Complexity Theory

Jan Křetínský

Chair for Foundations of Software Reliability and Theoretical Computer Science
Technical University of Munich

Summer 2016

Based on slides by Jörg Kreiker

Lecture 5

NP-completeness (2)

Teaser

A regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

What is the computational complexity of

- deciding whether two regular expressions are equivalent, that is $\mathcal{L}(r_1) = \mathcal{L}(r_2)$?
- deciding whether a regular expression is universal, that is $\mathcal{L}(r) = \{0, 1\}^*$?
- deciding the same for star-free regular expressions?

Agenda

- Cook-Levin
- SAT demo
- see old friends
  - $0/1-\mathrm{ILP}$
  - Indset
  - $3-\mathrm{Coloring}$
- teaser update

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP** ✓

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP** ✓
- 3SAT is **NP**-hard
  - choose $L \in$ **NP** arbitrary, $L \subseteq \{0,1\}^*$
  - find reduction f from L to 3SAT

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP** ✓
- 3SAT is **NP**-hard
  - choose $L \in$ **NP** arbitrary, $L \subseteq \{0,1\}^*$
  - find reduction f from L to 3SAT
    - $\forall x \in \{0,1\}^*$: $x \in L \Leftrightarrow f(x) \in$ 3SAT iff φ_x is satisfiable
    - f is polynomial time computable

TMs for L and f

$L \in$ **NP** iff there exists a TM M that runs in time T and there is a polynomial p such that

$$\forall x \in L \; \exists u \in \{0,1\}^{p(|x|)} \; M(x,u) = 1 \Leftrightarrow x \in L$$

TMs for L and f

$L \in$ **NP** iff there exists a TM M that runs in time T and there is a polynomial p such that

$$\forall x \in L \; \exists u \in \{0,1\}^{p(|x|)} \; M(x,u) = 1 \Leftrightarrow x \in L$$

Assumptions

- fix $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ arbitrary
- $m = n + p(n)$
- $M = (\Gamma, Q, \delta)$
- M is oblivious
- M has two tapes
- define TM M_f that takes M, T, p, x and outputs φ_x

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps
2. for each $1 \leq i \leq T(n+p(n))$ store
   - $inputpos(i)$: position of input head after i steps
   - $prev(i)$: previous step when work head was here (default 1)

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps
2. for each $1 \leq i \leq T(n+p(n))$ store
   - $inputpos(i)$: position of input head after i steps
   - $prev(i)$: previous step when work head was here (default 1)
3. compute and output φ_x

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps
2. for each $1 \leq i \leq T(n+p(n))$ store
   - $inputpos(i)$: position of input head after i steps
   - $prev(i)$: previous step when work head was here (default 1)
3. compute and output φ_x

It does all this in time polynomial in n!

Variables of φ_x

- $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$

Variables of φ_x

- $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape

Variables of φ_x

- $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \ldots, y_n$ determined by x

Variables of φ_x

- $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \ldots, y_n$ determined by x
  - $y_{n+1}, \ldots y_{n+p(n)}$ will be certificate

Variables of φ_x

- $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
    - to encode the read-only input tape
    - $y_1, \ldots, y_n$ determined by x
    - $y_{n+1}, \ldots y_{n+p(n)}$ will be certificate
- $$\begin{matrix} z_1 & z_2 & \cdots & z_{c-1} & z_c \\ z_{c+1} & z_{c+2} & \cdots & z_{2c-1} & z_{2c} \\ \vdots & & & & \vdots \\ z_{c(T(m)-1)+1} & & & & z_{cT(m)} \end{matrix}$$

Variables of φ_x

- $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \ldots, y_n$ determined by x
  - $y_{n+1}, \ldots y_{n+p(n)}$ will be certificate
- $$\begin{array}{lllll} z_1 & z_2 & \ldots & z_{c-1} & z_c \\ z_{c+1} & z_{c+2} & \ldots & z_{2c-1} & z_{2c} \\ \vdots & & & & \vdots \\ z_{c(T(m)-1)+1} & & & & z_{cT(m)} \end{array}$$
  - each row a snapshot
  - needs $c-2$ bits to encode state q (independent of x)

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

- first snapshot s_1 is $\langle q_{start}, \triangleright, \square \rangle$

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

- first snapshot s_1 is $\langle q_{start}, \triangleright, \square \rangle$
- last snapshot $s_{T(m)}$ has state q_{halt} and ouputs 1

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

- first snapshot s_1 is $\langle q_{start}, \triangleright, \square \rangle$
- last snapshot $s_{T(m)}$ has state q_{halt} and ouputs 1
- s_{i+1} computed from
  - δ
  - s_i
  - $y_{inputpos(i+1)}$
  - $s_{prev(i+1)}$

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$

$$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \square \rangle$
   $\rightarrow$ size $\mathcal{O}(c2^c)$ (CNF, independent of $|x|$)

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \square \rangle$
   $\rightarrow$ size $\mathcal{O}(c2^c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   - $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \Box \rangle$
   - $\rightarrow$ size $O(c2^c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   - $\rightarrow$ analogous
4. relate $z_{ci+1}, \ldots, z_{c(i+1)}$ (snapshot s_{i+1}) with
   - $y_{inputpos(i+1)}$
   - $z_{c(i-1)+1}, \ldots, z_{ci-2}$ (state of snapshot s_i)
   - $z_{prev(i)}$ (next work tape symbol, from snapshot $s_{prev(i)}$)

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \square \rangle$
   $\rightarrow$ size $\mathcal{O}(c2^c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous
4. relate $z_{ci+1}, \ldots, z_{c(i+1)}$ (snapshot s_{i+1}) with
   - $y_{inputpos(i+1)}$
   - $z_{c(i-1)+1}, \ldots, z_{ci-2}$ (state of snapshot s_i)
   - $z_{prev(i)}$ (next work tape symbol, from snapshot $s_{prev(i)}$)
   - CNF formula over $2c$ variables, size $\mathcal{O}(c2^{2c})$

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_n$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \Box \rangle$
   $\rightarrow$ size $\mathcal{O}(c2^c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous
4. relate $z_{ci+1}, \ldots, z_{c(i+1)}$ (snapshot s_{i+1}) with
   - $y_{inputpos(i+1)}$
   - $z_{c(i-1)+1}, \ldots, z_{ci-2}$ (state of snapshot s_i)
   - $z_{prev(i)}$ (next work tape symbol, from snapshot $s_{prev(i)}$)
   - CNF formula over $2c$ variables, size $\mathcal{O}(c2^{2c})$

Polynomial in n!

Stop!

- $|\varphi_x|$ polynomial in n
- if φ_x is satisfiable, the satisfying assignment yields certificate $y_{n+1}, \ldots y_{n+p(n)}$
- if a certificate exists in $\{0,1\}^{p(n)}$, we get a satisfying assignment
- M_f can output φ_x in polynomial time

$\Rightarrow$ reduction

Stop!

- $|\varphi_x|$ polynomial in n
- if φ_x is satisfiable, the satisfying assignment yields certificate $y_{n+1}, \ldots y_{n+p(n)}$
- if a certificate exists in $\{0,1\}^{p(n)}$, we get a satisfying assignment
- M_f can output φ_x in polynomial time

$\Rightarrow$ reduction

- but: not to 3SAT

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$
$$\leadsto$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\rightsquigarrow$$

$$\begin{array}{lcccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \end{array}$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\leadsto$$

$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \end{array}$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\rightsquigarrow$$

$$\begin{array}{lccccc}
 & (u_1 & \vee & u_2 & \vee & x_1) \\
\wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\
\wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \\
\ldots & & & & & \\
\wedge & (\overline{x_{k-2}} & \vee & u_{k-1} & \vee & u_k)
\end{array}$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\rightsquigarrow$$

$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \\ \ldots & & & & & \\ \wedge & (\overline{x_{k-2}} & \vee & u_{k-1} & \vee & u_k) \end{array}$$

Each clause with k variables transformed into equivalent $k - 2$ 3-clauses with $2k - 2$ variables. All x_i fresh.

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\rightsquigarrow$$

$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \\ \ldots & & & & & \\ \wedge & (\overline{x_{k-2}} & \vee & u_{k-1} & \vee & u_k) \end{array}$$

Each clause with k variables transformed into equivalent $k - 2$ 3-clauses with $2k - 2$ variables. All x_i fresh.

Example. $x \vee \overline{y} \vee \overline{z} \vee w$ becomes $x \vee \overline{y} \vee q$ and $\overline{q} \vee \overline{z} \vee w$.

What you need to remember

- for each $L \in$ **NP** take TM M deciding L in polynomial time
- define TM M_f computing a reduction to formula φ_x for each input
- due to obliviousness M_f pre-computes head positions and every computation takes time $T(n + p(n))$ steps
- and is a sequence of snapshots $\langle q, 0, 1 \rangle$
- φ has four parts
  - correct input x, u with u being the certificate
  - correct starting snapshot
  - correct halting snapshot
  - how to go from s_i to s_{i+1}
- finally: CNF transformed to 3CNF

Agenda

- Cook-Levin ✓
- SAT demo
- see old friends
  - $0/1-\mathrm{ILP}$
  - Indset
  - $3-\mathrm{Coloring}$
- teaser update

So 3SAT is intractable?

- if **P** $\neq$ **NP**, no polynomial time algorithm for SAT
- contrapositive: if you find one, you prove **P** $=$ **NP**
- every problem in **NP** solvable by exhaustive search for certificates
- which implies **NP** $\subseteq$ **PSPACE** (try each possible re-using space)

SAT **is easy!**

- well-researched problem
- has its own conference
- 1000s of tools, academic and commercial
- extremely useful for modelling
  - verification
  - planning and scheduling
  - AI
  - games (Sudoku!)
- useful for reductions due to low combinatorial complexity
- `satlive.org`: solvers, jobs, competitions

Demo

- `www.sat4j.org`
- two termination problems from string/term-rewriting
- 10000s of variables, millions of clauses
- solvable in a few seconds!

Agenda

- Cook-Levin ✓
- SAT demo ✓
- see old friends
    - 0/1−ILP
    - Indset
    - 3−Coloring
- teaser update

More reductions from 3SAT

We will now describe reductions from 3SAT to

- $0/1-\mathrm{ILP}$: the set of satisfiable sets of integer linear programs with boolean solutions
- $\mathrm{Indset} = \{\langle G, k\rangle \mid G$ has independent set of size at least $k\}$
- $3-\mathrm{Coloring} = \{G \mid G$ is 3-colorable$\}$

This establishes **NP**-hardness for all of the problems. Of course, they are easily in **NP** as well, hence complete.

3SAT $\leq_p$ 0/1−ILP

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$3SAT \leq_p 0/1-ILP$

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$$\begin{aligned} x + (1-y) + z &\geq 1 \\ x + (1-y) + (1-z) &\geq 1 \\ (1-x) + (1-y) + w &\geq 1 \\ (1-x) + y + (1-w) &\geq 1 \end{aligned}$$

3SAT $\leq_p$ 0/1−ILP

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$$\begin{array}{rcl} x + (1-y) + z & \geq & 1 \\ x + (1-y) + (1-z) & \geq & 1 \\ (1-x) + (1-y) + w & \geq & 1 \\ (1-x) + y + (1-w) & \geq & 1 \end{array}$$

- $f(x) = x$
- $f(\overline{x}) = (1-x)$
- $f(u_1 \vee \ldots \vee u_k) = f(u_1) + \ldots + f(u_k) \geq 1$

3SAT $\leq_p$ 0/1−ILP

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$$\begin{aligned} x + (1-y) + z &\geq 1 \\ x + (1-y) + (1-z) &\geq 1 \\ (1-x) + (1-y) + w &\geq 1 \\ (1-x) + y + (1-w) &\geq 1 \end{aligned}$$

- $f(x) = x$
- $f(\overline{x}) = (1-x)$
- $f(u_1 \vee \ldots \vee u_k) = f(u_1) + \ldots + f(u_k) \geq 1$
- linear reduction
- φ satisfiable iff $f(\varphi)$ has boolean solution

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \to \{0, 1\}, C_i \text{ holds under assignment } a_i\}$

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \to \{0, 1\}, C_i \text{ holds under assignment } a_i\}$
- edges denote conflicting assignments
  - $E = \{\{C_i^a, C_{i'}^{a'}\} \mid i \neq i' \in [m], \exists x. a(x) \neq a'(x)\}$

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \rightarrow \{0, 1\}, C_i \text{ holds under assignment } a_i\}$
- edges denote conflicting assignments
  - $E = \{\{C_i^{a}, C_{i'}^{a'}\} \mid i \neq i' \in [m], \exists x. a(x) \neq a'(x)\}$
- G has $7m$ nodes and $\mathcal{O}(m^2)$ edges and can be computed in polynomial time

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \to \{0, 1\}, C_i \text{ holds under assignment } a_i\}$
- edges denote conflicting assignments
  - $E = \{\{C_i^a, C_{i'}^{a'}\} \mid i \neq i' \in [m], \exists x.a(x) \neq a'(x)\}$
- G has $7m$ nodes and $\mathcal{O}(m^2)$ edges and can be computed in polynomial time

3SAT $\leq_p$ Indset

- φ is satisfiable

$\Rightarrow$ exists assignment $a : X \to \{0, 1\}$ that makes φ true

$\Rightarrow$ a makes every clause true

$\Rightarrow$ $\{C_i^{a|vars(i)} \mid 1 \leq i \leq m\}$ is an independent set of size m

3SAT $\leq_p$ Indset

- φ is satisfiable

$\Rightarrow$ exists assignment $a : X \to \{0, 1\}$ that makes φ true

$\Rightarrow$ a makes every clause true

$\Rightarrow$ $\{C_i^{a|vars(i)} \mid 1 \leq i \leq m\}$ is an independent set of size m

- G has an independent set of size m

$\Rightarrow$ ind. set covers all clauses

$\Rightarrow$ ind. set yields composable, partial assignments per clause

$\Rightarrow$ φ is satisfiable

3SAT $\leq_p$ 3−Coloring

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$

3SAT $\leq_p$ 3−Coloring

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$
- V is the union of
  - $X \cup \overline{X}$ to capture assignments
  - special nodes $\{u, v\}$
  - one little house per clause with 5 nodes: $\{v_{ij}, a_i, b_i \mid i \in [m], j \in [3]\}$

3SAT $\leq_p$ 3−Coloring

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$
- V is the union of
  - $X \cup \overline{X}$ to capture assignments
  - special nodes $\{u, v\}$
  - one little house per clause with 5 nodes: $\{v_{ij}, a_i, b_i \mid i \in [m], j \in [3]\}$
- E comprised of
  - edge $\{u, v\}$
  - for each literal in each clause, a connection to the assignment graph: $\{\{u_{ij}, v_{ij}\} \mid i \in [m], j \in [3]\}$
  - house edges: $\{\{v, a_i\}, \{v, b_i\}, \{v_{i1}, a_i\}, \{v_{i1}, b_i\}, \{v_{i2}, a_i\}, \{v_{i2}, v_{i3}\}, \{v_{i2}, b_i\} \mid i \in [m]\}$
- G has $2n + 5m + 2$ nodes and $\mathcal{O}(m^2)$ edges and can be computed in polynomial time
- three colors: $\{red, true, false\}$

3SAT $\leq_p$ 3−Coloring

- φ is satisfiable,

$\Rightarrow$ there is an assignment $a : X \to \{0, 1\}$ that makes every clause true

$\Rightarrow$ coloring u red, v false, and x true iff $a(x) = 1$ leads to a correct 3-coloring

3SAT $\leq_p$ 3−Coloring

- φ is satisfiable,

$\Rightarrow$ there is an assignment $a : X \to \{0, 1\}$ that makes every clause true

$\Rightarrow$ coloring u red, v false, and x true iff $a(x) = 1$ leads to a correct 3-coloring

- G is 3-colorable
- wlog. assume u is red and v is false
- assume there is a clause j such that all literals are colored false

$\Rightarrow$ v_{j2} and v_{j3} are colored true and red

$\Rightarrow$ a_j and b_j are colored true and red

$\Rightarrow$ v_{j1} colored false, which is a contradiction, because it is connected to a false literal

What have you learnt?

- SAT is **NP**-complete
- SAT is practically feasible
- SAT has lots of academic and industrial applications
- SAT can be reduced to independent set, 3-coloring and boolean ILP, which makes those **NP**-hard
- up next: **coNP**, Ladner

Can you guess now?

What is the computational complexity of

- deciding whether two regular expressions are equivalent, that is $\mathcal{L}(r_1) = \mathcal{L}(r_2)$?
- deciding whether a regular expression is universal, that is $\mathcal{L}(r) = \{0, 1\}^*$?
- deciding the same for star-free regular expressions?

Can you guess now?

What is the computational complexity of

- deciding whether two regular expressions are equivalent, that is $\mathcal{L}(r_1) = \mathcal{L}(r_2)$?
- deciding whether a regular expression is universal, that is $\mathcal{L}(r) = \{0, 1\}^*$?
- deciding the same for star-free regular expressions?
- what about the set of formulas, for which all assignments satisfy? certificates?

solution tomorrow