Foundations of Complexity Theory

Lecture 5: Reductions & Complete Problems I

November 11, 2026

Next Dates

Lecture (Wed 1:15pm)
Nov 11th Lecture 5: Reductions & complete problems I
18th (public holiday)
25th Lecture 6: Complete problems II
Dec 2nd (dies academicus)
9th Lecture 7: NP-complete problems
16th Lecture 8: Complete problems III
23rd

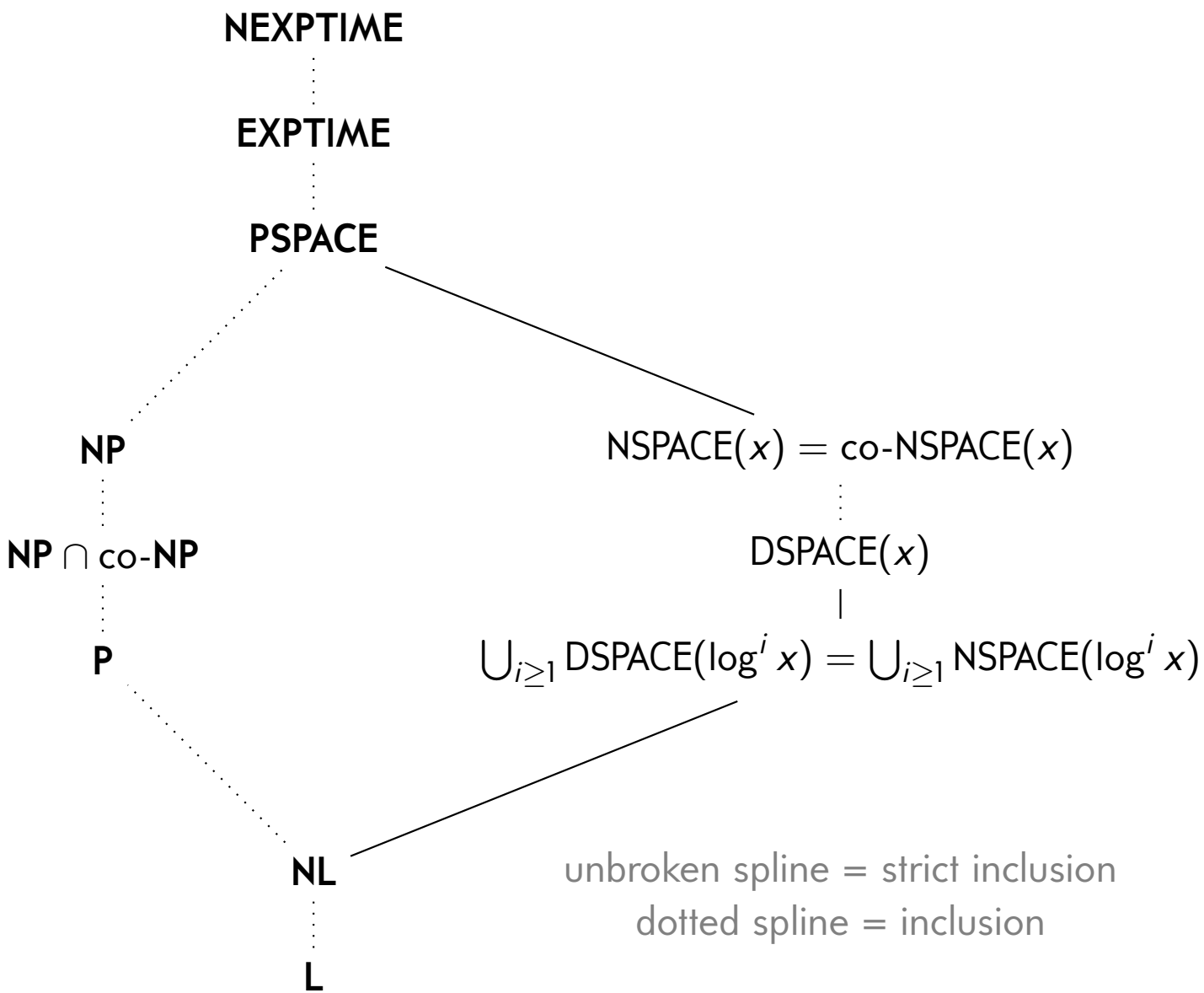
Lecture (Wed 1:15pm)
Dec 30th
Jan 6th Lecture 9: Boolean circuits I
13th Lecture 10: Boolean circuits II
20th Lecture 11: Razborov's theorem I
27th
Feb 3rd Lecture 12: Razborov's theorem II

Overview

Contents

1. Basic complexity classes & their relations
2. Reductions & complete problems
3. Boolean circuits & Razborov's theorem
4. Randomization & interactive proof systems
5. Polynomial primality test

Complexity Classes



Reductions

§5.1 Definition (Reduction)

Let $L \subseteq \Sigma^*$ and $L' \subseteq (\Sigma')^*$ be problems. Reduction of L to L' is total computable function $f : \Sigma^* \to (\Sigma')^*$ such that for all $w \in \Sigma^*$

$$w \in L \quad \text{iff} \quad f(w) \in L'$$

Notes

- Classic notion of computability
- Decision algorithm for L' translates to decision algorithm for L via reduction of L to L'
  (on input w compute $f(w)$ and decide $f(w) \in L'$)

Reductions

Illustration

- GAP is reducible to SAT

  Let (V, E) finite input graph and $s, t \in V$ given start and target vertex.
  If t is reachable from s (which is certainly computable),
  then reduction f yields satisfiable formula $f(\langle V, E, s, t \rangle) = x$ and
  otherwise unsatisfiable formula $x \wedge \neg x$

Reductions

Illustration

- GAP is reducible to SAT

  Let (V, E) finite input graph and $s, t \in V$ given start and target vertex.
  If t is reachable from s (which is certainly computable),
  then reduction f yields satisfiable formula $f(\langle V, E, s, t \rangle) = x$ and
  otherwise unsatisfiable formula $x \wedge \neg x$

- SAT is reducible to $\mathrm{ODD} = \{n \in \mathbb{N} \mid n \text{ odd}\}$

  Let F input formula. Then

$$f(F) = \begin{cases} 1 & \text{if } F \text{ is satisfiable} \\ 0 & \text{otherwise} \end{cases}$$

  is computable reduction

Reductions

§5.2 Definition (Polynomial-time & log-space reduction)

Let $L \subseteq \Sigma^*$ and $L' \subseteq (\Sigma')^*$ and $f : \Sigma^* \to (\Sigma')^*$ reduction of L to L'.
Then f is

- polynomial-time reduction if f computable by polynomial time-bounded deterministic TM

Reductions

§5.2 Definition (Polynomial-time & log-space reduction)

Let $L \subseteq \Sigma^*$ and $L' \subseteq (\Sigma')^*$ and $f : \Sigma^* \to (\Sigma')^*$ reduction of L to L'.
Then f is

- polynomial-time reduction if f computable by polynomial time-bounded deterministic TM
- log-space reduction if f computable by logarithmic space-bounded deterministic TM

Reductions

§5.2 Definition (Polynomial-time & log-space reduction)

Let $L \subseteq \Sigma^*$ and $L' \subseteq (\Sigma')^*$ and $f : \Sigma^* \to (\Sigma')^*$ reduction of L to L'.
Then f is

- polynomial-time reduction if f computable by polynomial time-bounded deterministic TM
- log-space reduction if f computable by logarithmic space-bounded deterministic TM

Notes

- Log-space reduction f is polynomial-time reduction
  (i.e. there exists $i \in \mathbb{N}_+$ such that $|f(w)| \leq |w|^i$ for all $w \in \Sigma^*$)
- Converse only holds if $\mathbf{L} = \mathbf{P}$

Reductions

§5.2 Definition (Polynomial-time & log-space reduction)

Let $L \subseteq \Sigma^*$ and $L' \subseteq (\Sigma')^*$ and $f : \Sigma^* \to (\Sigma')^*$ reduction of L to L'.
Then f is

- polynomial-time reduction if f computable by polynomial time-bounded deterministic TM
- log-space reduction if f computable by logarithmic space-bounded deterministic TM

Notes

- Log-space reduction f is polynomial-time reduction
  (i.e. there exists $i \in \mathbb{N}_+$ such that $|f(w)| \leq |w|^i$ for all $w \in \Sigma^*$)
- Converse only holds if $\mathbf{L} = \mathbf{P}$
- Log-space reductions allow finer classification and applicable to classes inside **P**

Reductions

Illustration

- Presented reduction of GAP to SAT is polynomial-time reduction, but no log-space reduction (unless **L** = **NL**)

Reductions

Illustration

- Presented reduction of GAP to SAT is polynomial-time reduction, but no log-space reduction (unless **L** = **NL**)

- Presented reduction of SAT to ODD is no polynomial-time reduction (unless **P** = **NP**) and no log-space reduction (unless **L** = **NP**)

Reductions

§5.3 Proposition

Let $L' \in \mathbf{P}$ and f polynomial-time reduction of L to L'. Then $L \in \mathbf{P}$

Reductions

§5.3 Proposition

Let $L' \in \mathbf{P}$ and f polynomial-time reduction of L to L'. Then $L \in \mathbf{P}$

Proof

Let $L' \in \mathrm{DTIME}(x^i)$ and f be computable by x^ℓ-time-bounded deterministic TM for $i, \ell \in \mathbb{N}_+$.

Reductions

§5.3 Proposition

Let $L' \in \mathbf{P}$ and f polynomial-time reduction of L to L'. Then $L \in \mathbf{P}$

Proof

Let $L' \in \mathrm{DTIME}(x^i)$ and f be computable by x^ℓ-time-bounded deterministic TM for $i, \ell \in \mathbb{N}_+$. Thus we can compute $f(w)$ in time $(n+2)^\ell$ for all inputs w of length n. In particular, $|f(w)| \leq (n+2)^\ell$ and thus we can decide $f(w) \in L'$ in time $\left((n+2)^\ell + 2\right)^i$.

Reductions

§5.3 Proposition

Let $L' \in \mathbf{P}$ and f polynomial-time reduction of L to L'. Then $L \in \mathbf{P}$

Proof

Let $L' \in \mathrm{DTIME}(x^i)$ and f be computable by x^ℓ-time-bounded deterministic TM for $i, \ell \in \mathbb{N}_+$. Thus we can compute $f(w)$ in time $(n+2)^\ell$ for all inputs w of length n. In particular, $|f(w)| \leq (n+2)^\ell$ and thus we can decide $f(w) \in L'$ in time $\left((n+2)^\ell + 2\right)^i$. Altogether we will need time $(n+2)^\ell + \left((n+2)^\ell + 2\right)^i$ for decision whether $w \in L$, which is still polynomial. □

Reductions

Notation

- Let L and L' problems (i.e. languages)
- We write $L \preceq_{\mathbf{P}} L'$
  if there exists polynomial-time reduction of L to L'

Reductions

Notation

- Let L and L' problems (i.e. languages)
- We write $L \preceq_{\mathbf{P}} L'$
  if there exists polynomial-time reduction of L to L'

- We write $L \preceq_{\mathbf{L}} L'$
  if there exists log-space reduction of L to L'

Reductions

Notation

- Let L and L' problems (i.e. languages)
- We write $L \preceq_{\mathbf{P}} L'$
  if there exists polynomial-time reduction of L to L'

- We write $L \preceq_{\mathbf{L}} L'$
  if there exists log-space reduction of L to L'
- Log-space reductions are standard in complexity theory
  (also *many-one* reductions since not necessarily injective)

§5.4 Proposition (Transitivity)

$\preceq_P$ and $\preceq_L$ are transitive

Proof

Statement is obvious for $\preceq_P$.

§5.4 Proposition (Transitivity)

$\preceq_{\mathbf{P}}$ and $\preceq_{\mathbf{L}}$ are transitive

Proof

Statement is obvious for $\preceq_{\mathbf{P}}$. Let $f\colon \Sigma^* \to \Delta^*$ and $g\colon \Delta^* \to \Gamma^*$ log-space reductions. Then there exists $i \in \mathbb{N}_+$ such that f generates output $f(w)$ on input w with $|f(w)| \leq (|w|+2)^i$. The application of g on this output needs space $O(\log((x+2)^i)) = O(\log(x+2))$.

§5.4 Proposition (Transitivity)

$\preceq_{\mathbf{P}}$ and $\preceq_{\mathbf{L}}$ are transitive

Proof

Statement is obvious for $\preceq_{\mathbf{P}}$. Let $f\colon \Sigma^* \to \Delta^*$ and $g\colon \Delta^* \to \Gamma^*$ log-space reductions. Then there exists $i \in \mathbb{N}_+$ such that f generates output $f(w)$ on input w with $|f(w)| \leq (|w|+2)^i$. The application of g on this output needs space $O(\log((x+2)^i)) = O(\log(x+2))$. However, we cannot store output $f(w)$ in logarithmic space on working tape, which yields failure of classical composition construction.

§5.4 Proposition (Transitivity)

$\preceq_{\mathbf{P}}$ and $\preceq_{\mathbf{L}}$ are transitive

Proof

Statement is obvious for $\preceq_{\mathbf{P}}$. Let $f\colon \Sigma^* \to \Delta^*$ and $g\colon \Delta^* \to \Gamma^*$ log-space reductions. Then there exists $i \in \mathbb{N}_+$ such that f generates output $f(w)$ on input w with $|f(w)| \leq (|w|+2)^i$. The application of g on this output needs space $O(\log((x+2)^i)) = O(\log(x+2))$. However, we cannot store output $f(w)$ in logarithmic space on working tape, which yields failure of classical composition construction. Instead we simulate g directly and utilize binary counter ℓ that stores at which position in the output of f the computation of g currently is.

§5.4 Proposition (Transitivity)

$\preceq_{\mathbf{P}}$ and $\preceq_{\mathbf{L}}$ are transitive

Proof

Statement is obvious for $\preceq_{\mathbf{P}}$. Let $f\colon \Sigma^* \to \Delta^*$ and $g\colon \Delta^* \to \Gamma^*$ log-space reductions. Then there exists $i \in \mathbb{N}_+$ such that f generates output $f(w)$ on input w with $|f(w)| \leq (|w|+2)^i$. The application of g on this output needs space $O(\log((x+2)^i)) = O(\log(x+2))$. However, we cannot store output $f(w)$ in logarithmic space on working tape, which yields failure of classical composition construction. Instead we simulate g directly and utilize binary counter ℓ that stores at which position in the output of f the computation of g currently is. For the next step of g we compute the output at position ℓ of f by simulating the computation of f on input w, but only generate the ℓ-th output. □

Hardness and Completeness

§5.5 Definition ($\mathcal{C}$-hard, $\mathcal{C}$-complete)

Let $\mathcal{C}$ be complexity class. Problem L is

- $\mathcal{C}$-hard if $L' \preceq_{\mathsf{L}} L$ for all $L' \in \mathcal{C}$

Hardness and Completeness

§5.5 Definition ($\mathcal{C}$-hard, $\mathcal{C}$-complete)

Let $\mathcal{C}$ be complexity class. Problem L is

- $\mathcal{C}$-hard if $L' \preceq_{\mathsf{L}} L$ for all $L' \in \mathcal{C}$
- $\mathcal{C}$-complete if L $\mathcal{C}$-hard and $L \in \mathcal{C}$

Hardness and Completeness

§5.5 Definition ($\mathcal{C}$-hard, $\mathcal{C}$-complete)

Let $\mathcal{C}$ be complexity class. Problem L is

- $\mathcal{C}$-hard if $L' \preceq_{\mathsf{L}} L$ for all $L' \in \mathcal{C}$
- $\mathcal{C}$-complete if L $\mathcal{C}$-hard and $L \in \mathcal{C}$

Notes

- $\mathcal{C}$-hard problems at least as complex as all problems in $\mathcal{C}$
  (under log-space reductions)

Hardness and Completeness

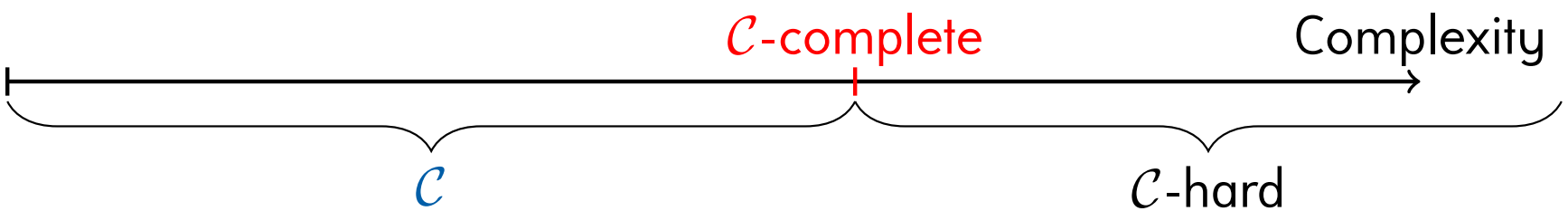
§5.5 Definition ($\mathcal{C}$-hard, $\mathcal{C}$-complete)

Let $\mathcal{C}$ be complexity class. Problem L is

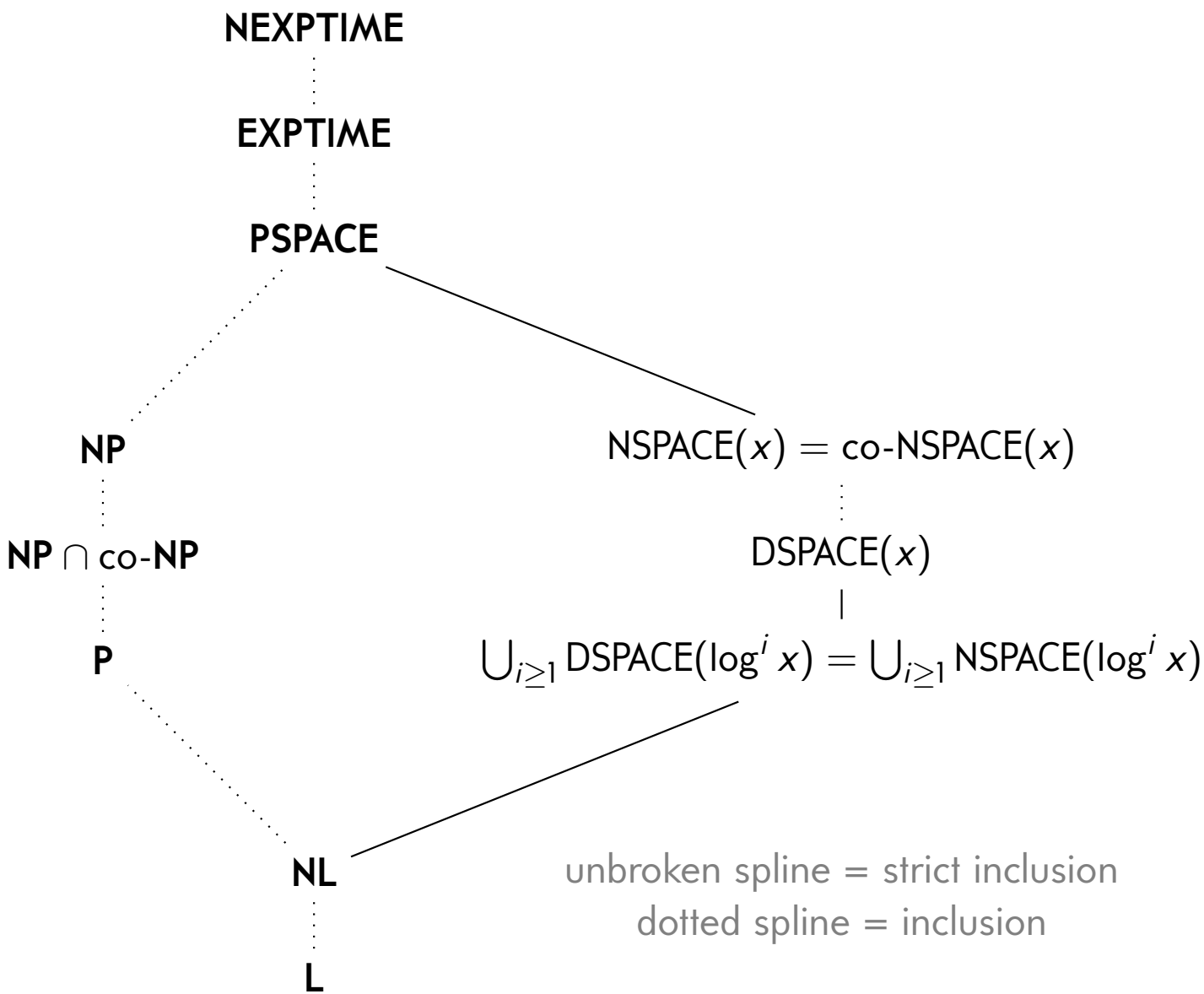
- $\mathcal{C}$-hard if $L' \preceq_{\mathsf{L}} L$ for all $L' \in \mathcal{C}$
- $\mathcal{C}$-complete if L $\mathcal{C}$-hard and $L \in \mathcal{C}$

Notes

- $\mathcal{C}$-hard problems at least as complex as all problems in $\mathcal{C}$
  (under log-space reductions)
- $\mathcal{C}$-complete problems are best representatives of complexity class $\mathcal{C}$



Complexity Classes



Complete Problems for L

$$\text{UGAP} = \{\langle (V, E), s, t\rangle \mid s, t \in V,\ (V, E) \text{ undir. finite graph}$$
$$t \text{ is reachable from } s \text{ in } (V, E)\}$$

§5.6 Theorem (UGAP is L-complete)

Graph reachability UGAP for undirected graphs is **L**-complete

Complete Problems for L

$$\text{UGAP} = \{\langle (V,E), s, t\rangle \mid s,t \in V,\ (V,E) \text{ undir. finite graph}\ t \text{ is reachable from } s \text{ in } (V,E)\}$$

§5.6 Theorem (UGAP is L-complete)

Graph reachability UGAP for undirected graphs is **L**-complete

Omer Reingold (* 1969)

- Isra. computer scientist; professor at Stanford
- Received "Grace Murray Hopper Award"
- Proof in "Undirected connectivity in log-space" (JACM 55(4), 2008)



Complete Problems for L

Notes

- Direct logical characterization

$$L \in \mathbf{L} \iff \exists \varphi \in \mathrm{FO(CTC)} : L = \{w \in \Sigma^* \mid w \models \varphi\} \quad \text{for all } L \subseteq \Sigma^*$$

- All languages that are expressible by first-order (FO) logic with commutative transitive closure (CTC)

  ▸ Literature Wolfgang Thomas. Languages, Automata, and Logic

- (here fixed formula characterizes language;
  i.e. check whether input is model for formula;
  for satisfiability SAT the formula is input)

Complete Problems for L

Notes

- Direct logical characterization

$$L \in \mathbf{L} \iff \exists \varphi \in \mathrm{FO(CTC)} : L = \{w \in \Sigma^* \mid w \models \varphi\} \quad \text{for all } L \subseteq \Sigma^*$$

- All languages that are expressible by first-order (FO) logic with commutative transitive closure (CTC)

  ▸ Literature Wolfgang Thomas. Languages, Automata, and Logic

- (here fixed formula characterizes language; i.e. check whether input is model for formula; for satisfiability SAT the formula is input)

Example

- Weak connectedness of finite graphs

$$\{(V, E) \mid (\exists v \in V)(\forall v' \in V) : (v = v') \vee (E^s)^+(v, v')\}$$

Complete Problems for **NL**

§5.7 Theorem (GAP is NL-complete)

Graph reachability GAP is **NL**-complete

Complete Problems for **NL**

§5.7 Theorem (GAP is NL-complete)

Graph reachability GAP is **NL**-complete

Proof (1/2)

We first show that GAP $\in$ **NL**. Let (V, E) be input graph and $s, t \in V$ start and target vertex.

1. Let $v = s$
2. If $v = t$ accept, otherwise guess $v' \in V$

Complete Problems for **NL**

§5.7 Theorem (GAP is NL-complete)

Graph reachability GAP is **NL**-complete

Proof (1/2)

We first show that GAP $\in$ **NL**. Let (V, E) be input graph and $s, t \in V$ start and target vertex.

1. Let $v = s$
2. If $v = t$ accept, otherwise guess $v' \in V$
3. If $(v, v') \notin E$, then reject, otherwise set $v \leftarrow v'$ and back to 2

Complete Problems for **NL**

§5.7 Theorem (GAP is NL-complete)

Graph reachability GAP is **NL**-complete

Proof (1/2)

We first show that GAP $\in$ **NL**. Let (V, E) be input graph and $s, t \in V$ start and target vertex.

1. Let $v = s$
2. If $v = t$ accept, otherwise guess $v' \in V$
3. If $(v, v') \notin E$, then reject, otherwise set $v \leftarrow v'$ and back to 2

Obviously this nondeterministic algorithm solves GAP and only space for 2 vertices is needed. Thus GAP $\in$ **NL**.

Complete Problems for **NL**

Proof (2/2)

Let $L \in \mathbf{NL}$ and M logarithmic space-bounded 1-TM that accepts L.
We consider reduction f given for all $w \in \Sigma^*$ by $f(w) = \langle (V, E), s, t \rangle$

- $V = \{\kappa \mid \text{configuration of } M \text{ on input } w \text{ with } |\kappa| \leq \log(|w| + 2)\}$
- $E = \{(\kappa, \kappa') \mid \kappa \Rightarrow_M \kappa'\}$
- $s = I_w$ and $t = (q_+, \square, 1, 1)$ wlog. single accepting configuration.

Complete Problems for **NL**

Proof (2/2)

Let $L \in \mathbf{NL}$ and M logarithmic space-bounded 1-TM that accepts L.
We consider reduction f given for all $w \in \Sigma^*$ by $f(w) = \langle (V, E), s, t \rangle$

- $V = \{\kappa \mid \text{configuration of } M \text{ on input } w \text{ with } |\kappa| \leq \log(|w| + 2)\}$
- $E = \{(\kappa, \kappa') \mid \kappa \Rightarrow_M \kappa'\}$
- $s = I_w$ and $t = (q_+, \square, 1, 1)$ wlog. single accepting configuration.

Obviously $w \in L(M)$ iff t is reachable from s in (V, E).

Complete Problems for **NL**

Proof (2/2)

Let $L \in \mathbf{NL}$ and M logarithmic space-bounded 1-TM that accepts L.
We consider reduction f given for all $w \in \Sigma^*$ by $f(w) = \langle (V, E), s, t \rangle$

- $V = \{\kappa \mid \text{configuration of } M \text{ on input } w \text{ with } |\kappa| \leq \log(|w| + 2)\}$
- $E = \{(\kappa, \kappa') \mid \kappa \Rightarrow_M \kappa'\}$
- $s = I_w$ and $t = (q_+, \square, 1, 1)$ wlog. single accepting configuration.

Obviously $w \in L(M)$ iff t is reachable from s in (V, E). Since we only store logarithmic space-bounded configurations, f is log-space reduction. $\square$

Complete Problems for **NL**

Notes

- Logical characterization

$$L \in \mathbf{NL} \iff \exists \varphi \in \mathrm{FO(TC)} : L = \{ w \in \Sigma^* \mid w \models \varphi \} \quad \text{for all } L \subseteq \Sigma^*$$

- All languages that are expressible by first-order (FO) logic with transitive closure (TC)

Complete Problems for **NL**

Notes

- Logical characterization

$$L \in \mathbf{NL} \iff \exists \varphi \in \mathrm{FO(TC)} \colon L = \{w \in \Sigma^* \mid w \models \varphi\} \quad \text{for all } L \subseteq \Sigma^*$$

- All languages that are expressible by first-order (FO) logic with transitive closure (TC)

Example

- Strong connectedness of finite graphs

$$\{(V, E) \mid (\exists v \in V)(\forall v' \in V) \colon (v = v') \vee E^+(v, v')\}$$

Complete Problems for **NL**

Notes

- Logical characterization

$$L \in \mathbf{NL} \iff \exists \varphi \in \mathrm{FO(TC)} \colon L = \{w \in \Sigma^* \mid w \models \varphi\} \quad \text{for all } L \subseteq \Sigma^*$$

- All languages that are expressible by first-order (FO) logic with transitive closure (TC)

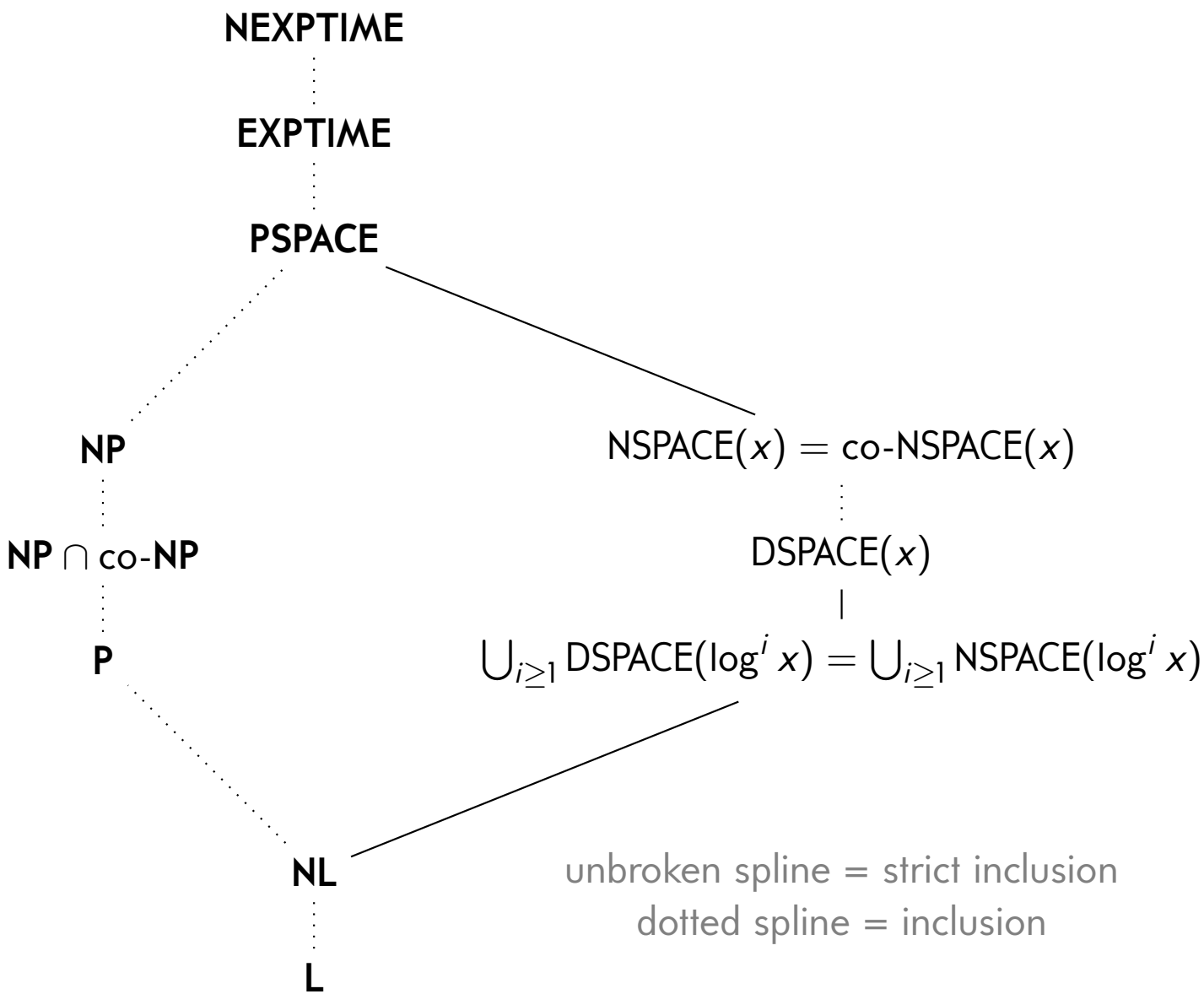
Example

- Strong connectedness of finite graphs

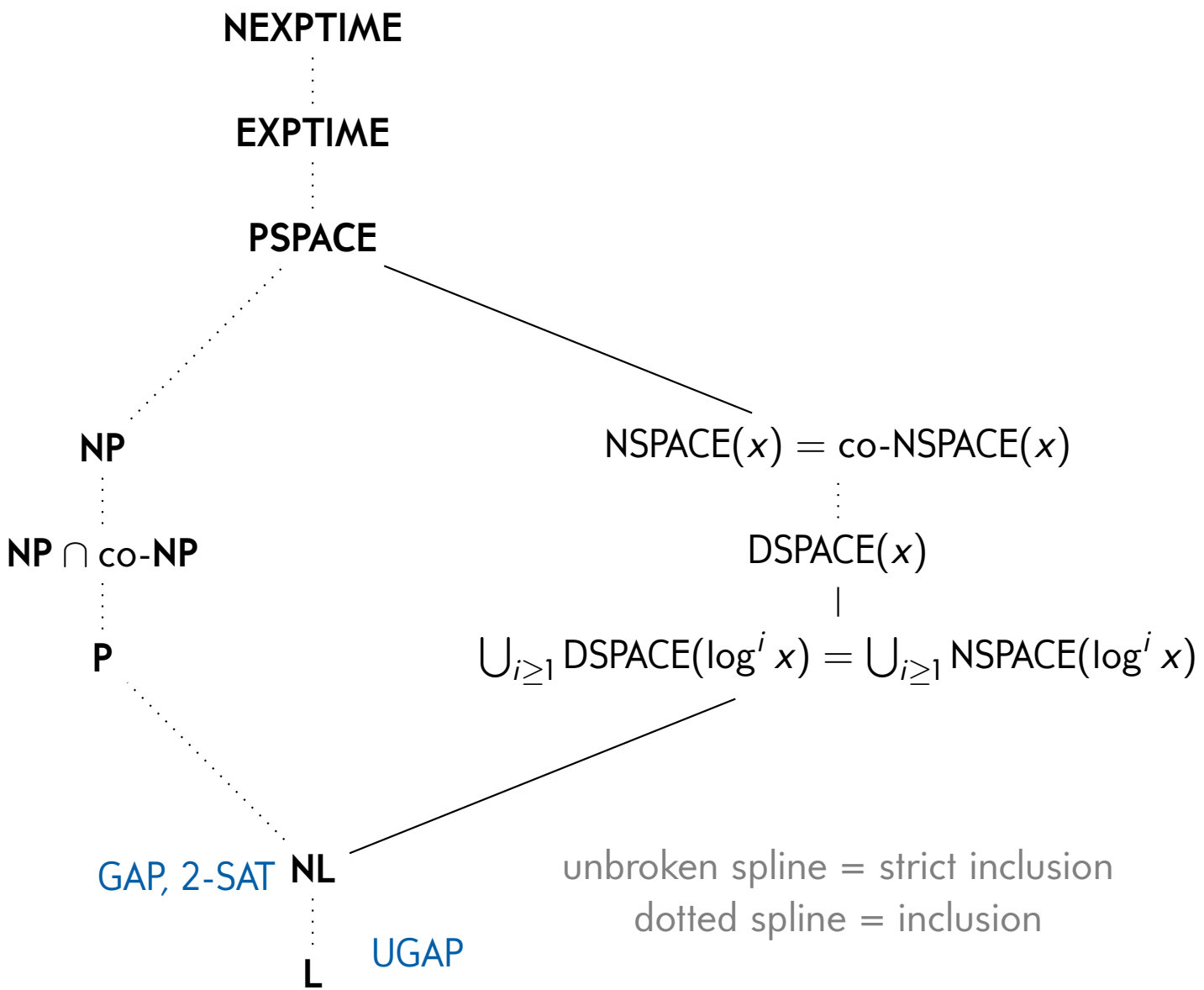
$$\{(V, E) \mid (\exists v \in V)(\forall v' \in V) \colon (v = v') \vee E^+(v, v')\}$$

Additional complete problem: Satisfiability of propositional formulas in conjunctive normal form with 2 literals (2-SAT)

Complextiy Classes



Complextiy Classes



Summary

- Log-space & polynomial-time reductions
- Hardness & completeness for complexity classes
- Complete problems for **L** and **NL** (UGAP and GAP)

Fourth exercise sheet available on my website