Complexity Theory
WS 2009/10

Prof. Dr. Erich Grädel

Mathematische Grundlagen der Informatik
RWTH Aachen

Contents

1 Deterministic Turing Machines and Complexity Classes 1
1.1 Turing machines 1
1.2 Time and space complexity classes 4
1.3 Speed-up and space compression 7
1.4 The Gap Theorem 9
1.5 The Hierarchy Theorems 11

2 Nondeterministic complexity classes 17
2.1 Nondeterministic Turing machines 17
2.2 Elementary properties of nondeterministic classes 19
2.3 The Theorem of Immerman and Szelepcsényi 21

3 Completeness 27
3.1 Reductions 27
3.2 NP-complete problems: Sat and variants 28
3.3 P-complete problems 34
3.4 NLogspace-complete problems 38
3.5 A Pspace-complete problem 42

4 Oracles and the polynomial hierarchy 47
4.1 Oracle Turing machines 47
4.2 The polynomial hierarchy 49
4.3 Relativisations 52

5 Alternating Complexity Classes 55
5.1 Complexity Classes 56
5.2 Alternating Versus Deterministic Complexity 57
5.3 Alternating Logarithmic Time 61

6 Complexity Theory for Probabilistic Algorithms 63
6.1 Examples of probabilistic algorithms 63
6.2 Probabilistic complexity classes and Turing machines 72
6.3 Probabilistic proof systems and Arthur-Merlin games 81

3 Completeness

3.1 Reductions

Definition 3.1. Let $A \subseteq \Sigma^*, B \subseteq \Gamma^*$ be two languages. A function $f : \Sigma^* \to \Gamma^*$ is called a *reduction from A to B* if, for all $x \in \Sigma^*$, $x \in A \Leftrightarrow f(x) \in B$. To put it differently: If $f(A) \subseteq B$ and $f(\bar{A}) = f(\Sigma^* \setminus A) \subseteq (\Gamma^* \setminus B) = \bar{B}$. Hence, a reduction from A to B is also a reduction from $\bar{A}$ to $\bar{B}$.

Let $\mathcal{C}$ be a complexity class (of decision problems). A class of functions $\mathcal{F}$ provides an appropriate notion of *reducibility* for $\mathcal{C}$ if

- $\mathcal{F}$ is closed under composition, i.e.,

$$\begin{array}{ll} \text{if} & f : \Sigma^* \to \Gamma^* \in \mathcal{F} \\ \text{and} & g : \Gamma^* \to \Delta^* \in \mathcal{F}, \\ \text{then} & g \circ f : \Sigma^* \to \Delta^* \in \mathcal{F}. \end{array}$$

- $\mathcal{C}$ is closed under $\mathcal{F}$: If $B \in \mathcal{C}$ and $f \in \mathcal{F}$ is a reduction from A to B, then $A \in \mathcal{C}$.

For two problems A, B we say that A is $\mathcal{F}$-deducible to B if there is a function $f \in \mathcal{F}$ that is a reduction from A to B.

Notation: $A \leq_{\mathcal{F}} B$.

Definition 3.2. A problem B is *$\mathcal{C}$-hard under $\mathcal{F}$* if all problems $A \in \mathcal{C}$ are $\mathcal{F}$-reducible to B ($A \in \mathcal{C} \Rightarrow A \leq_{\mathcal{F}} B$).

A problem B is *$\mathcal{C}$-complete (under $\mathcal{F}$)* if $B \in \mathcal{C}$ and B is $\mathcal{C}$-hard (under $\mathcal{F}$).

The most important notions of reducibility in complexity theory are

- $\leq_p$: polynomial-time reducibility (given by the class of all polynomial-time computable functions)
- $\leq_{\log}$: log-space reducibility (given by the class of functions computable with logarithmic space)

Closure under composition for polynomial-time reductions is easy to show. If

$$f : \Sigma^* \to \Gamma^* \text{ is computable in time } O(n^k) \text{ by } M_f \text{ and}$$
$$g : \Gamma^* \to \Delta^* \text{ is computable in time } O(n^m) \text{ by } M_g,$$

then there are constants c, d such that $g \circ f : \Sigma^* \to \Delta^*$ is computable in time $c \cdot n^k + d(c \cdot n^k)^m = O(n^{k+m})$ by a machine that writes the output of M_f (whose length is bounded by $c \cdot n^k$) to a working tape and use it as the input for M_g.

In case of log-space reductions this trivial composition does not work since $f(x)$ can have polynomial length in $|x|$ and hence cannot be completely written to the logarithmically bounded work tape. However, we can use a modified machine M'_f that computes, for an input x and a position i, the i-th symbol of the output $f(x)$. Thus, $g(f(x))$ can be computed by simulating M_g, such that whenever it accesses the i-th symbol of the input, M'_f is called to compute it. The computation of M'_f on (x, i) can be done in logarithmic space (space needed for computation and for the counter i: $\log(n^k)$) the symbol $f(x, i)$ written to the tape needs only constant space. Furthermore, the computation of M_g only needs space logarithmic in the input length as $c \cdot \log(|f(x)|) = c \cdot \log(|x|^k) = c \cdot k \cdot \log(|x|) = O(\log(|x|))$.

3.2 NP-complete problems: Sat and variants

NP can be defined as the class of problems decidable in nondeterministic polynomial time:

Definition 3.3. $\mathrm{NP} = \bigcup_{d \in \mathbb{N}} \textsc{Ntime}(n^d)$.

A different, in some sense more instructive, definition of NP is the class of problems with polynomially-time verifiable solutions:

Definition 3.4. $A \in \mathrm{NP}$ if, and only if, there is a problem $B \in \mathrm{P}$ and a polynomial p such that $A = \{x : \exists y(|y| \leq p(|x|) \wedge x\#y \in B)\}$.

The two definitions coincide: If A has polynomially verifiable solutions via $B \in \mathrm{P}$ and a polynomial p, then the following algorithm decides A in nondeterministic polynomial time:

Input: x
guess y with $|y| < p(n)$
check whether $x\#y \in B$
if *answer is yes* **then** accept **else** reject

Conversely, let $A \in \textsc{Ntime}(p(n))$, and M be a p-time bounded NTM that decides A. A computation of M on some input of length n is a sequence of at most $p(n)$ configurations of length $\leq p(n)$. Therefore, a computation of M can be described by a $p(n) \times p(n)$ table with entries from $Q \times \Sigma \cup \Sigma$ and thus by a word of length $p^2(n)$. Set

$$B = \{x\#y : y \text{ accepting computation of } M \text{ on } x\}.$$

We can easily see that $B \in \mathrm{P}$, and $x \in L$ if, and only if, there exists y with $|y| \leq p^2(n)$ such that $x\#y \in B$. Therefore, $L \in \mathrm{NP}$.

Theorem 3.5.

(i) $\mathrm{P} \subseteq \mathrm{NP}$.
(ii) $A \leq_{\mathrm{p}} B, B \in \mathrm{NP} \Rightarrow A \in \mathrm{NP}$.

Clearly NP is closed under polynomial-time reductions:

$$B \in \mathrm{NP}, A \leq_{\mathrm{p}} B \quad \Longrightarrow \quad A \in \mathrm{NP}.$$

B is NP-complete if

(1) $B \in \mathrm{NP}$ and
(2) $A \leq_{\mathrm{p}} B$ for all $A \in \mathrm{NP}$.

The most important open problem in complexity theory is **Cook's hypothesis**: $\mathrm{P} \neq \mathrm{NP}$.

For every NP-complete problem B we have:

$$P \neq \mathrm{NP} \quad \Longleftrightarrow \quad B \notin \mathrm{P}.$$

We recall the basics of propositional logic. Let $\tau = \{X_i : i \in \mathbb{N}\}$ be a finite set of propositional variables. The set AL of *propositional logic formulae* is defined inductively:

(1) $0, 1 \in \mathrm{PL}$ (the Boolean constants are formulae).
(2) $\tau \subseteq \mathrm{PL}$ (every propositional variable is a formula).
(3) If $\psi, \varphi \in \mathrm{PL}$, then also $\neg\psi$, $(\psi \wedge \varphi)$, $(\psi \vee \varphi)$ and $(\psi \rightarrow \varphi)$ are formulae in PL.

A *(propositional) interpretation* is a map $\mathfrak{I} : \sigma \rightarrow \{0,1\}$ for some $\sigma \subseteq \tau$. It is *suitable* for a formula $\psi \in \mathrm{PL}$ if $\tau(\psi) \subseteq \sigma$. Every interpretation $\mathfrak{I}$ that is suitable to ψ defines a logical value $[\![\psi]\!]^{\mathfrak{I}} \in \{0,1\}$ with the following definitions:

(1) $[\![0]\!]^{\mathfrak{I}} := 0$, $[\![1]\!]^{\mathfrak{I}} := 1$.
(2) $[\![X]\!]^{\mathfrak{I}} := \mathfrak{I}(X)$ for $X \in \sigma$.
(3) $[\![\neg\psi]\!]^{\mathfrak{I}} := 1 - [\![\psi]\!]^{\mathfrak{I}}$.
(4) $[\![\psi \wedge \varphi]\!]^{\mathfrak{I}} := \min([\![\psi]\!]^{\mathfrak{I}}, [\![\varphi]\!]^{\mathfrak{I}})$.
(5) $[\![\psi \vee \varphi]\!]^{\mathfrak{I}} := \max([\![\psi]\!]^{\mathfrak{I}}, [\![\varphi]\!]^{\mathfrak{I}})$.
(6) $[\![\psi \rightarrow \varphi]\!]^{\mathfrak{I}} := [\![\neg\psi \vee \varphi]\!]^{\mathfrak{I}}$.

A *model* of a formula $\psi \in \mathrm{PL}$ is an interpretation $\mathfrak{I}$ with $[\![\psi]\!]^{\mathfrak{I}} = 1$. Instead of $[\![\psi]\!]^{\mathfrak{I}} = 1$, we will write $\mathfrak{I} \models \psi$ and say $\mathfrak{I}$ *satisfies* ψ. A formula ψ is called *satisfiable* if a model for ψ exists. A formula ψ is called a *tautology* if every suitable interpretation for ψ is a model of ψ.

A formula ψ is obviously satisfiable iff $\neg\psi$ is not a tautology. Two formulae ψ and φ are called equivalent ($\psi \equiv \varphi$) if, for each $\mathfrak{I} : \tau(\psi) \cup \tau(\varphi) \rightarrow \{0,1\}$, we have $[\![\psi]\!]^{\mathfrak{I}} = [\![\varphi]\!]^{\mathfrak{I}}$. A formula φ follows from ψ (short, $\psi \models \varphi$) if, for every interpretation $\mathfrak{I} : \tau(\psi) \cup \tau(\varphi) \rightarrow \{0,1\}$ with $\mathfrak{I}(\psi) = 1$, $\mathfrak{I}(\varphi) = 1$ holds as well.

Comments. Usually, we omit unnecessary parentheses. As $\wedge$ and $\vee$ are semantically associative, we can use the following notations for conjunctions and disjunctions over $\{\psi_i \ : \ i \in I\}$: $\bigwedge_{i \in I} \psi_i$ respectively $\bigvee_{i \in I} \psi_i$. We fix the set of variables $\tau = \{X_i \ : \ i \in \mathbb{N}\}$ and encode X_i

by $X(\text{bin } i)$, i.e., a symbol X followed by the binary representation of the index i. This enables us to encode propositional logic formulae as words over a finite alphabet $\Sigma = \{X, 0, 1, \wedge, \vee, \neg, (,)\}$.

Definition 3.6. $\textsc{sat} := \{\psi \in \mathrm{PL} : \psi \text{ is satisfiable}\}$.

Theorem 3.7 (Cook, Levin)**.** $\textsc{sat}$ is NP-complete.

Proof. It is clear that $\textsc{sat}$ is in NP because

$$\{\psi\#\mathfrak{I} \mid \mathfrak{I} : \tau(\psi) \to \{0,1\}, \mathfrak{I} \models \psi\} \in \mathrm{P}.$$

Let A be some problem contained NP. We show that $A \leq_{\mathrm{p}} \textsc{sat}$. Let $M = (Q, \Sigma, q_0, F, \delta)$ be a nondeterministic 1-tape Turing machine deciding A in polynomial time $p(n)$ with $F = F^+ \cup F^-$. We assume that every computation of M ends in either an accepting or rejecting final configuration, i.e., C is a final configuration iff $\mathrm{Next}(C) = \varnothing$. Let $w = w_0 \cdots w_{n-1}$ be some input for M. We build a formula $\psi_w \in \mathrm{PL}$ that is satisfiable iff M accepts the input w.

Towards this, let ψ_w contain the following propositional variables:

- $X_{q,t}$ for $q \in Q, 0 \leq t \leq p(n)$,
- $Y_{a,i,t}$ for $a \in \Sigma, 0 \leq i, t \leq p(n)$,
- $Z_{i,t}$ for $0 \leq i, t \leq p(n)$,

with the following intended meaning:

- $X_{q,t}$: "at time t, M is in state q,"
- $Y_{a,i,t}$: "at time t, the symbol a is written on field i,"
- $Z_{i,t}$: "at time t, M is at position i."

Finally,

$$\psi_w := \textsc{start} \wedge \textsc{compute} \wedge \textsc{end}$$

with

$$\textsc{start} := X_{q_0,0} \wedge \bigwedge_{i=0}^{n-1} Y_{w_i,i,0} \wedge \bigwedge_{i=n}^{p(n)} Y_{\square,i,0} \wedge Z_{0,0}$$

$$\textsc{compute} := \textsc{nochange} \wedge \textsc{change}$$

$$\textsc{nochange} := \bigwedge_{t<p(n),a\in\Sigma,i\neq j} (Z_{i,t} \wedge Y_{a,j,t} \to Y_{a,j,t+1})$$

$$\textsc{change} := \bigwedge_{t<p(n),i,a,q} \Big((X_{q,t} \wedge Y_{a,i,t} \wedge Z_{i,t}) \to$$

$$\bigvee_{\substack{(q',b,m)\in\delta(q,a)\\ 0\leq i+m\leq p(n)}} (X_{q',t+1} \wedge Y_{b,i,t+1} \wedge Z_{i+m,t+1})\Big)$$

$$\textsc{end} := \bigwedge_{t\leq p(n),q\in F^-} \neg X_{q,t}$$

Here, START "encodes" the input configuration at time 0. NOCHANGE ensures that no changes are made to the field at the current position. CHANGE represents the transition function.

It is straightforward to see that the map $w \mapsto \psi_w$ is computable in polynomial time.

(1) Let $w \in L(M)$. Every computation of M induces an interpretation of the propositional variables $X_{q,t}, Y_{a,i,t}, Z_{i,t}$. An accepting computation of M on w induces an interpretation that satisfies ψ_w. Therefore, $\psi_w \in$ SAT.

(2) Let $C = (q, y, p)$ be some configuration of M, $t \leq p(n)$. Set

$$\textsc{conf}[C,t] := X_{q,t} \wedge \bigwedge_{i=0}^{p(n)} Y_{y_i,i,t} \wedge Z_{p,t}.$$

Please note that $\textsc{start} = \textsc{conf}[C_0(w), 0]$. Thus,

$$\psi_w \models \textsc{conf}[C_0(w), 0]$$

holds. For every non-final configuration C of M and all $t < p(n)$, we obtain (because of the subformula COMPUTE of ψ_w) :

$$\psi_w \wedge \textsc{conf}[C,t] \models \bigvee_{C'\in\mathrm{Next}(C)} \textsc{conf}[C', t+1].$$

(3) Let $\mathfrak{I}(\psi_w) = 1$. From (1) and (2) it follows that there is at least one computation $C_0(w) = C_0, C_1, \ldots, C_r = C_{\mathrm{end}}$ of M on w with

$r \leq p(n)$ such that $\mathfrak{I}(\textsc{conf}[C_t, t]) = 1$ for each $t = 0, \ldots, v$. Furthermore, $\psi_w \models \neg\textsc{conf}[C, t]$ holds for all rejecting final configurations C of M and all t because of the subformula END of ψ_w. Therefore, C_{end} is accepting, and M accepts the input w.

We have thus shown that $\psi_w \in \textsc{sat}$ if, and only if, $w \in A$. Q.E.D.

Remark. The reduction $w \mapsto \psi_w$ is particularly easy; it is computable with *logarithmic space*.

A consequence from Theorem 3.7 is that SAT is NP-complete via LOGSPACE-reductions.

Even though SAT is NP-complete, the satisfiability problem may still be polynomially solvable for some interesting formulae classes $S \subseteq \text{PL}$. We show that for certain classes $S \subseteq \text{PL}$, $S \cap \textsc{sat} \in \text{P}$ while in other cases $S \cap \textsc{sat}$ is NP-complete.

Reminder. A *literal* is a propositional variable or its negation. A formula $\psi \in \text{PL}$ is in *disjunctive normal form (DNF)* if it is of the form $\psi = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} Y_{ij}$, where Y_{ij} are literals. A formula ψ is in *conjunctive normal form (CNF)* if it has the form $\psi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} Y_{ij}$. A disjunction $\bigvee_j Y_{ij}$ is also called *clause*. Every formula $\psi \in \text{PL}$ is equivalent to a formula ψ_D in DNF and to a formula ψ_C in CNF.

$$\psi \equiv \psi_D := \bigvee_{\substack{\mathfrak{I}:\tau(\psi)\to\{0,1\} \\ \mathfrak{I}(\psi)=1}} \bigwedge_{X\in\tau(\psi)} X^{\mathfrak{I}}$$

with

$$X^{\mathfrak{I}} = \begin{cases} X & \text{if } \mathfrak{I}(X) = 1 \\ \neg X & \text{if } \mathfrak{I}(X) = 0\,, \end{cases}$$

and analogously for CNF.

The translations $\psi \mapsto \psi_D, \psi \mapsto \psi_C$ are computable but generally not in polynomial time. The formulae ψ_D and ψ_C can be exponentially longer than ψ as there are $2^{|\tau(\psi)|}$ possible maps $\mathfrak{I} : \tau(\psi) \to \{0,1\}$.

$$\textsc{sat-dnf} := \{\psi \text{ in DNF} \;:\; \psi \text{ satisfiable}\} \text{ and}$$

$$\textsc{sat-cnf} := \{\psi \text{ in CNF} \;:\; \psi \text{ satisfiable}\}$$

denote the set of all satisfiable formulae in DNF and CNF, respectively.

Theorem 3.8. $\textsc{sat-dnf} \in \textsc{Logspace} \subseteq \mathrm{P}$.

Proof. $\psi = \bigvee_i \bigwedge_{j=1}^{m_i} Y_{ij}$ is satisfiable iff there is an i such that no variable in $\{Y_{ij} : j = 1, \dots, m_i\}$ occurs both positively and negatively. Q.E.D.

Theorem 3.9. $\textsc{sat-cnf}$ is NP-complete via $\textsc{Logspace}$-reduction.

Proof. The proof follows from the one of Theorem 3.7. Consider the formula

$$\psi_w = \textsc{start} \wedge \textsc{compute} \wedge \textsc{end}\,.$$

From the proof, we see that $\textsc{start}$ and $\textsc{end}$ are already in CNF. The same is true for the subformula $\textsc{nochange}$ of $\textsc{compute}$, only $\textsc{change}$ is left. $\textsc{change}$ is a conjunction of formulae that have the form

$$\alpha : X \wedge Y \wedge Z \to \bigvee_{j=1}^{r} X_j \wedge Y_j \wedge Z_j.$$

Here, $r \leq \max_{(q,a)} |\delta(q,a)|$ is fixed, i.e., independent of n and w. But we have

$$\alpha \equiv (X \wedge Y \wedge Z \to \bigvee_{j=1}^{r} U_j) \wedge \bigwedge_{j=1}^{r} (U_j \to X_j) \wedge (U_j \to Y_j) \wedge (U_j \to Z_j)).$$

Therefore, $A \leq_{\log} \textsc{sat-cnf}$ for each $A \in \mathrm{NP}$. Q.E.D.

3.3 P-complete problems

A (propositional) *Horn formula* is a formula $\psi = \bigwedge_i \bigvee_j Y_{i_j}$ in CNF where every disjunction $\bigvee_j Y_j$ contains at most one positive literal. Horn formulae can also be written as implications by the following equivalences:

$$\neg X_1 \vee \dots \vee \neg X_k \vee X \equiv (X_1 \wedge \dots \wedge X_k) \to X,$$
$$\neg X_1 \vee \dots \vee \neg X_k \equiv (X_1 \wedge \dots \wedge X_k) \to 0.$$

Let HORN-SAT $= \{\psi \in \mathrm{PL} \ : \ \psi$ a satisfiable Horn formula$\}$. We know from mathematical logic:

Theorem 3.10. HORN-SAT $\in$ P.

This follows, e.g., by unit resolution or the marking algorithm.

Theorem 3.11. HORN-SAT is P-complete via logspace reduction.

Proof. Let $A \in P$ and M a deterministic 1-tape Turing machine, that decides A in time $p(n)$. Looking at the reduction $w \mapsto \psi_w$ from the proof of Theorem 3.7, we see that the formulae START, NOCHANGE and END are already Horn formulae. Since M was chosen to be deterministic, i.e., $|\delta(q,a)| = 1$, CHANGE takes the form $(X \wedge Y \wedge Z) \to (X' \wedge Y' \wedge Z')$. This is equivalent to the Horn formula $(X \wedge Y \wedge Z) \to X' \wedge (X \wedge Y \wedge Z) \to Y' \wedge (X \wedge Y \wedge Z) \to Z'$. We thus have a logspace computable function $w \mapsto \widehat{\psi}_w$ such that

- $\widehat{\psi}_w$ is a Horn formula,
- M accepts w iff $\widehat{\psi}_w$ is satisfiable.

Therefore, $A \leq_{\log}$ HORN-SAT. Q.E.D.

Another fundamental P-complete problem is the computation of winning regions in finite (reachability) games.

Such a game is given by a game graph $G = (V, V_0, V_1, E)$ with a finite set V of positions, partitioned into V_0 and V_1, such that Player 0 moves from positions $v \in V_0$, moves from positions $v \in V_1$. All moves are along edges, and we use the term *play* to describe a (finite or infinite) sequence $v_0 v_1 v_2 \ldots$ with $(v_i, v_{i+1}) \in E$ for all i. We use a simple positional winning condition: Move or lose! Player σ wins at position v if $v \in V_{1-\sigma}$ and $vE = \varnothing$, i.e., if the position belongs to the opponent and there are no possible moves possible from that position. Note that this winning condition only applies to finite plays, infinite plays are considered to be a draw.

A *strategy* for Player σ is a mapping

$$f : \{v \in V_\sigma : vE \neq \varnothing\} \to V$$

with $(v, f(v)) \in E$ for all $v \in V$. We call f *winning* from position v if all plays that start at v and are consistent with f are won by Player σ.

We now can define *winning regions* W_0 and W_1:

$$W_\sigma = \{v \in V : \text{Player } \sigma \text{ has a winning strategy from position } v\}.$$

This leads to several algorithmic problems for a given game G: The computation of winning regions W_0 and W_1, the computation of winning strategies, and the associated decision problem

$$\text{GAME} := \{(G, v) : \text{Player 0 has a winning strategy for } G \text{ from } v\}.$$

Theorem 3.12. GAME is P-complete and decidable in time $\mathrm{O}(|V| + |E|)$.

A simple polynomial-time approach to solve GAME is to compute the winning regions inductively: $W_\sigma = \bigcup_{n\in\mathbb{N}} W_\sigma^n$, where

$$W_\sigma^0 = \{v \in V_{1-\sigma} : vE = \varnothing\}$$

is the set of terminal positions which are winning for Player σ, and

$$W_\sigma^{n+1} = \{v \in V_\sigma : vE \cap W_\sigma^n \neq \varnothing\} \cup \{v \in V_{1-\sigma} : vE \subseteq W_\sigma^n\}$$

is the set of positions from which Player σ can win in at most $n+1$ moves.

After $n \leq |V|$ steps, we have that $W_\sigma^{n+1} = W_\sigma^n$, and we can stop the computation here.

To solve GAME in linear time, use the slightly more involved Algorithm 3.1. Procedure Propagate will be called once for every edge in the game graph, so the running time of this algorithm is linear with respect to the number of edges in $\mathcal{G}$.

The problem GAME is equivalent to the satisfiability problem for propositional Horn formulae. We recall that propositional Horn formulae are finite conjunctions $\bigwedge_{i\in I} C_i$ of clauses C_i of the form

$$\begin{array}{lcl} X_1 \wedge \ldots \wedge X_n & \rightarrow & X \quad \text{or} \\ \underbrace{X_1 \wedge \ldots \wedge X_n}_{\text{body}(C_i)} & \rightarrow & \underbrace{0}_{\text{head}(C_i)} \quad . \end{array}$$

A clause of the form X or $1 \rightarrow X$ has an empty body.

Algorithm 3.1. A linear time algorithm for GAME

Input: A game $\mathcal{G} = (V, V_0, V_1, E)$
Output: Winning regions W_0 and W_1

```
foreach v ∈ V do                                /* 1: Initialisation */
    win[v] := ⊥
    P[v] := ∅
    n[v] := 0
endfor
foreach (u, v) ∈ E do                          /* 2: Calculate P and n */
    P[v] := P[v] ∪ {u}
    n[u] := n[u] + 1
endfor
foreach v ∈ V_0 do                              /* 3: Calculate win */
    if n[v] = 0 then Propagate(v, 1)
endfor
foreach v ∈ V \ V_0 do
    if n[v] = 0 then Propagate(v, 0)
endfor
returnwin
procedure Propagate(v, σ)
if win[v] ≠ ⊥ then return
win[v] := σ                     /* 4: Mark v as winning for player σ */
foreach u ∈ P[v] do  /* 5: Propagate change to predecessors */
    n[u] := n[u] − 1 if u ∈ V_σ or n[u] = 0 then Propagate(u, σ)
endfor
```

We will show that SAT-HORN and GAME are mutually reducible via logspace and linear-time reductions.

(1) GAME $\leq_{\text{log-lin}}$ SAT-HORN
For a game $\mathcal{G} = (V, V_0, V_1, E)$, we construct a Horn formula $\psi_{\mathcal{G}}$ with clauses

$$\begin{aligned} v \to u \qquad & \text{for all } u \in V_0 \text{ and } (u,v) \in E, \text{ and} \\ v_1 \wedge \ldots \wedge v_m \to u \qquad & \text{for all } u \in V_1 \text{ and } uE = \{v_1, \ldots, v_m\}. \end{aligned}$$

The minimal model of $\psi_{\mathcal{G}}$ is precisely the winning region of Player 0, so

$$(\mathcal{G}, v) \in \text{GAME} \quad \Longleftrightarrow \quad \psi_{\mathcal{G}} \wedge (v \to 0) \text{ is unsatisfiable.}$$

(2) SAT-HORN $\leq_{\text{log-lin}}$ GAME
For a Horn formula $\psi(X_1, \dots, X_n) = \bigwedge_{i \in I} C_i$, we define a game $\mathcal{G}_\psi = (V, V_0, V_1, E)$ as follows:

$$V = \underbrace{\{0\} \cup \{X_1, \dots, X_n\}}_{V_0} \cup \underbrace{\{C_i : i \in I\}}_{V_1} \text{ and}$$

$$E = \{X_j \to C_i : X_j = \text{head}(C_i)\} \cup \{C_i \to X_j : X_j \in \text{body}(C_i)\},$$

i.e., , Player 0 moves from a variable to some clause containing the variable as its head, and Player 1 moves from a clause to some variable in its body. Player 0 wins a play if, and only if, the play reaches a clause C with $\text{body}(C) = \varnothing$. Furthermore, Player 0 has a winning strategy from position X if, and only if, $\psi \models X$, so we have

$$\text{Player 0 wins from position 0} \quad \Longleftrightarrow \quad \psi \text{ is unsatisfiable.}$$

In particular, GAME is P-complete, and SAT-HORN is solvable in linear time.

3.4 NLogspace-complete problems

We already know that the reachability problem, i.e. to decide, given a directed graph G and two nodes a and b, whether there is a path from a to b in G, is in NLogspace.

Theorem 3.13. REACHABILITY is NLogspace-complete.

Proof. Let A be an arbitrary problem in NLogspace. There is a nondeterministic Turing machine M that decides A with workspace $c \log n$. We prove that $A \leq_{\log}$ REACHABILITY by associating, with every input x for M, a graph $G_x = (X_x, E_x)$ and two nodes a and b, such that M accepts x if, and only if, there is a path from a to b in G_x. The set of nodes of G_x is

$$V_x := \{C : C \text{ is a partial configuration of } M \text{ with workspace } c \log |x|\} \cup \{b\},$$

and the set of edges is

$$E_x := \{(C, C') : (C, x) \vdash_M (C'x)\} \cup \{(C_a, b) : C_a \text{ is accepting}\}.$$

Recall that a partial configuration is a configuration without the description of the input. Each partial configuration in V_x can be described by a word of length $O(\log |x|)$. Further we define a to be the initial partial configuration of M. Clearly (G_x, a, b) is constructible with logarithmic space from x and there is a path from a to b in G_x if, and only if, there is an accepting computation of M on x. Q.E.D.

We next discuss a variant of SAT that is NLOGSPACE-complete.

Definition 3.14. A formula is in r-CNF if it is in CNF and every clause contains at most r literals: $\psi = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m_i} Y_{i_j}$ with $m_i \leq r$ for all i. Furthermore, r-SAT $:= \{\psi \text{ in } r\text{-CNF} \ : \ \psi \text{ is satisfiable}\}$.

It is known that r-SAT is NP-complete for all $r \geq 3$.

To the contrary, 2-SAT can be solved in polynomial time, e.g., by resolution:

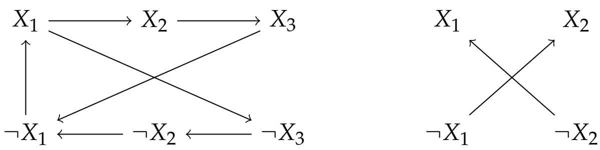
- The resolvent of two clauses with ≤ 2 literals contains at most 2 literals.
- At most $O(n^2)$ clauses with ≤ 2 literals can be formed with n variables.

Hence, we obtain that $\text{Res}^*(\psi)$ for a formula ψ in 2-CNF can be computed in polynomial time. One can show an even stronger result.

Theorem 3.15. 2-SAT is in NLOGSPACE.

Proof. We show that $\{\psi : \psi \text{ in 2-CNF}, \psi \text{ unsatisfiable}\} \in$ NLOGSPACE. Then, by the Theorem of Immerman and Szelepcsényi, also 2-SAT $\in$ NLOGSPACE. The reduction maps a formula $\psi \in$ 2-CNF to the following directed graph $G_\psi = (V, E)$:

- $V = \{X, \neg X : X \in \tau(\psi)\}$ represents the literals of ψ.
- $E = \{(Y, Z) : \psi \text{ contains a clause equivalent to } (Y \rightarrow Z)\}$.



(a) G_{ψ_a} (b) G_{ψ_b}

Figure 3.1. Graphs for $\psi_a = X_1 \wedge (\neg X_1 \vee X_2) \wedge (X_3 \vee \neg X_2) \wedge (\neg X_3 \vee \neg X_1)$ and $\psi_b = (X_1 \vee X_2)$

Example 3.16. Figures 3.1(a) and 3.1(b) show the graphs constructed for an unsatisfiable and a satisfiable 2-CNF formula, respectively.

Lemma 3.17 (Krom-Criterion)**.** Let ψ be in 2-CNF. ψ is unsatisfiable if, and only if, there exists a variable $X \in \tau(\psi)$ such that G_ψ contains a path from X to $\neg X$ and one from $\neg X$ to X.

The problem

$$L = \{(G, a, b) : G \text{ directed graph, there is a path from } a \text{ to } b\}$$

is also called the labyrinth problem. A formula ψ is unsatisfiable if, and only if, there exists a variable $X \in \tau(\psi)$ such that $(G_\psi, X, \neg X) \in L$ and $(G_\psi, \neg X, X) \in L$. Since $L \in$ NLogspace, the claim follows. Q.E.D.

Proof (of Lemma 3.17). We use the notation $Y \rightarrow^*_\psi Z$ to denote that there exists a path from Y to Z in G_ψ.

Let $\mathfrak{I}$ be an interpretation such that $\mathfrak{I}(\psi) = 1$. Then, $\mathfrak{I}(Y) = 1, Y \rightarrow^*_\psi Z \implies \mathfrak{I}(Z) = 1$. Hence, if $X \rightarrow^*_\psi \neg X \rightarrow^*_\psi X$, then ψ is unsatisfiable.

Conversely, for all $X \in \tau(\psi)$, either not $X \rightarrow^*_\psi \neg X$ or not $\neg X \rightarrow^*_\psi X$. In this case, Algorithm 3.2 constructs an interpretation $\mathfrak{I}$ such that $\mathfrak{I}(\psi) = 1$.

It is not possible to produce conflicting assignments resulting from $Y \rightarrow^*_\psi Z$ as well as $Y \rightarrow^*_\psi \neg Z$ since this would imply $\neg Z \rightarrow^*_\psi \neg Y$ and $Z \rightarrow^*_\psi \neg Y$, and hence $Y \rightarrow^*_\psi Z \rightarrow^*_\psi \neg Y$. But Y was chosen as to not have this property.

Algorithm 3.2

$U := \tau(\psi) \cup \neg\tau(\psi)$
while $U \neq \varnothing$ **do**
  choose $Y \in U$ such that $Y \to_{\psi}^{*} \neg Y$ does not hold
  $\mathfrak{I}(Y) := 1$
  $U := U - \{Y, \neg Y\}$
  foreach Z *such that* $Y \to_{\psi}^{*} Z$ **do**
    $\mathfrak{I}(Z) := 1$
    $U := U - \{Z, \neg Z\}$
  endfor
endwhile

Thus, Algorithm 3.2 constructs an interpretation $\mathfrak{I}$ since, for every variable $X \in \tau(\psi)$, either $\mathfrak{I}(X) = 1$ or $\mathfrak{I}(\neg X) = 1$. However, due to the nondeterministic choice of Y in each loop, the resulting interpretation is not uniquely determined.

Let $\mathfrak{I}$ be an interpretation constructed by Algorithm 3.2. It remains to prove that $\mathfrak{I}$ satisfies each clause $(Z \vee Z')$, and thus ψ.

Otherwise, there is a clause $(Z \vee Z')$ such that $\mathfrak{I}(Z) = \mathfrak{I}(Z') = 0$, i.e., $\mathfrak{I}(\neg Z) = 1$. This implies, that the algorithm has chosen a literal Y such that $Y \to_{\psi}^{*} \neg Z$ but $Y \to_{\psi}^{*} \neg Y$ does not hold. Since $\neg Z \to_{\psi}^{*} Z'$, we obtain $Y \to_{\psi}^{*} Z'$ and hence $\mathfrak{I}(Z') = 1$, which is a contradiction. Q.E.D.

Remark 3.18. Formulae in 2-CNF are sometimes called *Krom-formulae*.

Theorem 3.19. 2-SAT is NLOGSPACE-complete.

Proof. We prove that REACHABILITY $\leq_{\log}$ 2-SAT.

Given a directed graph $G = (V, E)$ with nodes a and b, we construct the 2-CNF formula

$$\psi_{G,a,b} := a \wedge \bigwedge_{(u,v) \in E} (u \to v) \wedge \neg b.$$

Clearly this defines a logspace-reduction from the reachability problem to 2-SAT. Q.E.D.

3.5 A Pspace-complete problem

Let us first recall two important properties of the complexity class Pspace $:= \bigcup_k$ Dspace(n^k).

- Pspace $= \bigcup_{k\in\mathbb{N}}$ Nspace$(n^k) =$ NPspace because, by the Theorem of Savitch, Nspace$(S) \subseteq$ Dspace(S^2).
- NP $\subseteq$ Pspace since Ntime$(n^k) \subseteq$ Nspace$(n^k) \subseteq$ Dspace$(n^{2k}) \subseteq$ Pspace.

A problem A is Pspace-hard if $B \leq_{\mathrm{p}} A$ for all $B \in$ Pspace. A is Pspace-complete if $A \in$ Pspace and A is Pspace-hard.

As an example of Pspace-complete problems, we consider the evaluation problem for quantified propositional formulae (also called QBF for "quantified Boolean formulae").

Definition 3.20. *Quantified propositional logic* is an extension of (plain) propositional logic. It is the smallest set closed under disjunction, conjunction and complement that allows quantification over propositional variables in the following sense: If ψ is a formula from quantified propositional logic and X a propositional variable, then $\exists X\psi, \forall X\psi$ are also quantified propositional formulae.

Example 3.21. $\exists X(\forall Y(X \vee Y) \wedge \exists Z(X \vee Z))$.

By free(ψ) we denote the set of free propositional variables in ψ. Every propositional interpretation $\mathfrak{I} : \sigma \to \{0,1\}$ with $\sigma \subseteq \tau$ defines logical values $\mathfrak{I}(\psi)$ for all quantified propositional formulae ψ with free$(\psi) \subseteq \sigma$. Let $\mathfrak{I}$ be an interpretation and $X \in \tau$ a propositional variable. Further, we write $\mathfrak{I}[X = 1]$ for the interpretation that agrees with $\mathfrak{I}$ on all $Y \in \tau, Y \neq X$ and interprets X with 1. Analogously, let $\mathfrak{I}[X = 0]$ be the interpretation with $\mathfrak{I}[X/0](Y) = \mathfrak{I}(Y)$ for $Y \neq X$ and $\mathfrak{I}[X/0](X) = 0$. Then, $\mathfrak{I}(\exists X\psi) = 1$ if, and only if, $\mathfrak{I}[X/0](\psi) = 1$ or $\mathfrak{I}[X/1](\psi) = 1$. Similarly, $\mathfrak{I}(\forall X\psi) = 1$ if, and only if, $\mathfrak{I}[X/0](\psi) = 1$ and $\mathfrak{I}[X/1](\psi) = 1$.

Observe that if free$(\psi) = \varnothing$ the value $\mathfrak{I}(\psi) \in \{0,1\}$ does not depend on a concrete interpretation $\mathfrak{I}$; we have either $\mathfrak{I}(X) = 1$ (ψ is *satisfied*) or $\mathfrak{I}(X) = 0$ (ψ is *unsatisfied*). The formula $\exists X(\forall Y(X \vee Y) \wedge \exists Z(X \vee Z))$ is satisfied, for example.

Algorithm 3.3. Eval($\psi, \mathfrak{I}$)

Input: $\psi, \mathfrak{I}$
if $\psi = X \in V$ **then** return $\mathfrak{I}(X)$
if $\psi = (\varphi_1 \vee \varphi_2)$ **then**
  if *Eval*$(\varphi_1, \mathfrak{I}) = 1$ **then** return 1 **else** return Eval$(\varphi_2, \mathfrak{I})$
endif
if $\psi = (\varphi_1 \wedge \varphi_2)$ **then**
  if *Eval*$(\varphi_1, \mathfrak{I}) = 0$ **then** return 0 **else** return Eval$(\varphi_2, \mathfrak{I})$
endif
if $\psi = \neg\varphi$ **then** return $1 -$ Eval$(\varphi, \mathfrak{I})$
if $\psi = \exists X \varphi$ **then**
  if *Eval*$(\varphi, \mathfrak{I}[X = 0]) = 1$ **then** return 1 **else**
    return Eval$(\varphi, \mathfrak{I}[X = 1])$
  endif
endif
if $\psi = \forall X \varphi$ **then**
  if *Eval*$(\varphi, \mathfrak{I}[X = 0]) = 0$ **then** return 0 **else**
    return Eval$(\varphi, \mathfrak{I}[X = 1])$
  endif
endif

Definition 3.22.

$$\text{QBF} := \{\psi \text{ a quantified PL formula} : \text{free}(\psi) = \varnothing \ , \ \psi \text{ true}\}.$$

Remark 3.23. Let $\psi = \psi(X_1, \dots, X_n)$ be a propositional formula (i.e., one that does not contain quantifiers). Then,

$$\psi \in \text{SAT} \iff \exists X_1 \dots \exists X_n \psi \in \text{QBF}.$$

QBF is therefore at least as hard as SAT. Actually, we will show that QBF is PSPACE-complete.

Theorem 3.24. QBF $\in$ PSPACE.

Proof. The recursive procedure Eval$(\psi, \mathfrak{I})$ presented in Algorithm 3.3 computes the value $\mathfrak{I}(\psi)$ for a quantified propositional formula ψ and $\mathfrak{I}\colon \text{free}(\psi) \to \{0, 1\}$.

This procedure uses $O(n^2)$ space. It is easy to see that $\mathfrak{I}(\psi)$ is computed correctly. Q.E.D.

Theorem 3.25. QBF is PSPACE-hard.

Proof. Consider a problem A in PSPACE and let M be some n^k-space bounded 1-tape TM with $L(M) = A$. Every configuration of M on some input w of length n can be described by a tuple $\bar{X}$ of propositional variables consisting of:

X_q	(q is state of M)	: "M is in state q",
$X'_{a,i}$	(a tape symbol, $i \leq n^k$)	: "symbol a is on field i",
X''_j	($j \leq n^k$)	: "M is on position j".

As in the NP-completeness proof for SAT, we construct formulae $\textsc{conf}(\bar{X})$, $\textsc{next}(\bar{X}, \bar{Y})$, $\textsc{input}_w(\bar{X})$ and $\textsc{acc}(\bar{X})$ with the following intended meanings:

- $\textsc{conf}(\bar{X})$: $\bar{X}$ encodes some configuration, i.e., exactly one X_q is true, exactly one $X'_{a,i}$ is true for every i, and exactly one X''_j is true.
- $\textsc{input}_w(\bar{X})$: $\bar{X}$ encodes the initial configuration of M on $w = w_0 \dots w_{n-1}$:

$$\textsc{input}_w(\bar{X}) := \textsc{conf}(\bar{X}) \wedge X_{q_0} \wedge \bigwedge_{i=0}^{n-1} X'_{w_i,i} \wedge \bigwedge_{i=n}^{n^k} X'_{\Box,i} \wedge X''_0.$$

- $\textsc{acc}(\bar{X})$: $\bar{X}$ is an accepting configuration:

$$\textsc{acc}(\bar{x}) := \textsc{conf}(\bar{X}) \wedge \bigvee_{q \in E^+} X_q.$$

- $\textsc{next}(\bar{X}, \bar{Y})$: $\bar{Y}$ is a successor configuration of $\bar{X}$:

$$\textsc{next}(\bar{X}, \bar{Y}) := \bigwedge_i \Big(X''_i \rightarrow \big(\bigwedge_{a,j \neq i} (Y'_{a,j} \leftrightarrow X_{a,j}) \wedge \\ \bigwedge_{\substack{\delta(q,a)=(q',b,m) \\ 0 \leq m+i \leq n^k}} (X_q \wedge X'_{a,i} \rightarrow Y_{q'} \wedge Y'_{b,i} \wedge Y''_{i+m}) \big) \Big).$$

Given w, these formulae can be constructed in polynomial time.

Furthermore, we define the predicate

$$\textsc{eq}(\bar{X}, \bar{Y}) := \bigwedge_q (X_q \leftrightarrow Y_q) \wedge \bigwedge_{a,i} (X'_{a,i} \leftrightarrow Y'_{a,i}) \wedge \bigwedge_j (X''_j \leftrightarrow Y''_j).$$

We inductively construct formulae $\textsc{reach}_m(\bar{X}, \bar{Y})$ expressing that $\bar{X}$ and $\bar{Y}$ encode configurations and $\bar{Y}$ is accessible from $\bar{X}$ in at most 2^m steps. For $m = 0$, let

$$\textsc{reach}_0(\bar{X}, \bar{Y}) := \textsc{conf}(\bar{X}) \wedge \textsc{conf}(\bar{Y}) \wedge (\textsc{eq}(\bar{X}, \bar{Y}) \vee \textsc{next}(\bar{X}, \bar{Y})).$$

A naïve way to define $\textsc{reach}_{m+1}$ would be

$$\textsc{reach}_{m+1}(\bar{X}, \bar{Y}) := \exists \bar{Z} (\textsc{reach}_m(\bar{X}, \bar{Z}) \wedge \textsc{reach}_m(\bar{Z}, \bar{Y})).$$

But then $|\textsc{reach}_{m+1}| \geq 2 \cdot |\textsc{reach}_m|$ so that $|\textsc{reach}_m| \geq 2^m$ and hence grows exponentially. We can, however, construct $\textsc{reach}_{m+1}$ differently so that the exponential growth of the formula length is avoided by using universal quantifiers:

$$\begin{aligned} &\textsc{reach}_{m+1}(\bar{X}, \bar{Y}) := \\ &\qquad \exists \bar{Z} \forall \bar{U} \forall \bar{V} \begin{pmatrix} (\textsc{eq}(\bar{X}, \bar{U}) \wedge \textsc{eq}(\bar{Z}, \bar{V})) \\ \vee (\textsc{eq}(\bar{Z}, \bar{U}) \wedge \textsc{eq}(\bar{Y}, \bar{V})) \end{pmatrix} \rightarrow \textsc{reach}_m(\bar{U}, \bar{V}). \end{aligned}$$

We now obtain:

$$\begin{aligned} |\textsc{reach}_0| &= O(n^k) \text{ for some appropriate } k \text{, and} \\ |\textsc{reach}_{m+1}| &= |\textsc{reach}_m| + O(n^k). \end{aligned}$$

Hence, $|\text{Reach}_m| = O(m \cdot n^k)$.

If M accepts the input w using space n^k it performs at most $\leq 2^{c \cdot n^k}$ steps for some constant c. Set $m := c \cdot n^k$ and

$$\psi_w := \exists \bar{X}\, \exists \bar{Y} (\textsc{input}(\bar{X}) \wedge \textsc{acc}(\bar{Y}) \wedge \textsc{reach}_m(\bar{X}, \bar{Y})).$$

Obviously, ψ_w is constructable from w in polynomial time and $\psi_w \in$ QBF if and only if $w \in L(M)$. Therefore, QBF is PSPACE-complete. Q.E.D.