamiltonian cycle problem belongs to NP.

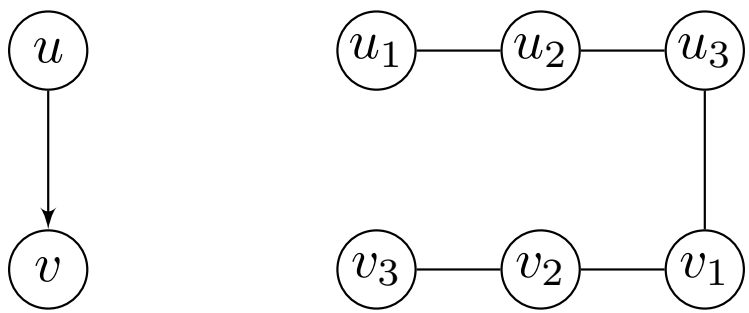
- Guess in linear time a mapping σ from $\{0, \ldots, n-1\}$ to itself.
- Check in polynomial time that σ is a permutation.
- Check in linear time that σ defines a cycle.

The Hamiltonian Cycle Problem (2)

Let $G = (V, E)$ be a directed graph.

One builds a non directed graph $G = (V', E')$ as follows.

- $V' = \{u_i \mid u \in V \wedge i \in \{1, 2, 3\}\}$;
- $E' = \{\{u_3, v_1\} \mid (u, v) \in E\} \cup \{\{u_1, u_2\}, \{u_2, u_3\} \mid u \in V\}$.



The Hamiltonian Cycle Problem (3)

Denote $V = \{0, \ldots, n-1\}$.

• Assume there is a Hamiltonian circuit of G: $(\sigma(0), \ldots, \sigma(n-1))$.

By construction, $(\sigma(0)_1, \sigma(0)_2, \sigma(0)_3, \ldots, \sigma(n-1)_1, \sigma(n-1)_2, \sigma(n-1)_3)$ is a Hamiltonian cycle of G'.

• Assume there is a Hamiltonian cycle of G'.

By construction, for all $u \in V$ either u_1, u_2, u_3 or u_3, u_2, u_1 occurs in the cycle.

W.l.o.g. assume that the cycle starts with $0_1, 0_2, 0_3$.

By induction the cycle cannot revert this order since there is no edge $\{u_3, v_3\}$.

So the cycle can be denoted $(\sigma(0)_1, \sigma(0)_2, \sigma(0)_3, \ldots, \sigma(n-1)_1, \sigma(n-1)_2, \sigma(n-1)_3)$ implying that σ defines a circuit in G.

The Subset Sum Problem (1)

Let $\{v_1, \ldots, v_n, w\}$ be $n+1$ integers.
The *subset sum problem* asks whether there exists $I \subseteq \{1, \ldots, n\}$ such that:

$$\sum_{i \in I} v_i = w$$

The subset sum problem belongs to NP.

- Guess I in linear time;
- Compute $\sum_{i \in I} v_i$ in linear time;
- Check whether $\sum_{i \in I} v_i = w$ in linear time.

The Subset Sum Problem (2)

Let φ be a 3CNF formula with m clauses $\{c_j\}_{j \leq m}$ and n atomic propositions $p_1, \ldots, p_n$.

Build the numbers $\{v_1, \ldots, v_{2n+2m}\}$ written with $n+m$ *digits* where we consider that every digit is associated either with a clause or a proposition.

Given a number x, $x[j]$ is the j^{th} digit of x.

- for all $i \leq n$, for all $j \leq n$, $v_{2i-1}[j] = v_{2i}[j] = 1_{i=j}$;
- for all $i \leq n$, for all $j \leq m$,
  1. if p_i occurs in c_j then $v_{2i-1}[n+j] = 1$ else $v_{2i-1}[n+j] = 0$;
  2. if $\neg p_i$ occurs in c_j then $v_{2i}[n+j] = 1$ else $v_{2i}[n+j] = 0$;
- for all $i \leq m$, for all $j \leq n+m$, $v_{2n+2i-1}[j] = v_{2n+2i}[j] = 1_{n+i=j}$.

For all $j \leq n$, $w[j] = 1$ and for all $n < j \leq n+m$, $w[j] = 3$.

The Subset Sum Problem (3)

The table describes the different numbers with digits from left to right.

Here we have supposed that p_i occurs in c_j and $\neg p_i$ occurs in c_k.

	1	i	n	$n+1$	$n+j$	$n+k$	$n+m$
v_{2i-1}	0	1	0		1	0	
v_{2i}	0	1	0		0	1	
$v_{2n+2j-1}, v_{2n+2j}$	$\cdots$	0	$\cdots$	0	1	0	0
w	1	$\cdots$	1	3	$\cdots$	$\cdots$	3

The Subset Sum Problem (4)

• Assume $\nu \models \varphi$.

Then define I by:

- If $\nu(p_i) = \mathbf{true}$ then $2i - 1 \in I$ else $2i \in I$;
- Let $1 \leq \sharp_j \leq 3$ be the number of literals of c_j satisfied by ν.
  1. if $\sharp_j < 3$ then $2n + 2j - 1 \in I$;
  2. if $\sharp_j = 1$ then $2n + 2j \in I$.

It is routine to check, digit by digit, that $\sum_{i \in I} v_i = w$.

• Assume there exists I such that $\sum_{i \in I} v_i = w$.

Observe that, whatever I, $\sum_{i \in I} v_i$ does not rise a carry.

Thus for all $1 \leq i \leq n$, either v_{2i-1} or v_{2i} but not both belongs to I.

Define ν by $\nu(p_i) = \mathbf{true}$ if $v_{2i-1} \in I$.

Let c_j be a clause. Since there are only two numbers in $\{v_j\}_{j>2n}$ with 1 in the $n + j^{th}$ digit there is a v_j with $j \leq 2n$ and $j \in I$ with 1 in the $n + j^{th}$ digit.

- If $j = 2i - 1$ then p_i occurs in c_j and $\nu(p_i) = \mathbf{true}$ so $\nu(c_j) = \mathbf{true}$;
- If $j = 2i$ then $\neg p_i$ occurs in c_j and $\nu(p_i) = \mathbf{false}$ so $\nu(c_j) = \mathbf{true}$.

The Subset Sum Problem (4)

The subset sum problem is *pseudo-polynomial*: i.e. polynomial when the integers are written in unary.

The following algorithm operates in $O(nw)$.

For i **from** 1 **to** w **do** $T[i] \leftarrow$ **false**

$T[0] \leftarrow$ **true**

For i **from** 1 **to** n **do**

  For j **from** $w - v_i$ **downto** 0 **do**

    If $T[j]$ **then** $T[j + v_i] \leftarrow$ **true**

Return $T[w]$

The Weighted Graph Problem (1)

Let $G = (V, E)$ be a directed graph and $p : E \to \mathbb{N}$ a weight on edges.

The weight of a path $(v_0, \ldots, v_n)$ is (additively) defined by $\sum_{i<n} p(v_i, v_{i+1})$.

Given G, p, $s, t \in V$ and $a \in \mathbb{N}$, the *weighted graph problem* asks whether there exists a path from s to t whose weight is a.

This problem is NP-hard by reduction from the subset sum problem.

A shortest path from s to t of weight a may have (exponential) length:

$$|V|(a+1) - 1$$

So guessing a path does not show that this problem belongs to NP!

The Weighted Graph Problem (2)

The *Parikh image* $\mathbf{v}_\rho \in \mathbb{N}^E$ of a path $\rho \in E^*$ is the vector who counts the occurrence of edges in ρ inductively defined by:

$$\mathbf{v}_\varepsilon[e] = 0, \text{ for } e' \neq e, \ \mathbf{v}_{\rho e'}[e] = \mathbf{v}_{\rho e'}[e] \text{ and } \mathbf{v}_{\rho e}[e] = \mathbf{v}_\rho[e] + 1$$

The *support* $Supp(\mathbf{v})$ of a vector $\mathbf{v} \in \mathbb{N}^E$ is defined by:

$$Supp(\mathbf{v}) = \{e \in E \mid \mathbf{v}[e] > 0\}$$

Let $E' \subseteq E$, $G_{E'} = (V_{E'}, E')$ the graph *induced* by E' is defined by:

$$V_{E'} = \{v, v' \mid (v, v') \in E'\}$$

Euler Lemma. Let $G = (V, E)$ be a graph, $s \neq t \in V$ and $\mathbf{v} \in \mathbb{N}^E$.

Then $\mathbf{v}$ is the Parikh image of path from s to t if and only if:

- $G_{Supp(\mathbf{v})}$ is connected;
- $\sum_{(s,u)\in E} \mathbf{v}[(s,u)] = 1 + \sum_{(u,s)\in E} \mathbf{v}[(u,s)]$;
- $\sum_{(u,t)\in E} \mathbf{v}[(u,t)] = 1 + \sum_{(t,u)\in E} \mathbf{v}[(t,u)]$;
- For all $w \notin \{s,t\}$, $\sum_{(u,w)\in E} \mathbf{v}[(u,w)] = \sum_{(w,u)\in E} \mathbf{v}[(w,u)]$.

The necessity of the conditions is straightforward.

The Weighted Graph Problem (3)

Assume that $\mathbf{v}$ satisfies the hypotheses of Euler Lemma.

• Build a path starting from s.

Iterate the following process until blocking.

- Let u be the current end of the path;
- If there is an edge $e = (u, u')$ in $Supp(\mathbf{v})$ then concatenate e to the path;
- Decrement $\mathbf{v}[e]$.

Looking at the hypotheses of Euler Lemma,
the last vertex of the path is t.

• If $\mathbf{v} = 0$ we are done.

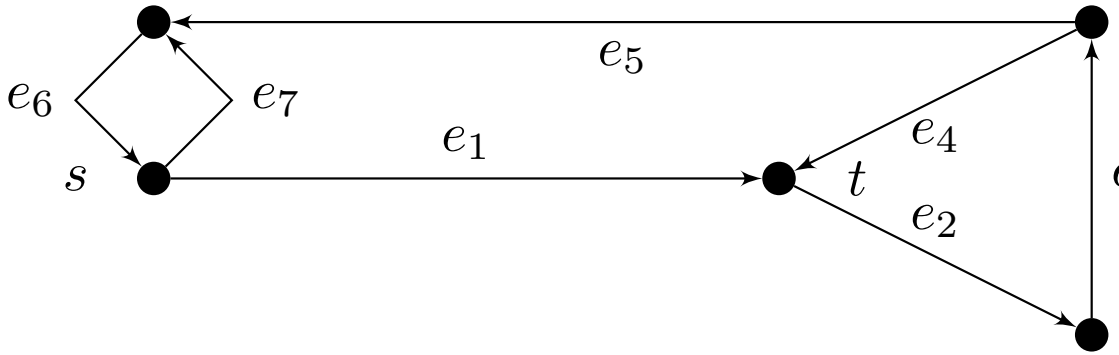
• Otherwise, *by connectivity*, there is an edge $(u, u') \in Supp(\mathbf{v})$ with u belonging to the path.

Repeat the previous process.

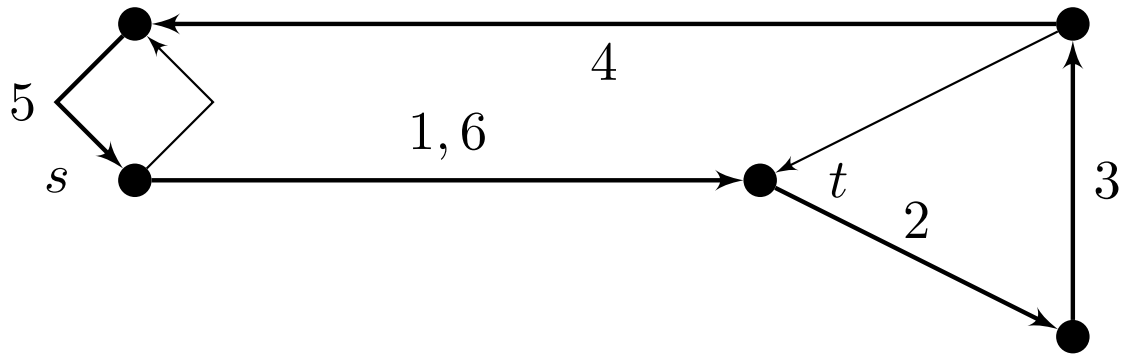
Looking at the hypotheses of Euler Lemma,
this new path is a cycle that can be added to the path.

• By iteration, one builds a path ρ with $\mathbf{v}_\rho = \mathbf{v}$.

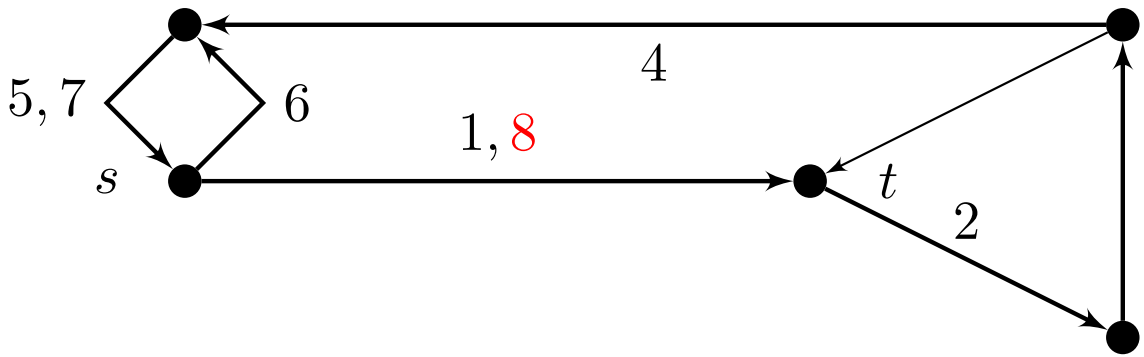
The Weighted Graph Problem (4)



$(2, 1, 1, 0, 1, 2, 1)$



$(0, 0, 0, 0, 0, 1, 1)$



$(0, 0, 0, 0, 0, 0, 0)$

The Weighted Graph Problem (5)

A non deterministic polynomial time algorithm

A vertex is visited by a shortest path of weight a from s to t at most $a+1$ times.

1. Guess a vector $\mathbf{v} \in [0, a+1]^E$.

2. Check the conditions of Euler Lemma.

3. Check that $\sum_{e \in E} \mathbf{v}[e]p(e) = a$.

Plan

Savitch Theorem (1)

Let $G = (V, E)$ be a directed graph and $s \neq t \in V$.
Then the existence of a path from s to t is decidable in space $O(\log^2(|V|))$.

Reach(u, v, ℓ)
  If $\ell \leq 1$ **return**$(u = v \vee (u, v) \in E)$
  For $w \in V$ **do**
    If Reach$(u, w, \lfloor \frac{\ell}{2} \rfloor) \wedge$ Reach$(w, v, \lceil \frac{\ell}{2} \rceil)$ **return**$(\mathbf{true})$
  return$(\mathbf{false})$
Reach$(s, t, |V| - 1)$

At most $\lceil \log(|V|) \rceil$ nested calls of Reach.
u, v, w, ℓ are represented in space $O(\log(|V|))$.

Warning: This algorithm does not operate in polynomial time.

Savitch Theorem (2)

For all measure function $f(n) \geq \log(n)$, NSPACE$(f(n)) \subseteq$ SPACE$(f^2(n))$.

Proof.

Let $\mathcal{M}$ be a NTM operating in non deterministic space $f(n)$ on a word w of size n.

W.l.o.g. there is a single accepting configuration.

Define $\mathcal{M}'$, a DTM that operates as follows on w.

- Compute $f(n)$
- Execute a *modified* Savitch algorithm to check the reachability of the accepting configuration from the initial configuration in the configuration graph $G_{\mathcal{M},w}$.

The modification consists in replacing the test $(u, v) \in E$ by $u \to_{\mathcal{M}} v$ *avoiding* to build $G_{\mathcal{M},w}$.

□

Corollary: NPSPACE = PSPACE

Warning: This does not imply NLOGSPACE = LOGSPACE.

Quantified Boolean Formula (QBF)

A quantified boolean formula φ is inductively defined by:

- φ may be **true**, **false** or an atomic proposition;
- $\varphi = \neg\varphi_1$, $\varphi = \varphi_1 \vee \varphi_2$, $\varphi = \varphi_1 \wedge \varphi_2$ where φ_1, φ_2 are QBF;
- $\varphi = \exists p\ \varphi_1$, $\varphi = \forall p\ \varphi_1$, where φ_1 is a QBF and p is an atomic proposition not quantified in φ_1.

Let ν be an interpretation. Then ν may be extended to QBF as follows.

- $\nu(\neg\varphi_1) = \overline{\neg}(\nu(\varphi_1))$;
- $\nu(\varphi_1 \vee \varphi_2) = \overline{\vee}(\nu(\varphi_1), \nu(\varphi_2))$;
- $\nu(\varphi_1 \wedge \varphi_2) = \overline{\wedge}(\nu(\varphi_1), \nu(\varphi_2))$;
- $\nu(\exists p\ \varphi_1) = \overline{\vee}(\nu(\varphi_1[\mathbf{true}/p]), \nu(\varphi_1[\mathbf{false}/p]))$;
- $\nu(\forall p\ \varphi_1) = \overline{\wedge}(\nu(\varphi_1[\mathbf{true}/p]), \nu(\varphi_1[\mathbf{false}/p]))$.

Quantifiers may be pushed at the beginning of the formula (*prenex form*):

- $\neg\exists p\ \varphi \equiv \forall p\ \neg\varphi$;
- $(\exists p\ \varphi) \wedge \theta \equiv \exists q\ (\varphi[q/p] \wedge \theta)$ where q does not occur in φ and θ.
- etc.

Evaluation of a QBF

An occurrence of a proposition is either under the scope of a quantifier or *free*.

A *closed* QBF is a QBF where all occurrences of propositions are quantified.

The truth value of a closed QBF is independent of the interpretation.

EvalQBF($Q_1 p_1 \ldots Q_n p_n \ \psi$)

If $n = 0$ **then return**(EvalCons(ψ))

$b_1 \leftarrow$ EvalQBF($Q_2 p_2 \ldots Q_n p_n \ \psi[\mathbf{true}/p_1]$)

$b_2 \leftarrow$ EvalQBF($Q_2 p_2 \ldots Q_n p_n \ \psi[\mathbf{false}/p_1]$)

If $Q_1 = \exists$ **then return**(b_1 **or** b_2) **else return**(b_1 **and** b_2)

This algorithm operates in polynomial space.

So the (closed) QBF evaluation problem belongs to PSPACE.

We will prove that (closed) QBF evaluation problem is PSPACE-hard.

A Reduction for PSPACE-hardness (1)

Let $\mathcal{M}$ be a DTM operating in polynomial space $p(n)$ where n is the size of the input w.

We code a configuration by the following propositions indexed by some x:

- For all $q \in Q$, q^x is **true** if the state is q;
- For all $0 \leq i \leq p(n)$, i^x is **true** if the position of the head is i;
- For all $0 \leq i \leq p(n), a \in \Sigma$, a_i^x is **true** if the i^{th} cell contains a.

The set of these propositions is denoted **x** and also $\{x_i\}_{i \leq m}$ (via some renaming).

Observe that $m = O(p(n))$.

Let **c** be a configuration. The interpretation of **x** corresponding to **c** is denoted $\nu_{\mathbf{c}}^x$.

We define a QBF formula φ_i with free propositions **x** and **y** such that for all configurations **c**, **c**':

- Letting ν be defined by $\nu(\mathbf{x}) = \nu_{\mathbf{c}}^x(\mathbf{x})$ and $\nu(\mathbf{y}) = \nu_{\mathbf{c}'}^y(\mathbf{y})$;
- $\nu(\varphi_i) = \mathbf{true}$ iff **c**' is reachable from **c** in at most 2^i steps;
- For all ν' such that $\nu'(\mathbf{x}) = \nu_{\mathbf{c}}^x(\mathbf{x})$ and $\nu'(\varphi_i) = \mathbf{true}$ there exists a configuration **c**'' such that $\nu'(\mathbf{y}) = \nu_{\mathbf{c}''}^y(\mathbf{y})$.

A Reduction for PSPACE-hardness (2)

Equality of configurations

Let $\theta = \bigwedge_{i \leq m} x_i \Leftrightarrow y_i$.

A step of $\mathcal{M}$

Let $\delta(q, a) = (nq(q, a), na(q, a), dp(q, a))$.

Let ψ be the conjunction of the subformulas:

- For all i, a $\neg i^x \Rightarrow a_i^x \Leftrightarrow a_i^y$
- For all i, q, a $i^x \wedge q^x \wedge a_i^x \Rightarrow nq(q, a)^y \wedge na(q, a)_i^y \wedge (i + dp(q, a))^y$ with a conjunctive clause of the conclusion substituted by $\mathbf{false}$ when $i + dp(q, a) \notin [0, p(n)]$.
- For all $q \neq q'$ $\neg q^y \vee \neg (q')^y$
- For all $i \neq i'$ $\neg i^y \vee \neg (i')^y$
- For all $a \neq a'$, for all i, $\neg a_i^y \vee \neg a'^y_i$

Then $\varphi_0 = \theta \vee \psi$.

A Reduction for PSPACE-hardness (3)

The formula φ_{i+1} guesses an intermediate configuration.

First attempt.

$$\varphi_{i+1} = \exists \mathbf{z}\ \varphi_i[\mathbf{z}/\mathbf{y}] \wedge \varphi_i[\mathbf{z}/\mathbf{x}]$$

Problem: the size of φ_{i+1} is twice the size of φ_i.

Using a logical trick.

$$\varphi_{i+1} = \exists \mathbf{z}\ \forall \mathbf{t}\ \forall \mathbf{u}\ (\mathbf{t} = \mathbf{x} \wedge \mathbf{u} = \mathbf{z}) \vee (\mathbf{t} = \mathbf{z} \wedge \mathbf{u} = \mathbf{y}) \Rightarrow \varphi_i[\mathbf{t}/\mathbf{x}, \mathbf{u}/\mathbf{y}]$$

The formula corresponding to the reduction is $\varphi_m[\nu_{\mathbf{c}_0}/\mathbf{x}, \nu_{\mathbf{c}_f}/\mathbf{y}]$
where $\mathbf{c}_0$ (resp. $\mathbf{c}_f$) is the initial (resp. accepting) configuration.

Another PSPACE-complete problem

The QBF evaluation problem where $\varphi = Q_1 p_1 \ldots Q_n p_n \ \psi$ with ψ in 3CNF is PSPACE-complete.

Proof.

Let $\varphi = Q_1 p_1 \ldots Q_n p_n \ \theta$ be a QBF formula.

Let ψ be the 3CNF formula of the reduction for 3SAT starting from θ.

Let $x_1, \ldots, x_m$ be the additional propositions of ψ.

Our previous proof establishes that $\theta \equiv \exists x_1 \ldots \exists x_m \ \psi$.

Thus $\varphi \equiv Q_1 p_1 \ldots Q_n p_n \ \exists x_1 \ldots \exists x_m \ \psi$.

The Universality of Regular Languages

Let Σ be an alphabet. A regular expression E is inductively defined by:

- E is $\emptyset$, ε or $a \in \Sigma$;
- E is $E_1 + E_2$, $E_1 \cdot E_2$, E_1^* where E_1, E_2 are regular expressions.

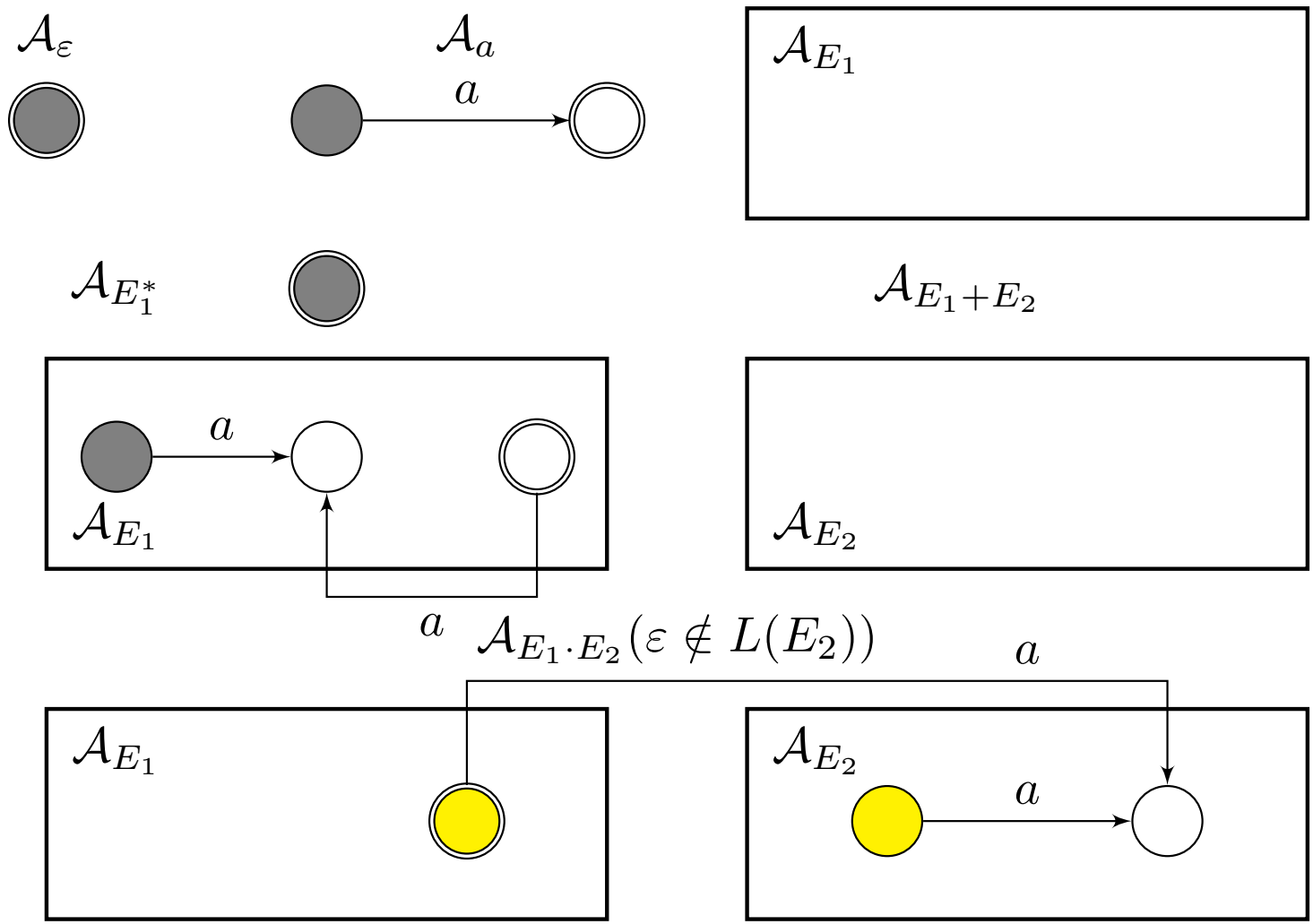
The regular language $L(E) \subseteq \Sigma^*$ is inductively defined by:

- $L(\varepsilon) = \{\varepsilon\}$, $L(a) = \{a\}$;
- $L(E_1 + E_2) = L(E_1) \cup L(E_2)$;
- $L(E_1 \cdot E_2) = \{w_1 w_2 \mid w_1 \in L(E_1) \wedge w_2 \in L(E_2)\}$;
- $L(E_1^*) = \{w_1 \dots w_n \mid \forall i \leq n \; w_i \in L(E_1)\}$.

The *universability problem* for a regular expression E asks whether $L(E) = \Sigma^*$.

Solving the Universality Problem (1)

Build a non deterministic automaton $\mathcal{A}_E$ that accepts $L(E)$.



Solving the Universality Problem (2)

Let $\overline{\mathcal{A}}_E$ be the *complementary* automaton of $\mathcal{A}_E$ obtained by the determinization procedure.

Check in $\overline{\mathcal{A}}_E$ without building it that $L(\overline{\mathcal{A}}_E) \neq \emptyset$.

Let Q the set of states of $\mathcal{A}_E$ with Q_0 the initial states and Q_f the final states.

$cpt \leftarrow 0$
$Q' \leftarrow Q_0$
Repeat
  If $Q' \cap F = \emptyset$ **return true**
  $cpt \leftarrow cpt + 1$
  Guess $a \in \Sigma$
  $Q' \leftarrow \bigcup_{q \in Q'} \delta(q, a)$
until $cpt = 2^{|Q|}$
return false

Thus the universality problem belongs to NPSPACE = PSPACE.

Universality Problem is PSPACE-hard (1)

Let $\mathcal{M}$ be DTM operating in space $p(n)$ on a word w (with $w[1] = \$$) of size n.

We code a run of $\mathcal{M}$ by a word of Σ^*:

- Let T be the alphabet of the tape.
  Then $QT = \{qX \mid q \in Q \wedge X \in T\}$, $\Delta = T \cup QT$ and $\Sigma = \Delta \cup \{\sharp\}$;
- A configuration $\mathbf{c} = (t, i, q)$ where t is the tape content, i the position of the head and q is the state is coded by a word $v_{\mathbf{c}} = v_{\mathbf{c}}[1] \dots v_{\mathbf{c}}[p(n)]$ with:
  1. for all $j \neq i$, $v_{\mathbf{c}}[j] = t[j]$;
  2. $v_{\mathbf{c}}[i] = qt[i]$.
- A run $\mathbf{c}_1 \dots \mathbf{c}_k$ is coded by the word $\sharp v_{\mathbf{c}_1} \sharp \dots \sharp v_{\mathbf{c}_k} \sharp$.

We build a regular expression $E_{\mathcal{M},w}$ that specifies:

- the words that are not code of runs of $\mathcal{M}$ on w;
- the words that are code of non accepting runs of $\mathcal{M}$ on w.

Thus $L(E_{\mathcal{M},w}) = \Sigma^*$ if and only if w is not accepted by $\mathcal{M}$.

Universality Problem is PSPACE-hard (2)

Let us decompose $E_{\mathcal{M},w} = A + B + C + D$ where expressions A, B, C, D correspond to the possible cases.

1. A denotes the words v that do not contain some letter $q_f X$;
2. B denotes the words v such that $\sharp v_{\mathbf{c_0}}$ is not a prefix of v;
3. C denotes the words v that have not the following pattern $v = \sharp v_1 \sharp v_2 \sharp \dots \sharp v_k \sharp$ where for all $i \leq k$:
   - 3.1 $|v_i| = p(n)$;
   - 3.2 exactly one letter of v_i belongs to QT;
   - 3.3 all other letters of v_i belong to T.
4. D denotes the words v with $|v| > 2 + p(n)$ that cannot contain two successive configurations of $\mathcal{M}$.

Notations.

- Let $S = \{s_1, \dots, s_k\} \subseteq \Sigma$, $\hat{S} \equiv s_1 + \dots + s_k$;
- Let E be an expression and k be an integer, $E^k \equiv E \cdot E \cdots E \cdot E$ with k occurrences of E.

Universality Problem is PSPACE-hard (3)

Let $\Sigma_r = \Sigma \setminus \{q_f X \mid X \in T\}$. Then $A = \hat{\Sigma}_r^*$.

Let:

- $E_{2,1} = \hat{\Delta} \cdot \hat{\Sigma}^*$;
- Let $\Sigma_1 = \Sigma \setminus \{q_0 w[1]\}$. Then $E_{2,2} = \hat{\Sigma} \cdot \hat{\Sigma}_1 \cdot \hat{\Sigma}^*$;
- For all $2 \leq i \leq n$, let $\Sigma_i = \Sigma \setminus \{w[i]\}$. Then $E_{2,i} = \hat{\Sigma}^i \cdot \hat{\Sigma}_i \cdot \hat{\Sigma}^*$;
- Let $\Sigma_\flat = \Sigma \setminus \{\flat\}$. Then for all $n+1 \leq i \leq p(n), E_{2,i} = \hat{\Sigma}^i \cdot \hat{\Sigma}_\flat \cdot \hat{\Sigma}^*$.

Then $B = E_{2,1} + \ldots + E_{2,p(n)} + \varepsilon + \hat{\Sigma} + \cdots + \hat{\Sigma}^{p(n)}$.

Let:

- For all $0 \leq i \leq p(n) - 1$, $E_{3,i} = \hat{\Sigma}^* \cdot \sharp \cdot \hat{\Delta}^i \cdot \sharp \cdot \hat{\Sigma}^*$ *(too short configurations)*
- $E_{3,p(n)+1} = \hat{\Sigma}^* \cdot \sharp \cdot \hat{\Delta}^{p(n)+1} \cdot \hat{\Sigma}^*$ *(too long configurations)*
- $F_3 = \hat{\Delta}^* + \hat{\Delta}^* \cdot \sharp \cdot \hat{\Delta}^* + \hat{\Delta} \cdot \hat{\Sigma}^* + \hat{\Sigma}^* \cdot \hat{\Delta}$ *(wrong pattern of $\sharp$)*
- $G_3 = \hat{\Sigma}^* \cdot \sharp \cdot T^* \cdot \sharp \cdot \hat{\Sigma}^* + \hat{\Sigma}^* \cdot \widehat{QT} \cdot \hat{T}^* \cdot \widehat{QT} \cdot \hat{\Sigma}^*$ *(wrong pattern of states)*

Then $C = E_{3,1} + \cdots + E_{3,p(n)-1} + E_{3,p(n)+1} + F_3 + G_3$.

Universality Problem is PSPACE-hard (4)

Assume that a word v codes a run with $|v| > 2 + p(n)$.

Then for $i > 1$, $v[i + p(n) + 1] = f_{\mathcal{M}}(v[i-1]v[i]v[i+1])$
for some function that only depends on $\mathcal{M}$.
(arbitrarily defined for triples that cannot occur in runs)

Let $g_{\mathcal{M}}$ from Σ^3 to 2^{Σ} defined by $g_{\mathcal{M}}(abc) = \Sigma \setminus \{f_{\mathcal{M}}(abc)\}$

Let $D_{abc} = \hat{\Sigma}^* \cdot a \cdot b \cdot c \cdot \hat{\Sigma}^{p(n)} \cdot \widehat{g_{\mathcal{M}}(abc)} \cdot \hat{\Sigma}^*$

Then $D = \sum_{abc \in \Sigma^3} D_{abc}$.

It is routine to check that E is built in polynomial time.

Plan

Horn Normal Form (HNF)

Let φ be a CNF formula. Then φ is in HNF if for all clause ψ of φ:

- either $\psi = \neg p_1 \vee \cdots \vee \neg p_k \vee q$ (a positive literal)
  equivalently $\psi \equiv (p_1 \wedge \cdots \wedge p_k) \Rightarrow q$
- either $\psi = \neg p_1 \vee \cdots \vee \neg p_k$ (no positive literal)
  equivalently $\psi \equiv (p_1 \wedge \cdots \wedge p_k) \Rightarrow \mathbf{false}$

$H_\psi = p_1 \wedge \cdots \wedge p_k$ is the hypothesis. When $k = 0$, $H_\psi = \mathbf{true}$.
q or $\mathbf{false}$ is the conclusion, denoted C_ψ.

Example.

$$\varphi = (p) \wedge (p \Rightarrow q) \wedge (p \wedge q \Rightarrow r) \wedge (r \wedge s \Rightarrow \mathbf{false})$$

φ is satisfiable: $\nu(p) = \nu(q) = \nu(r) = \mathbf{true}$ and $\nu(r) = \mathbf{false}$.

The HORNSAT *problem* asks whether a HNF formula is satisfiable.

Solving HORNSAT

$Clauses \leftarrow Clauses_\varphi$; **For** $p \in Prop_\varphi$ **do** $\nu(p) = \bot$

Repeat

Invariant: Every $\nu' \models \varphi$ *extends* ν.

Invariant: $\nu(\psi) = \mathbf{true}$ *for* $\psi \in Clauses_\varphi \setminus Clauses$.

Invariant: For all $p \in Prop_\varphi$, $\nu(p) = \mathbf{true}$ *or* $\nu(p) = \bot$.

$done \leftarrow \mathbf{true}$

For $\psi \in Clauses$ **do**

If $\nu(H_\psi) = \mathbf{true}$ **then**

If $C_\psi = \mathbf{false}$ **then return false**

$\nu(C_\psi) \leftarrow \mathbf{true}$; $Clauses \leftarrow Clauses \setminus \{\psi\}$; $done \leftarrow \mathbf{false}$

Until $done$

Assertion: For all $\psi \in Clauses$, *there exists* $p \in H_\psi$ *with* $\nu(\psi) = \bot$

For $p \in Prop_\varphi$ **do if** $\nu(p) = \bot$ **then** $\nu(p) = \mathbf{false}$

return ν

Thus HORNSAT belongs to PTIME.

HORNSAT is PTIME-hard (1)

Let $\mathcal{M}$ be a DTM operating in polynomial time $p(n)$ where n is the size of the input w.

We code a configuration at time $j \leq p(n)$ by the following propositions:

- For all $q \in Q$, q^j is **true** if the state is q;
- For all $0 \leq i \leq p(n)$, i^j is **true** if the position of the head is i;
- For all $0 \leq i \leq p(n), a \in \Sigma$, a_i^j is **true** if the i^{th} cell contains a.

We denote $\mathbf{s}^j$ the set of these propositions.

Given a configuration $\mathbf{c}$ of $\mathcal{M}$ operating on w, $\nu_{\mathbf{c}}^j$ is the interpretation of $\mathbf{s}^j$ corresponding to $\mathbf{c}$.

The formula $\varphi_{\mathcal{M},w}$ is a conjunction of subformulas among them for all $j \leq p(n)$:

- for all $q \neq q' \in Q$, $\neg q^j \vee \neg (q')^j$;
- for all $i < i' \leq p(n)$, $\neg i^j \vee \neg (i')^j$;
- for all $i \leq p(n)$ and for all $a \neq a' \in \Sigma$, $\neg a_i^j \vee \neg (a')_i^j$.

These subformulas ensure that, given an interpretation $\nu \models \varphi_{\mathcal{M},w}$,
for all j, there is *at most* one configuration $\mathbf{c}_\nu^j$ such that $\nu(\mathbf{s}^j) = \nu_{\mathbf{c}_\nu^j}(\mathbf{s}^j)$.

HORNSAT is PTIME-hard (2)

The following subformulas are related to initial and final configurations:

$$qinit^0,\ 1^0,\ \$^0,\ w[1]^0_1, \ldots, w[n]^0_n,\ \flat^0_{n+1}, \ldots, \flat^0_{p(n)},\ qacc^{p(n)}.$$

These subformulas ensure that, given an interpretation $\nu \models \varphi_{\mathcal{M},w}$,
$\mathbf{c}^0_\nu$ is defined and is the initial configuration
and, *if defined*, $\mathbf{c}^{p(n)}_\nu$ is an accepting configuration.

Let $\delta(q,a) = (nq(q,a), na(q,a), dp(q,a))$. The following subformulas are related to the steps of $\mathcal{M}$. For all $j < p(n)$, $i \leq p(n)$, $a \in \Sigma$:

- $i^j \vee a^j_i \vee \neg a^{j+1}_i$, $i^j \vee a^j_i \vee \neg a^j_i$, $i^j \vee a^{j+1}_i \vee \neg a^{j+1}_i$, $i^j \vee a^{j+1}_i \vee \neg a^j_i$;
- for all $q \in Q$, $(q^j \wedge i^j \wedge a^j_i) \Rightarrow nq(q,a)^{j+1} \wedge (i + dp(q,a))^{j+1} \wedge na(q,a)^{j+1}_i$ with the conclusion substituted by $\mathbf{false}$ when $i + dp(q,a) \notin [0, p(n)]$.

These subformulas ensure that, given an interpretation $\nu \models \varphi_{\mathcal{M},w}$,
for all $j < p(n)$, if $\mathbf{c}^j_\nu$ is defined then $\mathbf{c}^{j+1}_\nu$ is defined and $\mathbf{c}^j_\nu \rightarrow_{\mathcal{M}} \mathbf{c}^{j+1}_\nu$.

• Thus if $\nu \models \varphi_{\mathcal{M},w}$ then $\mathbf{c}^0_\nu, \ldots, \mathbf{c}^{p(n)}_\nu$ are all defined and correspond to an accepting run for w.

• Conversely assume $\mathbf{c}^0, \ldots, \mathbf{c}^{p(n)}$ is an accepting run.

Then ν, defined for all j by: $\nu(\mathbf{s}^j) = \nu_{\mathbf{c}^j}(\mathbf{s}^j)$, satisfies φ.

HORNSAT is PTIME-hard (3)

The construction of $\varphi_{\mathcal{M},w}$ is performed in LOGSPACE.

For instance to generate the subformulas,
for all $j < p(n)$, $i \leq p(n)$, $q \in Q$, $a \in \Sigma$:

$$(q^j \wedge i^j \wedge a_i^j) \Rightarrow nq(q,a)^{j+1} \wedge (i + dp(q,a))^{j+1} \wedge na(q,a)_i^{j+1}$$

The algorithm consists in four nested loops where:

- The sizes of i and j belong to $O(\log(p(n))) = O(\log(n))$;
- The sizes of q and a belong to $O(1)$.

3HORNSAT is PTIME-hard

Let φ be a HNF formula. Build φ' as follows. For all clause ψ:

- If $\psi = (p_1 \wedge \cdots \wedge p_k) \Rightarrow q$ then
  φ' has clauses $(p_1 \wedge p_2) \Rightarrow c_2$, $(c_2 \wedge p_3) \Rightarrow c_3$, $\ldots$, $(c_{k-1} \wedge p_k) \Rightarrow q$;
- If $\psi = (p_1 \wedge \cdots \wedge p_k) \Rightarrow \mathbf{false}$ then
  φ' has clauses $(p_1 \wedge p_2) \Rightarrow c_2$, $(c_2 \wedge p_3) \Rightarrow c_3$, $\ldots$, $(c_{k-1} \wedge p_k) \Rightarrow \mathbf{false}$.

where $c_2, \ldots, c_{k-1}$ are new propositions.

Sketch of proof.

Let $\psi = (p_1 \wedge \cdots \wedge p_k) \Rightarrow q$.

• Assume $\nu(\varphi) = \mathbf{true}$. Extend ν to ν' as follows.

If $\nu(H_\psi) = \mathbf{true}$ then $\nu'(c_2) = \cdots = \nu'(c_k) = \mathbf{true}$.

Otherwise let i be the first i with $\nu(p_i) = \mathbf{false}$

If $i \leq 2$ then $\nu'(c_2) = \cdots = \nu'(c_k) = \mathbf{false}$

else for $j < i$ then $\nu(c_j) = \mathbf{true}$ and for $j \geq i$, $\nu(c_j) = \mathbf{false}$

• Assume $\nu'(\varphi') = \mathbf{true}$.

If for some i, $\nu'(p_i) = \mathbf{false}$ then $\nu'(\psi) = \mathbf{true}$.

Otherwise by induction, for all i, $\nu'(c_i) = \mathbf{true}$ implying $\nu'(q) = \mathbf{true}$.

Solving the Closure Problem

Let $(G, \bullet)$ be a finite set equipped with a binary law.

Let $H \subseteq G$, the *closure* of H denoted $Cl(H)$ is the smallest set such that:

- $H \subseteq Cl(H)$;
- for all $a, b \in Cl(H)$, $a \bullet b \in Cl(H)$.

The *closure problem* asks given $(G, \bullet)$, $H \subseteq G$ and $g \in G$ whether $g \in Cl(H)$.

$CH \leftarrow H$
Repeat
  If $g \in CH$ **then return** $\mathbf{true}$
  $done \leftarrow \mathbf{true}$
  For $g', g'' \in CH$ **do**
    If $g' \bullet g'' \notin CH$ **then**
      $CH \leftarrow CH \cup \{g' \bullet g''\}$; $done \leftarrow \mathbf{false}$
Until $done$
return $\mathbf{false}$

The Closure Problem is PTIME-hard (1)

One reduces 3HORNSAT to the closure problem.

Let φ be a formula in 3HNF.

Let $\psi = \bigvee_{i \in I} \ell_i$ be a clause. The *components* of ψ are $\{\ell_i\}_{i \in J}$ with $J \subseteq I$.

The reduction

The items of G are the components of clauses of φ and a special item \$.

There are at most eight components per original clause.

The commutative law $\bullet$ is defined as follows.

- If $u = \{p\}$ and $v = \{\neg p\} \cup C$ then $u \bullet v = C$;
  (if $v = \{\neg p\}$ then $C = \emptyset$)
- Otherwise $u \bullet v =$ \$.

$g = \emptyset$ and $H = \{\{\ell_i\}_{i \in I} \mid \bigvee_{i \in I} \ell_i$ is a clause of $\varphi\}$.

The Closure Problem is PTIME-hard (2)

Correctness of the reduction.

• Assume there exists ν with $\nu(\varphi) = \mathbf{true}$.

By induction, for all $\{\ell_i\}_{i \in J} \in Cl(H)$, $\nu(\bigvee_{i \in J} \ell_i) = \mathbf{true}$.

Thus the empty set cannot belong to $Cl(H)$.

• Assume the empty set does not belong to $Cl(H)$.

Define $\nu(p) = \mathbf{true}$ iff $\{p\} \in Cl(H)$.

◦ Let $\psi = \neg p_1 \vee \neg p_2 \vee \neg p_3$ be a clause of φ with $\nu(\psi) = \mathbf{false}$.

Then $\{p_1\}$, $\{p_2\}$ and $\{p_3\}$ belong to $Cl(H)$.

Since $\{p_1\} \bullet (\{p_2\} \bullet (\{p_3\} \bullet \{\neg p_1, \neg p_2, \neg p_3\}))$ is the empty set, this yields a contradiction.

◦ Let $\psi = \neg p_1 \vee \neg p_2 \vee q$ be a clause of φ with $\nu(\psi) = \mathbf{false}$.

Then $\{p_1\}$ and $\{p_2\}$ belong to $Cl(H)$ and $\{q\}$ does not belong to $Cl(H)$.

Since $\{p_1\} \bullet (\{p_2\} \bullet \{\neg p_1, \neg p_2, q\}) = \{q\}$.

So $\{q\} \in Cl(H)$ yielding a contradiction.

Why this reduction does not work for 3SAT?

Reachability in non Deterministic Graph

A *non deterministic graph* $G = (V, A)$ is defined by vertices V and triples $A \subseteq V^3$.

Interpretation. From u, *choosing* $(u, v, w) \in A$ leads to either v or w.

Let $W \subseteq V$. Then $Reach(W)$ is the smallest set such that:

- $W \subseteq Reach(W)$;
- For all (u, v, w), if $\{v, w\} \subseteq Reach(W)$ then $u \in Reach(W)$.

Given G, W and $s \in V$, the *reachability problem* asks whether $s \in Reach(W)$.

$Z \leftarrow W$
Repeat
  If $s \in Z$ **then return true**
  $done \leftarrow$ **true**
  For $(u, v, w) \in A$ **do**
    If $u \notin Z \wedge \{v, w\} \subseteq Z$ **then**
      $Z \leftarrow Z \cup \{u\}$; $done \leftarrow$ **false**
Until $done$
return false

Reduction from the closure problem. $(u, v, w) \in A$ iff $u = v \bullet w$.

Context-Free Grammars

A context-free grammar G is defined by:

- a terminal alphabet Σ;
- a non terminal alphabet Γ including an axiom S;
- a set of production rules R where $T \to w \in R$ implies $T \in \Gamma$ and $w \in (\Sigma \cup \Gamma)^*$

The languages $\{L(G,T)\}_{T\in\Gamma}$ are mutually inductively defined by:

- If $T \to w$ is a rule with $w \in \Sigma^*$ then $w \in L(G,T)$;
- Si $T \to w_0 T_1 w_1 \dots T_n w_n$ is a rule
  with $w_i \in \Sigma^*$, $T_i \in \Gamma$ and $u_i \in L(G,T_i)$
  then $w_0 u_1 \dots u_n w_n \in L(G,T)$.

A Context-Free Grammar

A grammar for arithmetic expressions.

$\Sigma = \{(,), +, \cdot, 0, \ldots, 9\}$ and $\Gamma = \{E, T, F\}$ with E the axiom.

An expression is a sum of terms.

$E \to T \mid E + T$

A term is a product of factors.

$T \to F \mid T \cdot F$

A factor is a digit or an expression in parentheses.

$F \to 0 \mid 1 \mid 2 \mid \ldots \mid 9 \mid (E)$

To obtain $(3 + 5) \cdot 2$:

$\underline{E} \to \underline{T} \to \underline{T} \cdot F \to F \cdot \underline{F} \to \underline{F} \cdot 2 \to (\underline{E}) \cdot 2 \to (T + \underline{E}) \cdot 2 \to (T + \underline{T}) \cdot 2$

$\to (T + \underline{F}) \cdot 2 \to (\underline{T} + 5) \cdot 2 \to (\underline{F} + 5) \cdot 2 \to (3 + 5) \cdot 2$

The Emptiness Problem Belongs to PTIME

The *emptiness problem* asks whether $L(G, S) = \emptyset$.

Let $Prod(G) = \{T \mid L(G, T) \neq \emptyset\}$.

$Prod \leftarrow \emptyset$
Repeat
  If $S \in Prod$ **then return true**
  $done \leftarrow$ **true**
  For $T \rightarrow w_0 T_1 w_1 \ldots T_n w_n \in R$ **do**
    If $T \notin Prod \wedge \{T_1, \ldots, T_n\} \subseteq Prod$ **then**
      $Prod \leftarrow Prod \cup \{T\}$; $done \leftarrow$ **false**
Until $done$
return false

The Emptiness Problem is PTIME-hard

Reduction of the closure problem.

Let $((G, \bullet), g, H)$ be a closure problem.

Then the context-free grammar is built as follows.

- Σ is irrelevant *(and can be empty)*.
- $\Gamma = G$.
- The axiom is g.
- The production rules are defined by:
  $\{u \to vw \mid u = v \bullet w\} \bigcup \{u \to \varepsilon \mid u \in H\}$.

The correction is immediate.

Boolean Circuits

A *boolean circuit* is a acyclic directed graph where vertices are called *gates*.

- A gate has an *identifier*
  defined according to some topological sort of the graph;
- A gate has some incoming edges called the *inputs*;
- In addition a gate has a *type*: $\mathbf{false}$, $\mathbf{true}$, $\wedge$, $\vee$.

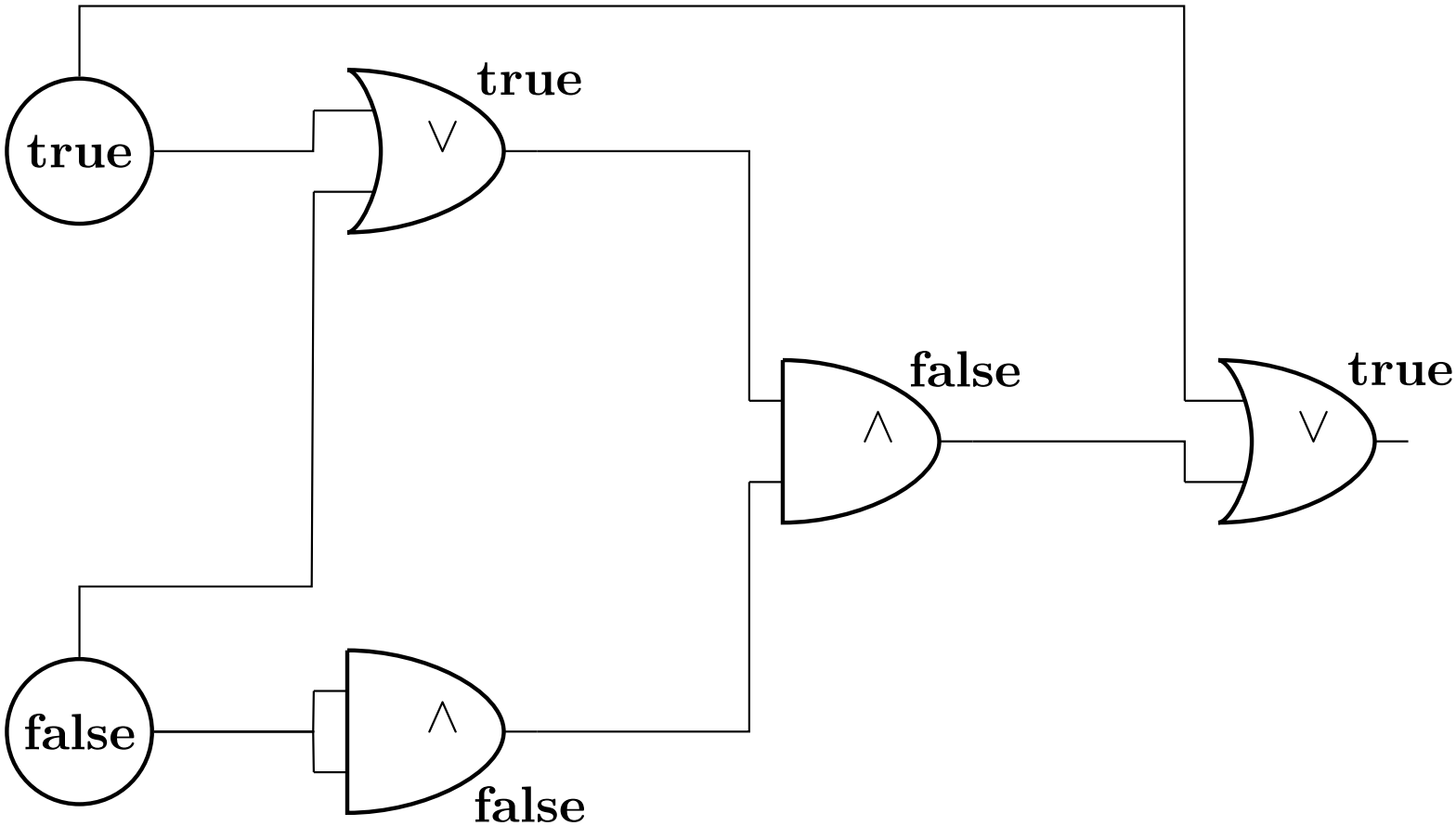
A circuit has the following restrictions:

- Gate 0 is the single $\mathbf{false}$-typed gate and has no input;
- Gate 1 is the single $\mathbf{true}$-typed gate and has no input;
- All other gates have at least one input.

The boolean values Val of the gates are inductively defined by:

- $Val(0) = \mathbf{false}$, $Val(1) = \mathbf{true}$;
- Let g be a $\vee$-gate with inputs $\{g_i\}_{i \leq k}$.
  If all $Val(g_i)$ are defined then $Val(g) = \bigvee_{i \leq k} Val(g_i)$
- Let g be a $\wedge$-gate with inputs $\{g_i\}_{i \leq k}$.
  If all $Val(g_i)$ are defined then $Val(g) = \bigwedge_{i \leq k} Val(g_i)$

A Boolean Circuit



The Circuit Value Problem

Let $\mathcal{C}$ be a circuit and out be a gate of $\mathcal{C}$.
The *circuit value problem* asks for $Val(out)$.

We assume that the circuit representation gates are ordered w.r.t. some topological sort of the graph.

A polynomial time algorithm for the circuit value problem.

For $g \in \mathcal{C}$ **do**
  Case $tp(g)$ **of**
    false : $Val[g] \leftarrow$ **false**;
    true : $Val[g] \leftarrow$ **true**;
    $\vee : Val[g] \leftarrow \bigvee_{h \in In(g)} Val[h]$;
    $\wedge : Val[g] \leftarrow \bigwedge_{h \in In(g)} Val[h]$;

It remains polynomial time with other gate types: $\neg$, $\Rightarrow$, etc.

From Closure to Circuits (1)

Let $((G, \bullet), H, g)$ be a closure problem with $n = |G|$.

Let $Cl_i(H)$ defined for i from 0 to n:

$$Cl_0(H) = H \text{ and } Cl_{i+1}(H) = \{x \mid \exists y, z \in Cl_i(H)\ x = y \bullet z\} \cup Cl_i(H).$$

If $Cl_{i+1}(H) = Cl_i(H)$ then for all $j > i$, $Cl_j(H) = Cl_i(H)$.
Thus $Cl_n(H) = Cl(H)$ where $n = |G|$.

The circuit has the following gates.

- For all $0 \leq i \leq n$ and $x \in G$, $\vee$-gates (i, x) evaluate to **true** iff $x \in Cl_i(H)$;
- For all $1 \leq i \leq n$ and $y, z \in G$ $\wedge$-gates (i, y, z) evaluate to **true** iff $\{y, z\} \subseteq Cl_{i-1}(H)$.

Gate **out** is (n, g).

The inputs of the gates are:

- For all x, $(0, x)$ has an input : **true** (resp. **false**) if $x \in H$ (resp. $x \notin H$);
- For all x and $i \geq 1$, (i, x) has the following inputs : $(i-1, x)$ and all (i, y, z) such that $x = y \bullet z$;
- For all y, z, (i, y, z) has two inputs : $(i-1, y)$ and $(i-1, z)$.

From Closure to Circuits (2)

Let $((G, \bullet), H, g)$ be a closure problem with $n = |G|$.

The reduction algorithm.

Write$(id : 0)$; Write$(tp : \mathbf{false})$; Write$(id : 1)$; Write$(tp : \mathbf{true})$
For $x \in G$ **do**
  Write$(id : (0, x))$; Write$(tp : \vee)$
  If $x \in H$ **then** Write$(in : 1)$ **else** Write$(in : 0)$
For i **from** 1 **to** n **do**
  For $y, z \in G$ **do**
    Write$(id : (i, y, z))$; Write$(tp : \wedge)$; Write$(in : (i-1, y))$; Write$(in : (i-1, z))$
  For $x \in G$ **do**
    Write$(id : (i, x))$; Write$(tp : \vee)$; Write$(in : (i-1, x))$
    For $y, z \in G$ **do**
      If $x = y \bullet z$ **then** Write$(in : (i, y, z))$
Write$(out : (n, g))$

Plan

The Reachability Problem in Graphs

Let $G = (V, E)$ be a graph and s, t two vertices.

The *reachability problem* asks whether there is a path in G from s to t.

Solving the reachability problem.

$u \leftarrow s$
$cpt \leftarrow 0$
Repeat
  If $u = t$ **return true**
  If there is no $(u, v) \in E$ **return false**
  guess $(u, v) \in E$;
  $u \leftarrow v$
  $cpt \leftarrow cpt + 1$
Until $cpt = n$
return false

This algorithm operates non deterministically in logarithmic space.

Reachability Problem is NLOGSPACE-hard

Let $\mathcal{M}$ be a NTM operating in space $\log(n)$ on a word w with size n.

W.l.o.g there is a single accepting configuration.

A configuration is given by the position of the two heads ($O(\log(n))$ and $O(\log(\log(n)))$), the state ($O(1)$), and the working tape content ($O(\log(n))$).

The input word is not required.

The output of the reduction.

The reachability problem has the configuration graph and the initial and accepting configurations as inputs.

The space complexity of the reduction.

- The reduction algorithm writes the number of configurations which can be computed in $O(\log(n))$.
- Then it performs a loop indexed by configuration **c** to build the edges outgoing **c**.
- Thus it needs the space for two configurations (and some overhead) in $O(\log(n))$.

Summary for Language Problems

	Emptiness	Universality
Non Deterministic Automata	NLOGSPACE-complete	PSPACE-complete
Context-Free Grammars	PTIME-complete	Undecidable

The undecidability result will be proven in the formal language lectures.

Non Deterministic Computations

A *computing* NTM $\mathcal{M}$ is a NTM with an output tape.

$\mathcal{M}$ (non deterministically) computes function f if for all word w:

- For all accepting runs on w, the output tape contains $f(w)$;
- There is at least one accepting run on w.

Example.

Assume that in w there is an unknown letter that occurs more than $\frac{|w|}{2}$.

Guess $a \in \Sigma$
$cpt \leftarrow 0$
For i **from** 1 **to** n **do**
  If $w[i] = a$ **then** $cpt \leftarrow cpt + 1$
If $cpt > \frac{|w|}{2}$ **then return** cpt **else return reject**

The Number of Reachable Configurations

Let $\mathcal{M}$ be a NTM that operates in $\log(n)$ on a word w of size n.
Then there is a NTM $\mathcal{M}'$ that operates in $O(\log(n))$
counting the number of reachable configurations by $\mathcal{M}$ on w.

Compute the size of a configuration of $\mathcal{M}$ on word w; $d \leftarrow 0$; $N \leftarrow 1$
Repeat
  $OldN \leftarrow N$; $N \leftarrow 0$ % *$oldN$ is the ♯ of configurations reachable with at most d steps*
  For $current \in Conf_{\mathcal{M},w}$ **do** % *check if current is reachable with at most $d+1$ steps*
    $acc \leftarrow$ **false**; $cpt \leftarrow 0$ % *cpt controls the validity of the guesses*
    For $local \in Conf_{\mathcal{M},w}$ **do** % *check if local is reachable with at most d steps*
      Guess a run σ from *init* to *local* with at most d steps
      without computing the configuration graph.
      If σ is found **then**
        $cpt \leftarrow cpt + 1$
        If $local = current$ **or** $local \rightarrow_{\mathcal{M}} current$ **then**
          $N \leftarrow N + 1$; $acc \leftarrow$ **true**; **break**
    End For
    If not acc **and** $cpt < OldN$ **then reject** % *some guesses were wrong*
  End For
  $d \leftarrow d + 1$
Until $N = OldN$
Return N

NLOGSPACE=coNLOGSPACE

Let $\mathcal{M}$ be a NTM that operates in $\log(n)$ on a word w of size n. Then there is a NTM $\mathcal{M}'$ that operates in $O(\log(n))$ that accepts w iff $\mathcal{M}$ rejects w.

Compute the size of a configuration of $\mathcal{M}$ on word w
$d \leftarrow 0$; $N \leftarrow 1$
Repeat
  $OldN \leftarrow N$; $N \leftarrow 0$
  For $current \in Conf_{\mathcal{M},w}$ **do**
    $acc \leftarrow$ **false**; $cpt \leftarrow 0$
    For $local \in Conf_{\mathcal{M},w}$ **do**
      Guess a run σ from *init* to *local* with at most d steps
      If σ is found **then**
        $cpt \leftarrow cpt + 1$
        If $local = current$ **or** $local \rightarrow_{\mathcal{M}} current$ **then**
          If $current$ is accepting **then reject**
          $N \leftarrow N + 1$; $acc \leftarrow$ **true**; **break**
    End For
    If not acc **and** $cpt < OldN$ **then reject**
  End For
  $d \leftarrow d + 1$
Until $N = OldN$
accept

Plan

Deterministic Space Hierarchy (1)

Let $f(n) \geq \log(n)$ and $g(n) \geq \log(n)$ two measure functions fulfilling:

$$\liminf_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Then there exists a language L accepted by a DTM operating on w in space $g(|w|)$ but by no DTM operating on w in space $f(|w|)$.

Thus, NLOGSPACE $\subseteq$ LOG^{2}SPACE $\subsetneq$ PSPACE $\subsetneq$ EXPSPACE.

We consider representations of DTM such that for all $\mathcal{M}$, there exists $n_{\mathcal{M}}$ with:

- for all $n \geq n_{\mathcal{M}}$, there exists $\widetilde{\mathcal{M}}_n$ a representation of $\mathcal{M}$ of size n;
- for instance pick an arbitrary representation $\widehat{\mathcal{M}}$. Then $n_{\mathcal{M}} = |\widehat{\mathcal{M}}| + 1$;
- and $\widetilde{\mathcal{M}}_n = 1^{n - n_{\mathcal{M}}} 0 \widehat{\mathcal{M}}$.

Deterministic Space Hierarchy (2)

Let $\mathcal{U}$ be a DTM that takes as input w and works as follows.

It has a special working tape managing a counter.

Let $n = |w|$, then:

- $\mathcal{U}$ computes $g(n)$.
- $\mathcal{U}$ marks all its working tapes at position $g(n)$ with some symbol so that it (possibly) later stops and rejects when reading it.
- If w is not the representation of a DTM then $\mathcal{U}$ rejects.

Otherwise assume $w = \widetilde{\mathcal{M}_k}$ for some k with $n_q = |Q_{\mathcal{M}}|$, n_t the number of working tapes of $\mathcal{M}$ and $n_a = |\Sigma_{\mathcal{M}}|$.

- $\mathcal{U}$ initializes its counter to $n_q f(n)^{n_t} \lceil \log(n+1) \rceil n_a^{n_t f(n)}$.
- $\mathcal{U}$ simulates the run of $\mathcal{M}$ on w decrementing its counter after any step.
- $\mathcal{U}$ stops and rejects if the counter equals 0.
- When the simulation ends, $\mathcal{U}$ rejects iff $\mathcal{M}$ accepts.

Deterministic Space Hierarchy (3)

• The initial value of this counter strictly bounds the maximal number of steps that $\mathcal{M}$ can perform if it operates in space $f(n)$.

Given a DTM $\mathcal{M}$, this counter uses space $O(f(n))$.

• Let L be the language accepted par $\mathcal{U}$.

By construction, $\mathcal{U}$ operates in space $g(n)$.

• Assume that L is accepted by a DTM $\mathcal{M}$ operating in space $f(n)$.

The simulation of $\mathcal{M}$ requires space $O(f(n))$.

Using $\liminf_{n\to\infty} \frac{f(n)}{g(n)} = 0$, one selects some k enough large such that on $\widetilde{\mathcal{M}}_k$, $\mathcal{U}$ achieves the simulation of $\mathcal{M}$.

Thus $\mathcal{M}$ accepts $\widetilde{\mathcal{M}}_k$ iff $\mathcal{U}$ rejects it, yielding a contradiction.

Deterministic Time Hierarchy

Let $f(n) \geq n$ and $g(n) \geq n$ two measure functions fulfilling:

$$\liminf_{n \to \infty} \frac{f(n) \log(f(n))}{g(n)} = 0 \text{ and } \lim_{n \to \infty} \frac{g(n)}{n} = \infty$$

Then there exists a language L accepted by a DTM operating on w in time $g(|w|)$ but by no DTM operating on w in time $f(|w|)$.

Non Deterministic Time Hierarchy

Let $f(n) \geq n$ and $g(n) \geq n$ two measure functions fulfilling:

$$\lim_{n \to \infty} \frac{f(n+1)}{g(n)} = 0 \text{ and } \lim_{n \to \infty} \frac{g(n)}{n} = \infty$$

Then there exists a language L accepted by a NTM operating on w in time $g(|w|)$ but by no NTM operating on w in time $f(|w|)$.