NP-completeness

1 Prelude about O-notation

Lemma 1. *Let* $f, g : \mathbb{N}_0 \to \mathbb{N}$ *be functions,* $g(n) > 0$ *for all* n *and* $f(n) = O(g(n))$.
Then there is a constant C *such that*

$$f(n) \leq C \cdot g(n) \quad \textit{for all } n$$

1 Prelude about O-notation

Lemma 1. *Let $f, g : \mathbb{N}_0 \to \mathbb{N}$ be functions, $g(n) > 0$ for all n and $f(n) = O(g(n))$.*
Then there is a constant C such that

$$f(n) \leq C \cdot g(n) \quad \textit{for all } n$$

- there is constant c such that

$$f(n) \leq c \cdot g(n) \quad \text{for all } n \geq n_0$$

- define

$$\begin{aligned}
r(n) &= f(n)/g(n) \\
c' &= \max\{r(n) \ : \ n < n_0\} \\
C &= \max\{c, c'\} \\
f(n) &= r(n) \cdot g(n) \\
&\leq \begin{cases} c' \cdot g(n) & n < n_0 \\ c \cdot g(n) & n \geq 0 \end{cases} \\
&\leq C \cdot g(n)
\end{aligned}$$

2 Polynomial reducibility and completeness

2.1 Polynomial reducibility

reminder: reducibility

- $L, L' \subseteq A^*$ languages
- $L \leq L'$ iff there is *computable* function

  $$f : A^* \to A^*$$

  translating the question $w \in L$ into $f(w) \in L'$

  $$\forall w. \quad w \in L \leftrightarrow f(w) \in L'$$

Lemma 2.

$$L' \textit{ decidable} \wedge L \leq L' \quad \to \quad L \textit{ decidable}$$

2 Polynomial reducibility and completeness

2.1 Polynomial reducibility

reminder: reducibility

- $L', L \subseteq A^*$ languages
- $L' \leq L$ iff there is *computable* function

  $$f : A^* \to A^*$$

  translating the question $w \in L'$ into $f(w) \in L$

  $$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

Lemma 2.

$$L \textit{ decidable } \wedge L' \leq L \quad \to \quad L' \textit{ decidable}$$

def: polynomially time bounded machines A (Turing) machine m is *polynomially time bounded* if there is an exponent k such that for all inputs w machine M started with w makes at most $O(|w|^k)$ steps.

def: computability in poynomial time A function f is *computable in polynomial time* if it is computed (i.e. $f = f_M$) by a polynomialy time bounded machine M,

2 Polynomial reducibility and completeness

2.1 Polynomial reducibility

reminder: reducibility

- $L', L \subseteq A^*$ languages
- $L' \leq L$ iff there is *computable* function

$$f : A^* \to A^*$$

translating the question $w \in L'$ into $f(w) \in L$

$$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

Lemma 2.

$$L \textit{ decidable } \wedge L' \leq L \quad \to \quad L' \textit{ decidable}$$

def: polynomially time bounded machines A (Turing) machine m is *polynomially time bounded* if there is an exponent k such that for all inputs w machine M started with w makes at most $O(|w|^k)$ steps.

def: computability in poynomial time A function f is *computable in polynomial time* if it is computed (i.e. $f = f_M$) by a polynomialy time bounded machine M,

def: polynomial reducibility: *reducibility in polynomial time* resp. *p-reducibilty*

- $L', L \subseteq A^*$ languages
- $L' \leq_p L$: iff there is function *computable in poynomial time*

$$f : A^* \to A^*$$

translating the question $w \in L'$ into $f(w) \in L$

$$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

2 Polynomial reducibility and completeness

2.1 Polynomial reducibility

reminder: reducibility

- $L', L \subseteq A^*$ languages
- $L' \leq L$ iff there is *computable* function

$$f : A^* \to A^*$$

translating the question $w \in L'$ into $f(w) \in L$

$$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

Lemma 2.

$$L \; decidable \; \wedge \; L' \leq L \quad \to \quad L' \; decidable$$

def: polynomially time bounded machines A (Turing) machine m is *polynomially time bounded* if there is an exponent k such that for all inputs w machine M started with w makes at most $O(|w|^k)$ steps.

def: computability in poynomial time A function f is *computable in polynomial time* if it is computed (i.e. $f = f_M$) by a polynomialy time bounded machine M,

def: polynomial reducibility: *reducibility in polynomial time* resp. *p-reducibilty*

- $L', L \subseteq A^*$ languages
- $L' \leq_p L$: iff there is function *computable in poynomial time*

$$f : A^* \to A^*$$

translating the question $w \in L'$ into $f(w) \in L$

$$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

Lemma 3.

$$L \in P \; \wedge \; L' \leq_p L \quad \to \quad L' \in P$$

2 Polynomial reducibility and completeness

2.1 Polynomial reducibility

reminder: reducibility

- $L', L \subseteq A^*$ languages
- $L' \leq L$ iff there is *computable* function

$$f : A^* \to A^*$$

translating the question $w \in L'$ into $f(w) \in L$

$$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

Lemma 2.

$$L \text{ decidable } \wedge L' \leq L \quad \to \quad L' \text{ decidable}$$

def: polynomially time bounded machines A (Turing) machine m is *polynomially time bounded* if there is an exponent k such that for all inputs w machine M started with w makes at most $O(|w|^k)$ steps.

def: computability in poynomial time A function f is *computable in polynomial time* if it is computed (i.e. $f = f_M$) by a polynomialy time bounded machine M,

def: polynomial reducibility: *reducibility in polynomial time* resp. *p-reducibilty*

- $L', L \subseteq A^*$ languages
- $L' \leq_p L$: iff there is function *computable in poynomial time*

$$f : A^* \to A^*$$

translating the question $w \in L'$ into $f(w) \in L$

$$\forall w. \quad w \in L' \leftrightarrow f(w) \in L$$

Lemma 3.

$$L \in P \wedge L' \leq_p L \quad \to \quad L' \in P$$

attention: there is a bit of arithmetic:

- Let M be $O(n^k)$-time bounded acceptor for L
- Let M' be $O(n^{k'})$ time bounded machine computing f
- acceptor M'' for L': with input w of length n
  1. compute $f(w)$ in time $O(n^{k'})$ and observe
  $$|f(w)| = O(n^{k'})$$
  2. behave as M' with input $f(w)$. This takes time
  $$O(|f(w)|^k) = O(n^{k \cdot k'})$$

2.2 Hardness and completeness

Let K be a set of languages (usually a complexity class) and L a language. Most important example

$$K = NP$$

- L is *K-hard* if every language $L' \in K$ is p-reducible to L

$$\forall L' \in K. \quad L' \leq_p L$$

- L' is *K-complete* if L is K-hard and $L \in K$

2.2 Hardness and completeness

Let K be a set of languages (usually a complexity class) and L a language. Most important example

$$K = NP$$

- L is *K-hard* if every language $L' \in K$ is p-reducible to L

$$\forall L' \in K. \quad L' \leq_p L$$

- L' is *K-complete* if L is K-hard and $L \in K$

Single K-complete languages L represent the feasability of accepting languages in the entire complexity class$K$

Lemma 4. *Let L be K-complete. Then*

$$K \subseteq P \quad \leftrightarrow \quad L \in P$$

- $\rightarrow$: trivial as $L \in K$
- $\leftarrow$: let $L' \in K$. Then $L' \leq_p L$ and $L' \in P$ by lemma 3.

2.2 Hardness and completeness

Let K be a set of languages (usually a complexity class) and L a language. Most important example

$$K = NP$$

- L is *K-hard* if every language $L' \in K$ is p-reducible to L

$$\forall L' \in K. \quad L' \leq_p L$$

- L' is *K-complete* if L is K-hard and $L \in K$

Single K-complete languages L represent the feasability of accepting languages in the entire complexity class$K$

Lemma 4. *Let L be K-complete. Then*

$$K \subseteq P \quad \leftrightarrow \quad L \in P$$

- $\rightarrow$: trivial as $L \in K$
- $\leftarrow$: let $L' \in K$. Then $L' \leq_p L$ and $L' \in P$ by lemma 3.

Lemma 5. *Let L be NP-complete. Then*

$$NP = P \quad \leftrightarrow \quad L \in P$$

Proof. if $L \in P$ we have by lemma 4

$$P \subseteq NP \subseteq P$$

3 A trivial NP-complete language

define language L as the set of words

$$u\#vX^j$$

with

- $u \in \mathbb{B}^+$ codes a nondeterministic 1-tape TM M_u
- $v \in \{0,1,\#\}^+$ is an input for M_u
- padding symbol X
- M_u can accept v in $j/|u|$ steps

Lemma 6.

$$L \in NP$$

- use nondeterministic universal machine U with 3 tapes
  1. tapes 1 and 2: preprocessor and simulation of steps of M_u
  2. tape 3 counting simulated steps.

3 A trivial NP-complete language

define language L as the set of words

$$u\#vX^j$$

with

- $u \in \mathbb{B}^+$ codes a nondeterministic 1-tape TM M_u
- $v \in \{0,1,\#\}^+$ is an input for M_u
- padding symbol X
- M_u can accept v in $j/|u|$ steps

Lemma 6.

$$L \in NP$$

- use nondeterministic universal machine U with 3 tapes
  1. tapes 1 and 2: preprocessor and simulation of steps of M_u
  2. tape 3 counting simulated steps.

Then time

1. for preprocessor using 2 tapes is
$$O(|u| \cdot |v|) = O(n^2)$$
2. for simulating $s = j/|u|$ steps
$$O(|u| \cdot s) = (O(j)) = O(n)$$

3 A trivial NP-complete language

define language L as the set of words

$$u\#vX^j$$

with

- $u \in \mathbb{B}^+$ codes a nondeterministic 1-tape TM M_u
- $v \in \{0,1,\#\}^+$ is an input for M_u
- padding symbol X
- M_u can accept v in $j/|u|$ steps

Lemma 6.

$$L \in NP$$

- use nondeterministic universal machine U with 3 tapes
  1. tapes 1 and 2: preprocessor and simulation of steps of M_u
  2. tape 3 counting simulated steps.

Then time

1. for preprocessor using 2 tapes is

$$O(|u| \cdot |v|) = O(n^2)$$

2. for simulating $s = j/|u|$ steps

$$O(|u| \cdot s) = (O(j)) = O(n)$$

Lemma 7. *L is NP-hard*

- Let $L' \in NP$ be accepted by $O(n^k)$-time bounded nondeterministic k-tape TM M.
- tape reduction gives $O(n^{2k})$ time bounded 1-tape machine M' accepting L.
- by lemma 1 there is C such that machine M started with input of length n makes at most $C \cdot n^{2k}$ steps.
- let $M' = M_u$.
- reducing L' to L by machine M'': with input v of length n output

$$u\#vX^j \quad \text{with} \quad j = C \cdot n^{2k} \cdot |u|$$

3 A trivial NP-complete language

define language L as the set of words

$$u\#vX^j$$

with

- $u \in \mathbb{B}^+$ codes a nondeterministic 1-tape TM M_u
- $v \in \{0,1,\#\}^+$ is an input for M_u
- padding symbol X
- M_u can accept v in $j/|u|$ steps

Lemma 6.

$$L \in NP$$

- use nondeterministic universal machine U with 3 tapes
  1. tapes 1 and 2: preprocessor and simulation of steps of M_u
  2. tape 3 counting simulated steps.

Then time

1. for preprocessor using 2 tapes is

$$O(|u|\cdot|v|) = O(n^2)$$

2. for simulating $s = j/|u|$ steps

$$O(|u|\cdot s) = (O(j)) = O(n)$$

Lemma 7. *L is NP-hard*

- Let $L' \in NP$ be accepted by $O(n^k)$-time bounded nondeterministic k-tape TM M.
- tape reduction gives $O(n^{2k})$ time bounded 1-tape machine M' accepting L.
- by lemma 1 there is C such that machine M started with input of length n makes at most $C\cdot n^{2k}$ steps.
- let $M' = M_u$.
- reducing L' to L by machine M'': with input v of length n output

$$u\#vX^j \quad \text{with} \quad j = C\cdot n^{2k}\cdot|u|$$

so far: all this could be abstract nonsense, possibly of some philosophical interest...

4 SAT

4.1 reminder: Boolean expressions

def: Boolean expressions

- Variables
$$V = \{X_1, X_2, \ldots\}$$
- Boolean expressions BE, the *non extended* version.
  1. $$V, \mathbb{B} \subset BE$$
  2. if $A, B \in BE$ then
  $$(A \wedge B)\ ,\ (A \vee B)\ ,\ \sim (A) \in BE$$
  3. this are all

priorities

$$\sim, \wedge, \vee$$

4 SAT

4.1 reminder: Boolean expressions

def: Boolean expressions

- Variables
$$V = \{X_1, X_2, \ldots\}$$
- Boolean expressions BE, the *non extended* version.
  1. $$V, \mathbb{B} \subset BE$$
  2. if $A, B \in BE$ then
  $$(A \wedge B)\ ,\ (A \vee B)\ ,\ \sim(A) \in BE$$
  3. this are all

priorities

$$\sim, \wedge, \vee$$

def: valuations

$$\varphi : V \to \mathbb{B}$$

extended to Boolean expressions by

$$\begin{aligned}
\varphi(a) &= a \quad \text{for } a \in \mathbb{B} \\
\varphi(\sim A) &= \sim \varphi(A) \\
\varphi(A \circ B) &= \varphi(A) \circ \varphi(B) \quad \text{for } \circ \in \{\wedge\ ,\ \vee\}
\end{aligned}$$

4.1 reminder: Boolean expressions

def: Boolean expressions

- Variables
$$V = \{X_1, X_2, \ldots\}$$
- Boolean expressions BE, the *non extended* version.
  1. $$V, \mathbb{B} \subset BE$$
  2. if $A, B \in BE$ then
  $$(A \wedge B)\ ,\ (A \vee B)\ ,\ \sim(A) \in BE$$
  3. this are all

priorities

$$\sim, \wedge, \vee$$

def: valuations

$$\varphi : V \to \mathbb{B}$$

extended to Boolean expressions by

$$\begin{aligned} \varphi(a) &= a \quad \text{for } a \in \mathbb{B} \\ \varphi(\sim A) &= \sim \varphi(A) \\ \varphi(A \circ B) &= \varphi(A) \circ \varphi(B) \quad \text{for } \circ \in \{\wedge\ ,\ \vee\} \end{aligned}$$

special Boolean expressions

- *literals*: variables X_i or their complement, written as
$$\sim X_i\ ,\ /X_i\ ,\ \overline{X_i}$$
For $a \in \mathbb{B}$ and $X \in V$
$$X^a = \begin{cases} \overline{X} & a = 0 \\ X & a = 1 \end{cases}$$
monomials: ANDing literals L_i together, written as
$$L_1 \wedge \ldots \wedge L_n \quad , \quad L_1 \ldots L_n$$
- *disjunctive normal forms, DNF, Boolean polynomials*: ORing monomials M_i together
$$M_1 \vee \ldots \vee M_s$$

4.1 reminder: Boolean expressions

def: Boolean expressions

- Variables
$$V = \{X_1, X_2, \ldots\}$$
- Boolean expressions BE, the *non extended* version.
  1. $$V, \mathbb{B} \subset BE$$
  2. if $A, B \in BE$ then
  $$(A \wedge B)\ ,\ (A \vee B)\ ,\ \sim(A) \in BE$$
  3. this are all

priorities

$$\sim, \wedge, \vee$$

def: valuations

$$\varphi : V \to \mathbb{B}$$

extended to Boolean expressions by

$$\begin{aligned}
\varphi(a) &= a \quad \text{for } a \in \mathbb{B} \\
\varphi(\sim A) &= \sim \varphi(A) \\
\varphi(A \circ B) &= \varphi(A) \circ \varphi(B) \quad \text{for } \circ \in \{\wedge\ ,\ \vee\}
\end{aligned}$$

special Boolean expressions

- *literals*: variables X_i or their complement, written as
$$\sim X_i\ ,\ /X_i\ ,\ \overline{X_i}$$
For $a \in \mathbb{B}$ and $X \in V$
$$X^a = \begin{cases} \overline{X} & a = 0 \\ X & a = 1 \end{cases}$$
monomials: ANDing literals L_i together, written as
$$L_1 \wedge \ldots \wedge L_n \quad , \quad L_1 \ldots L_n$$
- *disjunctive normal forms, DNF, Boolean polynomials*: ORing monomials M_i together
$$M_1 \vee \ldots \vee M_s$$
- *clauses*: ORing literals L_i together
$$(L_1 \vee \ldots \vee L_n)$$
- *conjunctive normal forms, CNF*: ANDing clauses C_i together
$$C_1 \wedge \ldots \wedge C_s$$
- 3-CNF: conjunctive normal form with at most 3 literals per clause.

4.2 Functions computed by expressions

substituting bit vector a **for variables** For $a \in \mathbb{B}^n$ let φ_a be the valuation which substitutes a_i for X_i

$$\varphi_a(X_i)) = a_i \quad \text{fo all } i$$

function computed by Boolean expression e For $e \in BE$ with variables $X_1, \dots, X_n$ define switching function

$$f_e : \mathbb{B}^n \to \mathbb{B}$$

by

$$f_e(a) = \phi_a(e)$$

i.e. in order to get function value $f_e(a)$ for a plug in a for variables (using valuation φ_a) and evaluate (to $\varphi_a(e)$).

4.2 Functions computed by expressions

substituting bit vector a for variables For $a \in \mathbb{B}^n$ let φ_a be the valuation which substitutes a_i for X_i

$$\varphi_a(X_i)) = a_i \quad \text{fo all } i$$

function computed by Boolean expression e For $e \in BE$ with variables $X_1, \ldots, X_n$ define switching function

$$f_e : \mathbb{B}^n \rightarrow \mathbb{B}$$

by

$$f_e(a) = \phi_a(e)$$

i.e. in order to get function value $f_e(a)$ for a plug in a for variables (using valuation φ_a) and evaluate (to $\varphi_a(e)$).

Using function symbols in expressions

- in Boolean expressions include for function symbols f_i with n_i arguments and the rule:

$$A_1, \ldots, A_{n_i} \in BE \quad \rightarrow \quad f_i(A_1, \ldots, A_{n_i}) \in BE$$

- evaluate

$$\varphi(f_i(A_1, \ldots, A_{n_i})) = f_i(\varphi(A_1), \ldots, \varphi(A_{n_i}))$$

- BE_{pure}: Boolean expressions without function symbols f_i.

4.2 Functions computed by expressions

substituting bit vector a for variables For $a \in \mathbb{B}^n$ let φ_a be the valuation which substitutes a_i for X_i

$$\varphi_a(X_i)) = a_i \quad \text{fo all } i$$

function computed by Boolean expression e For $e \in BE$ with variables $X_1, \ldots, X_n$ define switching function

$$f_e : \mathbb{B}^n \to \mathbb{B}$$

by

$$f_e(a) = \phi_a(e)$$

i.e. in order to get function value $f_e(a)$ for a plug in a for variables (using valuation φ_a) and evaluate (to $\varphi_a(e)$).

Using function symbols in expressions

- in Boolean expressions include for function symbols f_i with n_i arguments and the rule:

$$A_1, \ldots, A_{n_i} \in BE \quad \to \quad f_i(A_1, \ldots, A_{n_i}) \in BE$$

- evaluate

$$\varphi(f_i(A_1, \ldots, A_{n_i})) = f_i(\varphi(A_1), \ldots, \varphi(A_{n_i}))$$

- BE_{pure}: Boolean expressions without function symbols f_i.

equivalences For (extended) Boolean expressions e, e' we write

$$e \equiv e'$$

iff they evaluate for all valuaions to the same value

$$\varphi(e) = \varphi(e') \quad \text{for all } \varphi$$

Lemma 8. *Every switching function f is computed by a Boolean polynomial p*

Lemma 9. *Every switching function f is computed by a conjuctive normal form d*

- for $a \in \mathbb{B}^n$ define monomial

$$m(a) = \bigwedge_{i=1}^{n} X_i^{a_i}$$

  then

$$\varphi(m(a)) = 1 \leftrightarrow \varphi = \varphi_a$$

- OR these monmials together for a with $f(a) = 1$

$$p = \bigvee_{f(a)=1} m(a)$$

  complete conjunctive normal form for f.

Lemma 8. *Every switching function f is computed by a Boolean polynomial p*

- for $a \in \mathbb{B}^n$ define monomial

$$m(a) = \bigwedge_{i=1}^{n} X_i^{a_i}$$

then

$$\varphi(m(a)) = 1 \leftrightarrow \varphi = \varphi_a$$

- OR these monmials together for a with $f(a) = 1$

$$p = \bigvee_{f(a)=1} m(a)$$

complete conjunctive normal form for f.

Lemma 9. *Every switching function f is computed by a conjuctive normal form d*

- for $a \in \mathbb{B}^n$ define clause

$$c(a) = \bigvee_{i=1}^{n} X_i^{\overline{a_i}}$$

then

$$\varphi(c(a)) = 0 \leftrightarrow \varphi = \varphi_a$$

- AND these clauses together for a with $f(a) = 0$

$$d = \bigwedge_{f(a)=0} c(a)$$

complete conjunctive normal form for f.

Lemma 10. *For every clause c there is a formula C in 3-CNF such that for all valuations*

$$\exists\varphi.\ \varphi(c) = 1 \ \leftrightarrow\ \exists\varphi.\ \varphi(C) = 1$$

i.e. c is satisfiable iff C is satisfiable.

- For the Boolean predicate

  $$x \leftrightarrow y \vee z$$

  there is by lemma 9 an equivalent conjunctive normal form $e(x,y,z)$ satisfying

  $$\varphi(e(x,y,z)) = 1 \leftrightarrow \varphi(x) = \varphi(y) \vee \varphi(z)$$

  Exercise: derive this formula from the truth table of the predicate.

Lemma 10. *For every clause c there is a formula C in 3-CNF such that for all valuations*

$$\exists\varphi.\ \varphi(c) = 1 \ \leftrightarrow\ \exists\varphi.\ \varphi(C) = 1$$

i.e. c is satisfiable iff C is satisfiable.

- For the Boolean predicate

  $$x \leftrightarrow y \vee z$$

  there is by lemma 9 an equivalent conjunctive normal form $e(x,y,z)$ satisfying

  $$\varphi(e(x,y,z)) = 1 \leftrightarrow \varphi(x) = \varphi(y) \vee \varphi(z)$$

  Exercise: derive this formula from the truth table of the predicate.

- because only 3 variables are involved, the conjunctive normal form $e(x,y,z)$ is in $3-CNF$.

- Let

  $$c = X_{i_1}^{a_1} \vee \ldots \vee X_{i_s}^{a_s}$$

  be a clause with $s \geq 4$ literals. Introduce new Variables $Q_2, \ldots, Q_s$ and set

  $$C = \ e(Q_2, X_{i_2}^{a_2}, X_{i_1}^{a_1}) \ \wedge \ \bigwedge_{j=3}^{n} e(Q_j, X_{i_j}^{a_j} Q_{j-1})$$

Lemma 10. *For every clause c there is a formula C in 3-CNF such that for all valuations*

$$\exists \varphi.\ \varphi(c) = 1 \ \leftrightarrow\ \exists \varphi.\ \varphi(C) = 1$$

i.e. c is satisfiable iff C is satisfiable.

- For the Boolean predicate

  $$x \leftrightarrow y \vee z$$

  there is by lemma 9 an equivalent conjunctive normal form $e(x,y,z)$ satisfying

  $$\varphi(e(x,y,z)) = 1 \leftrightarrow \varphi(x) = \varphi(y) \vee \varphi(z)$$

  Exercise: derive this formula from the truth table of the predicate.

- because only 3 variables are involved, the conjunctive normal form $e(x,y,z)$ is in $3-CNF$.

- Let

  $$c = X_{i_1}^{a_1} \vee \ldots \vee X_{i_s}^{a_s}$$

  be a clause with $s \geq 4$ literals. Introduce new Variables $Q_2, \ldots, Q_s$ and set

  $$C = \ e(Q_2, X_{i_2}^{a_2}, X_{i_1}^{a_1}) \ \wedge \ \bigwedge_{j=3}^{n} e(Q_j, X_{i_j}^{a_j} Q_{j-1})$$

4.3 Satisfiability problems

def: satisfiable Boolean expression A (not extended) Boolean expression e is *satisfiable* iff there is a valuateion, which makes it 1

$$\exists \phi.\ :\ \phi(e) = 1$$

satisfiablility problems

$$SAT = \{e \in BE_{pure}\ :\ e \text{ satisfiable}\}$$

$$CNF - SAT = \{e\ :\ e \text{ is CNF and satisfiable}\}$$

$$3 - SAT = \{e\ :\ e \text{ is 3-CNF and satisfiable}\}$$

5 Cook's theorem 1971

this result changes everything

Lemma 11. *$CNF-SAT$ is NP-complete*

5 Cook's theorem 1971

this result changes everything

Lemma 11. *$CNF - SAT$ is NP-complete*

- why does it change the world?
- because you can polynomially reduce $CNF - SAT$ to thousands of optimization problems in NP (via long chains of reductions).

5 Cook's theorem 1971

this result changes everything

Lemma 11. *$CNF-SAT$ is NP-complete*

- why does it change the world?
- because you can polynomially reduce $CNF-SAT$ to thousands of optimization problems in NP (via long chains of reductions).

$SAT \in NP$**:**

- nondeterministic TM M with input e of length n
- guesses valuation φ for variables
- evaluates $\varphi(e)$ - easily in time $O(n^2)$
- accept if $\varphi(e)=1$

5 Cook's theorem 1971

this result changes everything

$\boxed{\textbf{Lemma 11. } \textit{CNF} - \textit{SAT is NP-complete}}$

- why does it change the world?
- because you can polynomially reduce $CNF - SAT$ to thousands of optimization problems in NP (via long chains of reductions).

$SAT \in NP$**:**

- nondeterministic TM M with input e of length n
- guesses valuation φ for variables
- evaluates $\varphi(e)$ - easily in time $O(n^2)$
- accept if $\varphi(e) = 1$

$CNF - SAT$ **is NP-hard:**

- Given
  1. $L \in NP$ and $p(n)$-time bounded 1-tape acceptor M of L
  2. input w of length n for M
- we have to construct in polynomial time (in n))
  1. a CNF $f(w)$ such that
  2. M accepts w iff $f(w)$ is stisfiable

5 Cook's theorem 1971

this result changes everything

Lemma 11. *$CNF - SAT$ is NP-complete*

- why does it change the world?
- because you can polynomially reduce $CNF - SAT$ to thousands of optimization problems in NP (via long chains of reductions).

$SAT \in NP$**:**

- nondeterministic TM M with input e of length n
- guesses valuation φ for variables
- evaluates $\varphi(e)$ - easily in time $O(n^2)$
- accept if $\varphi(e) = 1$

$CNF - SAT$ **is NP-hard:**

- Given
  1. $L \in NP$ and $p(n)$-time bounded 1-tape acceptor M of L
  2. input w of length n for M
- we have to construct in polynomial time (in n))
  1. a CNF $f(w)$ such that
  2. M accepts w iff $f(w)$ is stisfiable

the good news (as so often): the semantics of Turing machines are extremely simple.

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

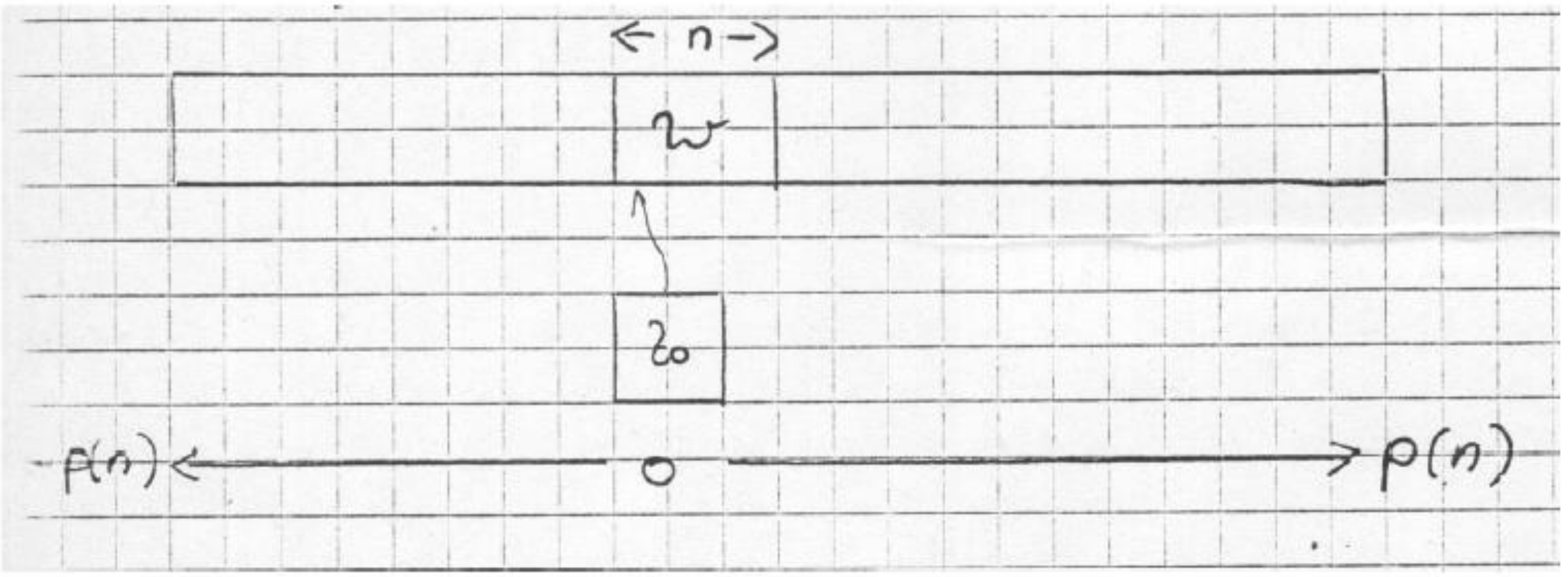

Figure 1: start configuration and reachable space for M started with w

$CNF - SAT$ **is NP-hard:**

- Given
  1. $L \in NP$ and $p(n)$-time bounded 1-tape acceptor M of L
  2. input w of length n for M
- we have to construct in polynomial time (in n))
  1. a CNF $f(w)$ such that
  2. M accepts w iff $f(w)$ is stisfiable

the good news (as so often): the semantics of Turing machines are extremely simple.

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

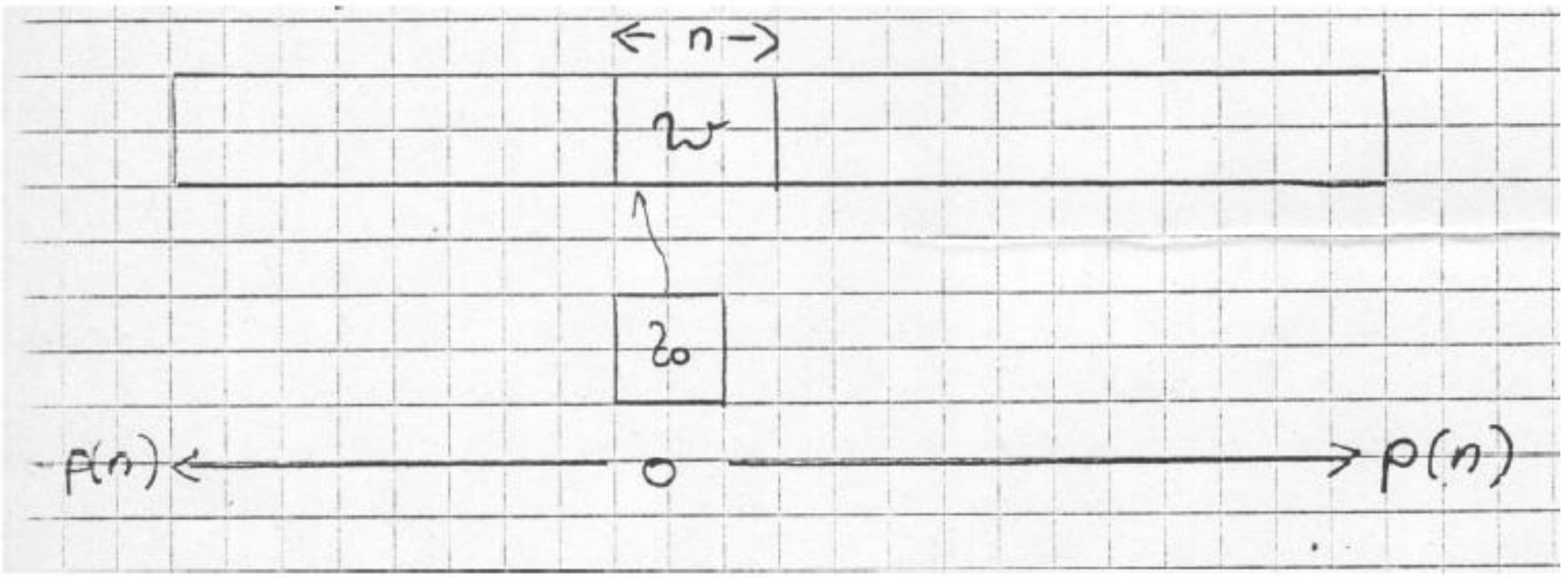

Figure 1: start configuration and reachable space for M started with w

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \,:\, a \in \Sigma\,,\, -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \,:\, -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \,:\, z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

$CNF - SAT$ **is NP-hard:**

- Given
  1. $L \in NP$ and $p(n)$-time bounded 1-tape acceptor M of L
  2. input w of length n for M
- we have to construct in polynomial time (in n))
  1. a CNF $f(w)$ such that
  2. M accepts w iff $f(w)$ is stisfiable

the good news (as so often): the semantics of Turing machines are extremely simple.

$$M = (Z, \Sigma, \delta, z_0, Z_A)$$

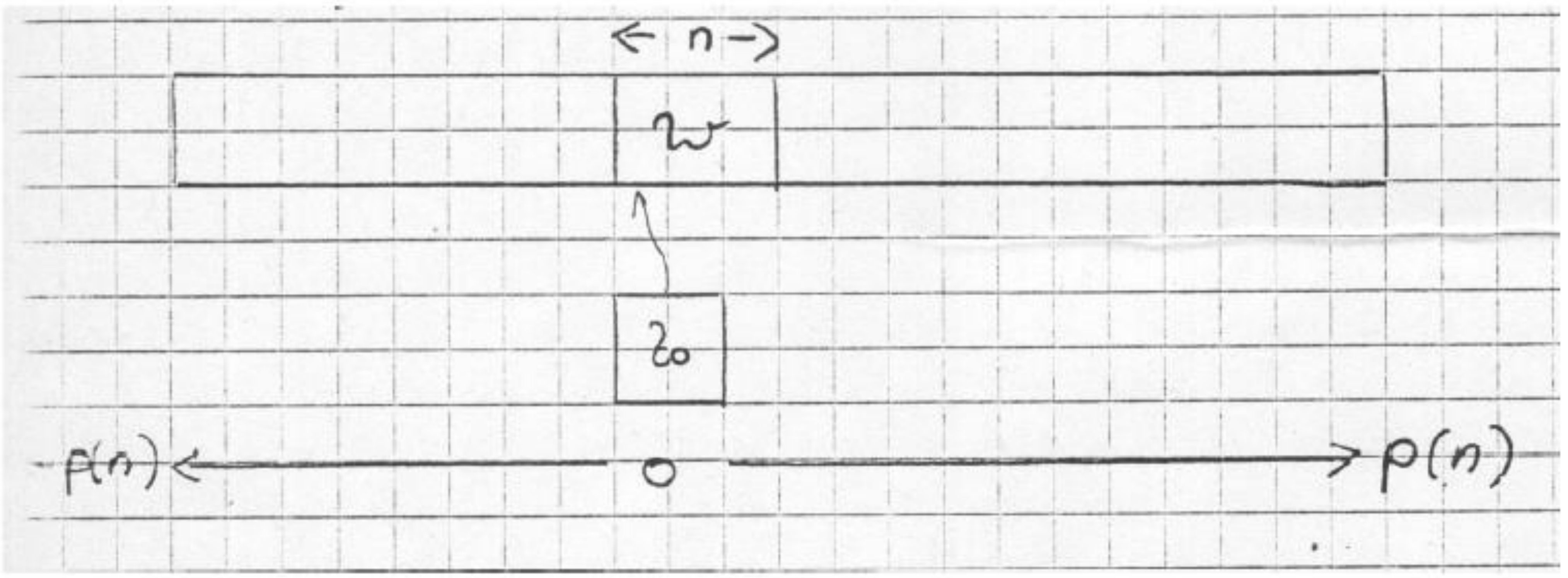

Figure 1: start configuration and reachable space for M started with w

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t \;=\; & \{c_{i,a,t} \,:\, a \in \Sigma\,,\, -p(n) \le i \le p(n)\} \\ & \cup \{h_{i,t} \,:\, -p(n) \le i \le p(n)\} \\ & \cup \{s_{z,t} \,:\, z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

M **accepts** w if there exists configurations $k_0, \ldots, k_{p(n)}$ such that

- $$k_0 = B^{p(n)} z_0 w B^{p(n)-n+1}$$
- $$k_i \vdash k_{i+1} \quad \text{for } i \in [0 : p(n) - 1]$$
- $k_{p(n)}$ is accepting end configuration

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \; : \; a \in \Sigma \,,\; -p(n) \le i \le p(n)\} \\ &\cup \{h_{i,t} \; : \; -p(n) \le i \le p(n)\} \\ &\cup \{s_{z,t} \; : \; z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

M **accepts** w if there exists configurations $k_0, \dots, k_{p(n)}$ such that

- $$k_0 = B^{p(n)} z_0 w B^{p(n)-n+1}$$
- $$k_i \vdash k_{i+1} \quad \text{for } i \in [0 : p(n) - 1]$$
- $k_{p(n)}$ is accepting end configuration

auxiliary formula: exactly one variable is 1

- $$u(x_1, \dots, x_r) = (x_1 \vee \dots x_r) \wedge \bigwedge_{i \neq j} (\overline{x_i} \vee \overline{x_j})$$
- satisfied by setting exactly one variable to 1
  $$\varphi(u(x_1, \dots, x_r)) = 1 \;\leftrightarrow\; \varphi(x_i) = 1 \text{ for exactly one } i$$
- lenght: $O(r^2)$. *We count length of variables as 1*
- coding indices in binary or decimal increases length only by factor $O(\log r)$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \, : \, a \in \Sigma \, , \, -p(n) \le i \le p(n)\} \\ &\cup \{h_{i,t} \, : \, -p(n) \le i \le p(n)\} \\ &\cup \{s_{z,t} \, : \, z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

M **accepts** w if there exists configurations $k_0, \ldots, k_{p(n)}$ such that

- $$k_0 = B^{p(n)} z_0 w B^{p(n)-n+1}$$
- $$k_i \vdash k_{i+1} \quad \text{for } i \in [0 : p(n) - 1]$$
- $k_{p(n)}$ is accepting end configuration

auxiliary formula: exactly one variable is 1

- $$u(x_1, \ldots, x_r) = (x_1 \vee \ldots x_r) \wedge \bigwedge_{i \neq j} (\overline{x_i} \vee \overline{x_j})$$

V_t **codes a configuration**

- Let
$$\begin{aligned} Z &= \{z_1, \ldots, z_q\} \\ \Sigma &= \{a_1, \ldots, a_s\} \end{aligned}$$
- then
$$\begin{aligned} conf(V_t) &= u(s_{z_1,t}, \ldots, s_{z_q,t}) \\ &\wedge u(h_{-p(n),t}, \ldots, h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \le i \le p(n)} u(c_{i,a_1,t}, \ldots, c_{i,a_s,t}) \end{aligned}$$
unique state, unique head position and unique symbol on each tape cell.
- length $O(p^2(n))$
- from a valuation with $\varphi(conf(V_t) = 1$ we can extract configuration $k_t(\varphi)$ of M in a unique and obvious way.

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \,:\, a \in \Sigma\,,\; -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \,:\, -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \,:\, z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

auxiliary formula: exactly one variable is 1

- $$u(x_1,\ldots,x_r) = (x_1 \vee \ldots x_r) \wedge \bigwedge_{i \neq j} (\overline{x_i} \vee \overline{x_j})$$

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t},\ldots,s_{z_q,t}) \\ &\wedge u(h_{-p(n),t},\ldots,h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \leq i \leq p(n)} u(c_{i,a_1,t},\ldots,c_{i,a_s,t}) \end{aligned}$$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \;:\; a \in \Sigma\,,\; -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \;:\; -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \;:\; z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

auxiliary formula: exactly one variable is 1

- $$u(x_1,\ldots,x_r) = (x_1 \vee \ldots x_r) \wedge \bigwedge_{i \neq j} (\overline{x_i} \vee \overline{x_j})$$

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t},\ldots,s_{z_q,t}) \\ &\wedge u(h_{-p(n),t},\ldots,h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \leq i \leq p(n)} u(c_{i,a_1,t},\ldots,c_{i,a_s,t}) \end{aligned}$$

auxiliary formula: if and only if

$$\begin{aligned} x \leftrightarrow y &= (x \wedge y) \vee (\overline{x} \wedge \overline{y}) \\ &= (\overline{x} \vee y) \wedge (x \vee \overline{y}) \end{aligned}$$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \;:\; a \in \Sigma\,,\ -p(n) \le i \le p(n)\} \\ &\quad \cup \{h_{i,t} \;:\; -p(n) \le i \le p(n)\} \\ &\quad \cup \{s_{z,t} \;:\; z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

auxiliary formula: exactly one variable is 1

- $$u(x_1,\dots,x_r) = (x_1 \vee \dots x_r) \wedge \bigwedge_{i \neq j} (\overline{x_i} \vee \overline{x_j})$$

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t},\dots,s_{z_q,t}) \\ &\quad \wedge u(h_{-p(n),t},\dots,h_{p(n),t}) \\ &\quad \wedge \bigwedge_{-p(n)\le i \le p(n)} u(c_{i,a_1,t},\dots,c_{i,a_s,t}) \end{aligned}$$

auxiliary formula: if and only if

$$\begin{aligned} x \leftrightarrow y &= (x \wedge y) \vee (\overline{x} \wedge \overline{y}) \\ &= (\overline{x} \vee y) \wedge (x \vee \overline{y}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n)\le i \le p(n), a \in \Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$
- length $O(p(n))$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \; : \; a \in \Sigma \, , \, -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \; : \; -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \; : \; z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t}, \ldots, s_{z_q,t}) \\ &\wedge u(h_{-p(n),t}, \ldots, h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \leq i \leq p(n)} u(c_{i,a_1,t}, \ldots, c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n) \leq i \leq p(n), a \in \Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \ : \ a \in \Sigma \,,\ -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \ : \ -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \ : \ z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t}, \ldots, s_{z_q,t}) \\ &\wedge u(h_{-p(n),t}, \ldots, h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \leq i \leq p(n)} u(c_{i,a_1,t}, \ldots, c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n) \leq i \leq p(n), a \in \Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case

$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \;:\; a \in \Sigma\,,\ -p(n) \le i \le p(n)\} \\ &\cup \{h_{i,t} \;:\; -p(n) \le i \le p(n)\} \\ &\cup \{s_{z,t} \;:\; z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t},\dots,s_{z_q,t}) \\ &\wedge u(h_{-p(n),t},\dots,h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n)\le i\le p(n)} u(c_{i,a_1,t},\dots,c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1}))$$

when M in state z reads a on cell i in step t:

- either this is not the case

$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$

- or δ^2

$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$

This is an implication

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \ : \ a \in \Sigma \ , \ -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \ : \ -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \ : \ z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t}, \ldots, s_{z_q,t}) \\ &\wedge u(h_{-p(n),t}, \ldots, h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \leq i \leq p(n)} u(c_{i,a_1,t}, \ldots, c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n) \leq i \leq p(n), a \in \Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case

$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$

- or δ^2

$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$

This is an implication

- $$\delta^t = \bigwedge_{-p(n) \leq i \leq p(n), a \in \Sigma, z \in Z} \delta^t_{iazt}$$
- length $O(p(n)) \cdot |\delta_{iazt}|$

Variables for expression $f(w)$ are motivated by figure 1.

for all steps $t \in [0 : p(n)]$ use variables

$$\begin{aligned} V_t &= \{c_{i,a,t} \ : \ a \in \Sigma \, , \ -p(n) \leq i \leq p(n)\} \\ &\cup \{h_{i,t} \ : \ -p(n) \leq i \leq p(n)\} \\ &\cup \{s_{z,t} \ : \ z \in Z\} \end{aligned}$$

interpretation

- $c_{i,a,t} = 1$: before step t cell i contains symbol a
- $h_{i,t} = 1$: before step t the head is on cell i
- $s_{z,z} = 1$: before step t machine M is in state z

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t}, \ldots, s_{z_q,t}) \\ &\wedge u(h_{-p(n),t}, \ldots, h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n) \leq i \leq p(n)} u(c_{i,a_1,t}, \ldots, c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n) \leq i \leq p(n), a \in \Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case

$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$

- or δ^2

$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$

  This is an implication

- $$\delta^t = \bigwedge_{-p(n) \leq i \leq p(n), a \in \Sigma, z \in Z} \delta^t_{iazt}$$

- length $O(p(n)) \cdot |\delta_{iazt}|$

Turing machine step

- $$\begin{aligned} \tilde{\delta}^2_{iazt} &= \bigvee_{(q,b,L) \in \delta(z,a)} c_{i,b,t+1} \wedge h_{i-1,t+1} \wedge s_{q,t+1} \\ &\vee \bigvee_{(q,b,N) \in \delta(z,a)} c_{i,b,t+1} \wedge h_{i,t+1} \wedge s_{q,t+1} \\ &\vee \bigvee_{(q,b,R) \in \delta(z,a)} c_{i,b,t+1} \wedge h_{i+1,t+1} \wedge s_{q,t+1} \end{aligned}$$

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t},\ldots,s_{z_q,t}) \\ &\wedge u(h_{-p(n),t},\ldots,h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n)\le i\le p(n)} u(c_{i,a_1,t},\ldots,c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case

$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$

- or δ^2

$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$

  This is an implication

- $$\delta^t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma,z\in Z} \delta^t_{iazt}$$

- length $O(p(n))\cdot|\delta_{iazt}|$

Turing machine step

- $$\begin{aligned} \tilde{\delta}^2_{iazt} &= \bigvee_{(q,b,L)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i-1,t+1} \wedge s_{q,t+1} \\ &\vee \bigvee_{(q,b,N)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i,t+1} \wedge s_{q,t+1} \\ &\vee \bigvee_{(q,b,R)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i+1,t+1} \wedge s_{q,t+1} \end{aligned}$$

V_t codes a configuration

$$\begin{aligned} conf(V_t) &= u(s_{z_1,t},\dots,s_{z_q,t}) \\ &\wedge u(h_{-p(n),t},\dots,h_{p(n),t}) \\ &\wedge \bigwedge_{-p(n)\le i\le p(n)} u(c_{i,a_1,t},\dots,c_{i,a_s,t}) \end{aligned}$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case

$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$

- or δ^2

$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$

This is an implication

- $$\delta^t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma,z\in Z} \delta^t_{iazt}$$

- length $O(p(n))\cdot|\delta_{iazt}|$

Turing machine step

- $$\begin{aligned} \tilde{\delta}^2_{iazt} &= \bigvee_{(q,b,L)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i-1,t+1} \wedge s_{q,t+1} \\ &\vee \bigvee_{(q,b,N)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i,t+1} \wedge s_{q,t+1} \\ &\vee \bigvee_{(q,b,R)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i+1,t+1} \wedge s_{q,t+1} \end{aligned}$$

- not CNF but length $O(1)$. Obtain δ^1_{iazt} by converting $\tilde{\delta}^2_{iazt}$ into equivalent *CNF* with length $O(1)$
- length of δ^t: $O(p(n))$

V_t codes a configuration

$$
\begin{aligned}
conf(V_t) &= u(s_{z_1,t},\ldots,s_{z_q,t}) \\
&\quad \wedge u(h_{-p(n),t},\ldots,h_{p(n),t}) \\
&\quad \wedge \bigwedge_{-p(n)\le i\le p(n)} u(c_{i,a_1,t},\ldots,c_{i,a_s,t})
\end{aligned}
$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case
$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$
- or δ^2
$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$
This is an implication
- $$\delta^t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma,z\in Z} \delta^t_{iazt}$$
- length $O(p(n))\cdot|\delta_{iazt}|$

Turing machine step

- $$
\begin{aligned}
\tilde{\delta}^2_{iazt} &= \bigvee_{(q,b,L)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i-1,t+1} \wedge s_{q,t+1} \\
&\quad \vee \bigvee_{(q,b,N)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i,t+1} \wedge s_{q,t+1} \\
&\quad \vee \bigvee_{(q,b,R)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i+1,t+1} \wedge s_{q,t+1}
\end{aligned}
$$
- not CNF but length $O(1)$. Obtain δ^1_{iazt} by converting $\tilde{\delta}^2_{iazt}$ into equivalent *CNF* with length $O(1)$
- length of δ^t: $O(p(n))$
- $$\boxed{\begin{aligned} &\varphi(conf(V_t) \wedge conf(V_{t+1}) \wedge \delta^t \wedge D_t) = 1 \\ \Leftrightarrow\ & k_t(\varphi) \vdash k_{t+1}(\varphi)\end{aligned}}$$

initial configuration

- with $w = w[0:n-1]$
$$A(w) = s_{z_0,0} \wedge \Big(\bigwedge_{i\in[-(p(n):-1]\cup[n:p(n)]} c_{i,B,0}\Big) \wedge \Big(\bigwedge_{0\le i\le n-1} c_{i,w[i],0}\Big)$$
- length $O(p(n))$

V_t codes a configuration

$$
\begin{aligned}
conf(V_t) &= u(s_{z_1,t},\dots,s_{z_q,t}) \\
&\wedge u(h_{-p(n),t},\dots,h_{p(n),t}) \\
&\wedge \bigwedge_{-p(n)\le i\le p(n)} u(c_{i,a_1,t},\dots,c_{i,a_s,t})
\end{aligned}
$$

symbols on cells without head don't change in step t

- $$D_t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma} (h_{i,t} \vee (c_{i,a,t} \leftrightarrow c_{i,a,t+1})$$

when M in state z reads a on cell i in step t:

- either this is not the case
$$\delta^1_{iazt} = \overline{c_{i,a,t}} \vee \overline{h_{i,t}} \vee \overline{s_{z,t}}$$
- or δ^2
$$\delta_{iazt} = \delta^1_{iazt} \vee \delta^1_{iazt}$$
This is an implication
- $$\delta^t = \bigwedge_{-p(n)\le i\le p(n),a\in\Sigma,z\in Z} \delta^t_{iazt}$$
- length $O(p(n))\cdot|\delta_{iazt}|$

Turing machine step

- $$
\begin{aligned}
\tilde{\delta}^2_{iazt} &= \bigvee_{(q,b,L)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i-1,t+1} \wedge s_{q,t+1} \\
&\vee \bigvee_{(q,b,N)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i,t+1} \wedge s_{q,t+1} \\
&\vee \bigvee_{(q,b,R)\in\delta(z,a)} c_{i,b,t+1} \wedge h_{i+1,t+1} \wedge s_{q,t+1}
\end{aligned}
$$
- $$\boxed{\begin{aligned} &\varphi(conf(V_t)\wedge conf(V_{t+1})\wedge \delta^t \wedge D_t)=1 \\ \Leftrightarrow\ & k_t(\varphi)\vdash k_{t+1}(\varphi)\end{aligned}}$$

initial configuration

- with $w = w[0:n-1]$
$$A(w) = s_{z_0,0} \wedge (\bigwedge_{i\in[-(p(n):-1]\cup[n:p(n)]} c_{i,B,0}) \wedge (\bigwedge_{0\le i\le n-1} c_{i,w[i],0})$$
- length $O(p(n))$
- $$\boxed{\begin{aligned} &\varphi(conf(V_0)\wedge A(w))=1 \\ \Leftrightarrow\ & k_0(\varphi) = B^{p(n)}z_0wB^{p(n)-n+1}\end{aligned}}$$

polynomial reduction

- $$\begin{aligned} f(w) &= \bigwedge_{0 \le t \le p(n)} (conf(V_t)) \\ &\wedge \bigwedge_{0 \le t \le p(n)} (D_t \wedge \delta_t) \\ &\wedge A(w) \\ &\wedge \Big(\bigvee_{z_a \in Z_A} s_{z_a, p(n)} \Big) \end{aligned}$$

- length: $O(p^3(n))$

polynomial reduction

- $$\begin{aligned} f(w) \quad = \quad & \bigwedge_{0\le t\le p(n)} (conf(V_t)) \\ & \wedge \bigwedge_{0\le t\le p(n)} (D_t \wedge \delta_t) \\ & \wedge A(w) \\ & \wedge (\bigvee_{z_a\in Z_A} s_{z_a,p(n)}) \end{aligned}$$

- length: $O(p^3(n))$

- with binary or decimal indices of variables $O(p^3(n)\log n)$

- $$\boxed{\exists\varphi.\ \varphi(f(w)) = 1 \quad \Leftrightarrow \quad M \text{ accepts } w}$$

Lemma 12. $3-SAT$ *is NP-complete*

Proof. Replace long clauses c clauses in $f(w)$ by equivalent expressions C in $3-CNF$ as indicated by lemma 10. □

6 More NP-complete problems

- This is a universe; thousands are known.
- M. Garey and D. Johnson: Computers and Intractability. W.H. Freeman and co, San Francisco 1979
- great stuff for a (not so difficult) seminar.
- here: only clique problem

6 More NP-complete problems

- This is a universe; thousands are known.
- M. Garey and D. Johnson: Computers and Intractability. W.H. Freeman and co, San Francisco 1979
- great stuff for a (not so difficult) seminar.
- here: only clique problem

cliques

- Let $G = (V, E)$ be undirected graph and $k \in \mathbb{N}$
- $V' \subseteq V$ is k-clique if
  1. $|V'| = k$ and
  2. $\forall u, v \in V'. \quad \{u, v\} \in E$

clique problem

$$CLIQUE = \{code(G, k) \;:\; G \text{ has } k\text{-clique}\}$$

6 More NP-complete problems

- This is a universe; thousands are known.
- M. Garey and D. Johnson: Computers and Intractability. W.H. Freeman and co, San Francisco 1979
- great stuff for a (not so difficult) seminar.
- here: only clique problem

cliques

- Let $G = (V,E)$ be undirected graph and $k \in \mathbb{N}$
- $V' \subseteq V$ is k-clique if
  1. $|V'| = k$ and
  2. $\forall u, v \in V'. \quad \{u,v\} \in E$

clique problem

$$CLIQUE = \{code(G,k) \ : \ G \text{ has } k\text{-clique}\}$$

Lemma 13. *$CLIQUE$ is NP complete*

$CLIQUE \in NP$**:** trivial. Guess clique and verify.

6 More NP-complete problems

- This is a universe; thousands are known.
- M. Garey and D. Johnson: Computers and Intractability. W.H. Freeman and co, San Francisco 1979
- great stuff for a (not so difficult) seminar.
- here: only clique problem

cliques

- Let $G = (V, E)$ be undirected graph and $k \in \mathbb{N}$
- $V' \subseteq V$ is k-clique if
  1. $|V'| = k$ and
  2. $\forall u, v \in V'. \quad \{u, v\} \in E$

clique problem

$$CLIQUE = \{code(G, k) \ : \ G \text{ has } k\text{-clique}\}$$

Lemma 13. *$CLIQUE$ is NP complete*

$CLIQUE \in NP$**:** trivial. Guess clique and verify.

$CLIQUE$ **is** NP**-hard:** we show

$$3-SAT \leq_p CLIQUE$$

- given
  1. conjunctive normal form
  $$A = c_1 \wedge \ldots, \wedge c_t \ \in 3-CNF$$
  2. literals $L_{i,j}$
  $$c_i = (L_{i,1} \vee \ldots \vee L_{i,n(i)}) \quad n(i) \leq 3$$
- we construct garph $G = (V, E)$ and clique size
  $$k = t = \#\text{ clauses}$$
  such that
  $$A \ \in 3-SAT \ \leftrightarrow (G, k) \in CLIQUE$$

***CLIQUE* is *NP*-hard:** we show

$$3-SAT \leq_p CLIQUE$$

- given
  1. conjunctive normal form
  $$A = c_1 \wedge \ldots, \wedge c_t \ \in 3-CNF$$
  2. literals $L_{i,j}$
  $$c_i = (L_{i,1} \vee \ldots \vee L_{i,n(i)}) \quad n(i) \leq 3$$
- we construct garph $G = (V,E)$ and clique size
$$k = t = \#\,\text{clauses}$$

  such that
$$A \ \in 3-SAT \ \leftrightarrow (G,k) \in CLIQUE$$

nodes: the indices (i,j) of literals $L_{i,j}$

$$V = \{(i,j) \ : \ 1 \leq i \leq t \ , \ 1 \leq j \leq n(i)\}$$

edges: between indices of literals

$$E = \{\{(i,j),(i',j')\} \ : \ i \neq i' \ , L_{i,j} \neq \overline{L_{i',j'}}\}$$

Interpretation:

1. $i \neq i'$: in different clauses
2. $L_{i,j} \neq \overline{L_{i',j'}}$: literals can be satisfied simultaneously

example:

$$A = (x_1 \vee \overline{x_2}) \ \wedge \ (x_1 \vee \overline{x_3}) \ \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

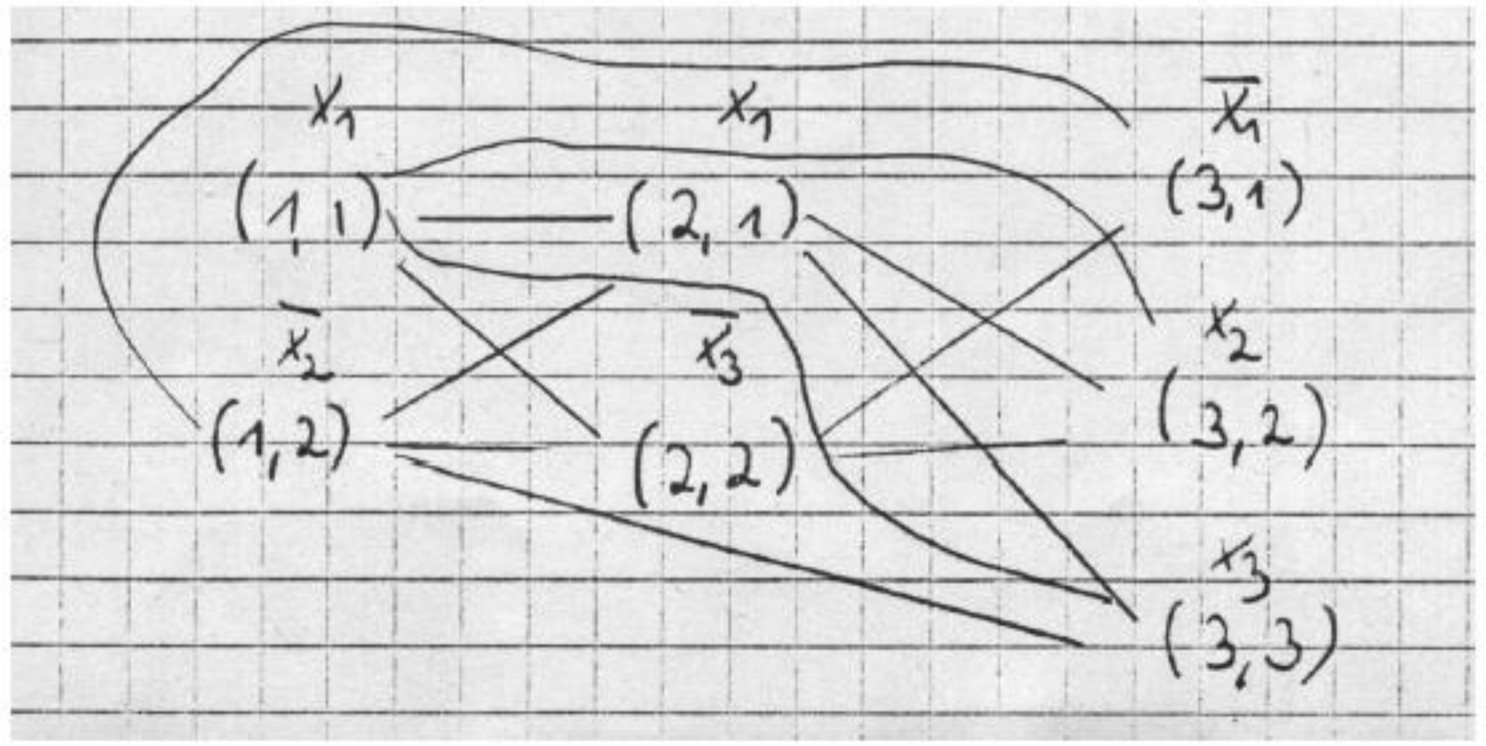

Figure 2: the graph G generated for A

- claim A satisfiable iff G has a t-clique.

$CLIQUE$ **is** NP**-hard:** we show

$$3-SAT \leq_p CLIQUE$$

- given
  1. conjunctive normal form
  $$A = c_1 \wedge \ldots, \wedge c_t \in 3-CNF$$
  2. literals $L_{i,j}$
  $$c_i = (L_{i,1} \vee \ldots \vee L_{i,n(i)}) \quad n(i) \leq 3$$
- we construct garph $G = (V, E)$ and clique size
$$k = t = \#\text{ clauses}$$

such that
$$A \in 3-SAT \leftrightarrow (G, k) \in CLIQUE$$

nodes: the indices (i, j) of literals $L_{i,j}$

$$V = \{(i, j) \ : \ 1 \leq i \leq t \ , \ 1 \leq j \leq n(i)\}$$

edges: between indices of literals

$$E = \{\{(i, j), (i', j')\} \ : \ i \neq i' \ , L_{i,j} \neq \overline{L_{i',j'}}\}$$

Interpretation:

1. $i \neq i'$: in different clauses
2. $L_{i,j} \neq \overline{L_{i',j'}}$: literals can be satisfied simultaneously

example:

$$A = (x_1 \vee \overline{x_2}) \ \wedge \ (x_1 \vee \overline{x_3}) \ \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

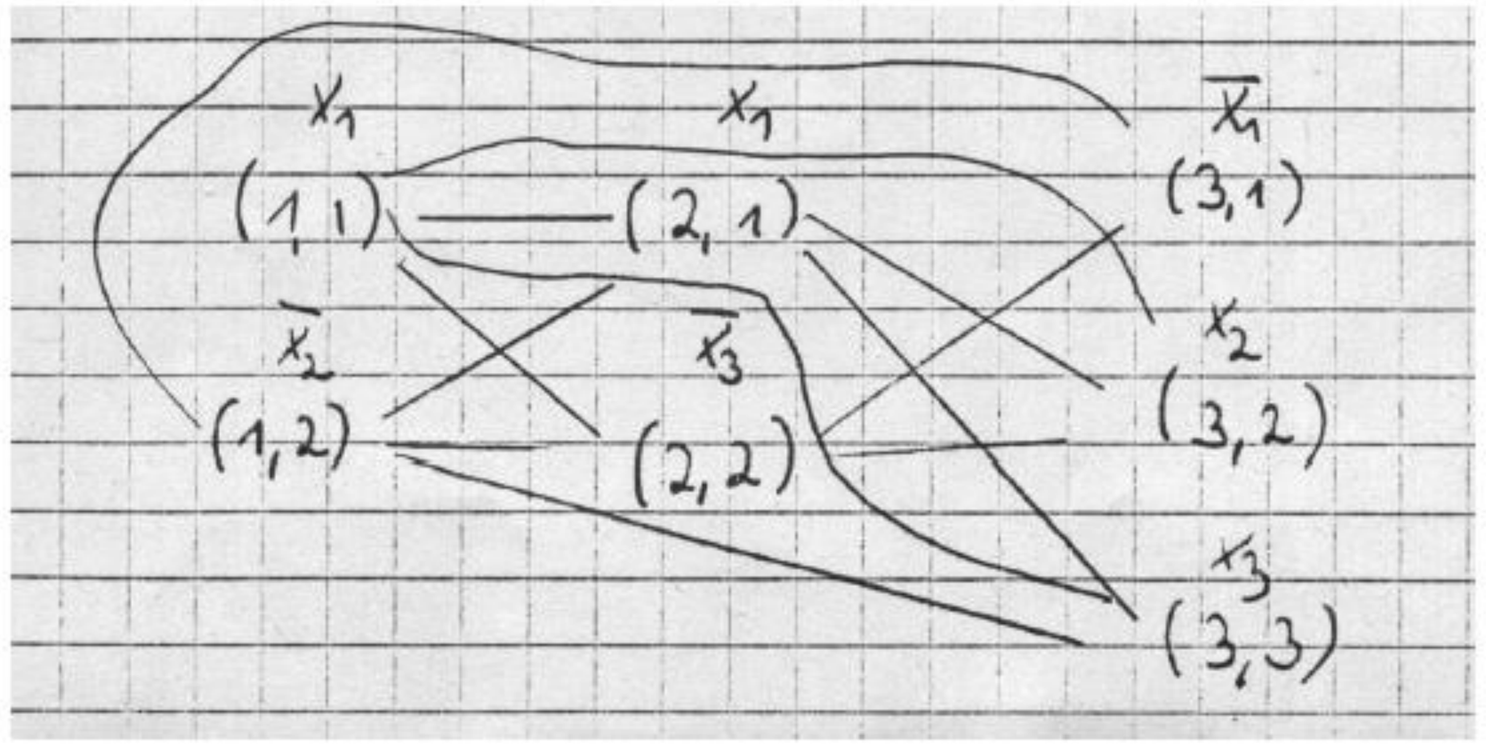

Figure 2: the graph G generated for A

- claim A satisfiable iff G has a t-clique.

$\rightarrow$:

$$\varphi(A) = 1 \ \rightarrow \forall i \exists j(i). \ \varphi(L_{i,j(i)}) = 1$$

$$\begin{aligned} i \neq i' \quad &\rightarrow \quad L_{i,j(i)} \neq \overline{L_{i',j(i')}} \\ &\rightarrow \quad \{(i, j(i)), (i, j(i'))\} \in E \\ &\rightarrow \quad \{(i, j(i)) \ : \ 1 \leq i \leq t\} \quad \text{is } t-\text{clique} \end{aligned}$$

CLIQUE **is** *NP***-hard:** we show

$$3-SAT \leq_p CLIQUE$$

- given
  1. conjunctive normal form
  $$A = c_1 \wedge \ldots, \wedge c_t \ \in 3-CNF$$
  2. literals $L_{i,j}$
  $$c_i = (L_{i,1} \vee \ldots \vee L_{i,n(i)}) \quad n(i) \leq 3$$
- we construct garph $G = (V,E)$ and clique size
$$k = t = \#\text{ clauses}$$

such that
$$A \ \in 3-SAT \ \leftrightarrow (G,k) \in CLIQUE$$

nodes: the indices (i,j) of literals $L_{i,j}$

$$V = \{(i,j) \ : \ 1 \leq i \leq t \ , \ 1 \leq j \leq n(i)\}$$

edges: between indices of literals

$$E = \{\{(i,j),(i',j')\} \ : \ i \neq i' \ , L_{i,j} \neq \overline{L_{i',j'}}\}$$

Interpretation:

1. $i \neq i'$: in different clauses
2. $L_{i,j} \neq \overline{L_{i',j'}}$: literals can be satisfied simultaneously

example:

$$A = (x_1 \vee \overline{x_2}) \ \wedge \ (x_1 \vee \overline{x_3}) \ \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

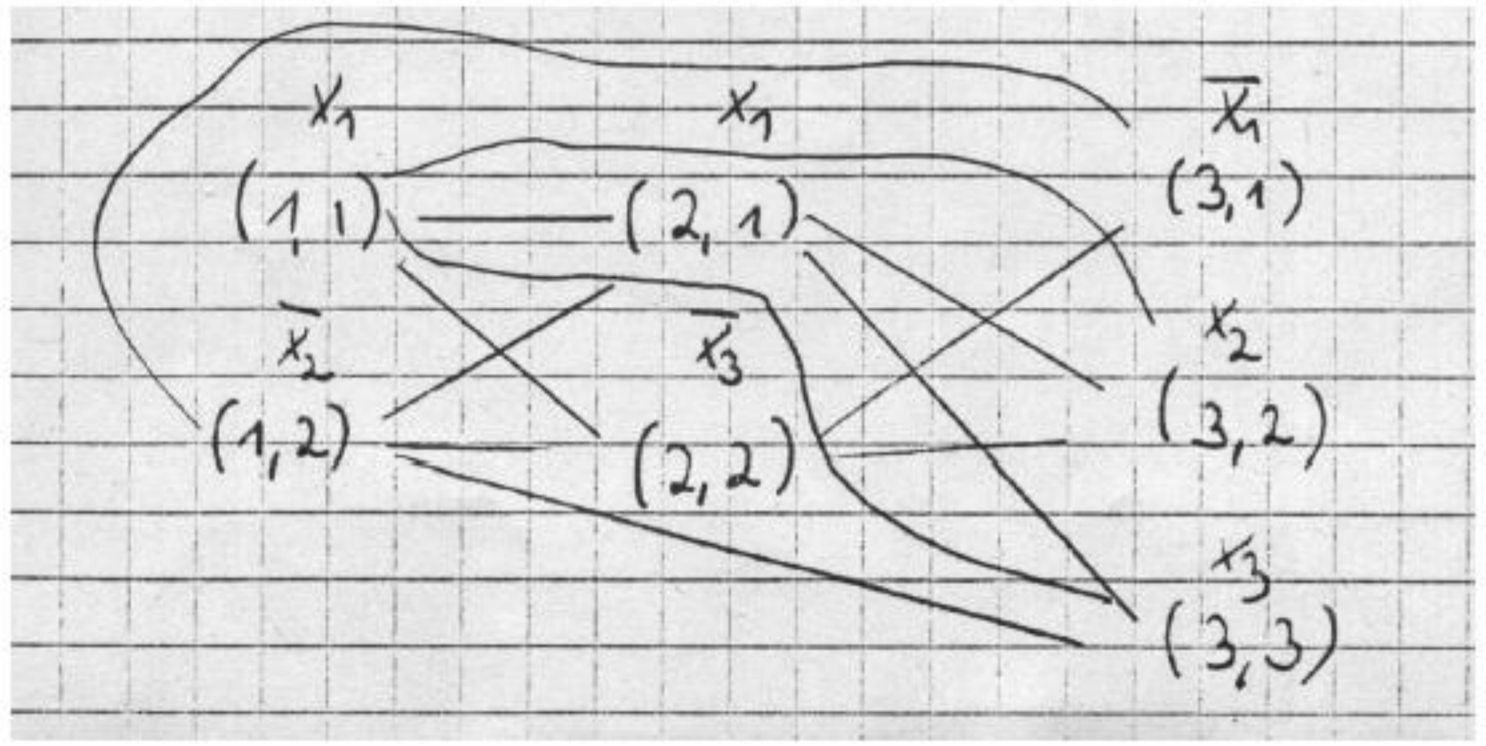

Figure 2: the graph G generated for A

- claim A satisfiable iff G has a t-clique.

$\leftarrow$: let
$$\{(i_r, j_r) \ : \ 1 \leq i \leq r\}$$
be t-clique
$$i_r \neq i_{r'} \quad \text{for } r \neq r'$$
assign
$$\varphi(L_{i_r,j_r}) = 1 \quad 1 \leq r \leq t$$
well defined as
$$L_{i_r,j_r} \neq \overline{L_{i_{r'},j_{r'}}}$$
assigning remaining variables in each clause arbitrary values does not change $\varphi(A) = 1$