Université Paris Cité – LIPADE

Algorithmic Complexity

Space Complexity

Jean-Guy Mailly (jean-guy.mailly@u-paris.fr)

2022

Outline

Space Used by a Turing Machine

- If $\mathcal{M}$ is a Turing Machine and x an input word, the space used by $\mathcal{M}$ on x is the number of different squares visited by $\mathcal{M}$ during the computation.
- For $f : \mathbb{N} \rightarrow \mathbb{N}$, we say that a Turing Machine $\mathcal{M}$ works with space $\mathcal{O}(f(n))$ if
  - $\mathcal{M}$ stops for every input x
  - on every input x, $\mathcal{M}$ uses on x a space $s(n)$ with $s(n) \in \mathcal{O}(f(n))$

Space Complexity Classes

- For $f : \mathbb{N} \rightarrow \mathbb{N}$, DSPACE($f(n)$) is the set of languages which are decided by a deterministic Turing Machine $\mathcal{M}$ working with space $\mathcal{O}(f(n))$.
- For $f : \mathbb{N} \rightarrow \mathbb{N}$, NSPACE($f(n)$) is the set of languages which are decided by a non-deterministic Turing Machine $\mathcal{M}$ working with space $\mathcal{O}(f(n))$.

Deterministic vs Non-Deterministic Space

Intuition: Non-determinism is not "so important" for space complexity...

Theorem [Savitch 1970]

For all function f space-constructible,

$$\mathrm{DSPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n)) \subseteq \mathrm{DSPACE}(f(n)^2)$$

Complement Classes

Proposition

For all function $f(n)$ space-constructible,

$$\mathrm{DSPACE}(f(n)) = \mathrm{coDSPACE}(f(n))$$

Theorem [Immerman 1988, Szelepcsényi 1988]

For all function $f(n) \geq \log(n)$ space-constructible,

$$\mathrm{NSPACE}(f(n)) = \mathrm{coNSPACE}(f(n))$$

Outline

Deterministic Space Complexity

- $\mathsf{L} = \mathsf{DSPACE}(\log(n))$ is the class of decision problems that can be solved by a DTM using logarithmic space
- $\mathsf{PSPACE} = \bigcup_{k>0} \mathsf{DSPACE}(n^k)$ is the class of decision problems that can be solved by a DTM using polynomial space

Deterministic Space Complexity

- $\mathrm{L} = \mathrm{DSPACE}(\log(n))$ is the class of decision problems that can be solved by a DTM using logarithmic space
- $\mathrm{PSPACE} = \bigcup_{k>0} \mathrm{DSPACE}(n^k)$ is the class of decision problems that can be solved by a DTM using polynomial space

$$\mathrm{L} \subseteq \mathrm{PSPACE}$$

Non-Deterministic Space Complexity

- $\mathrm{NL} = \mathrm{NSPACE}(\log(n))$ is the class of decision problems that can be solved by a NDTM using logarithmic space
- $\mathrm{NPSPACE} = \bigcup_{k>0} \mathrm{NSPACE}(n^k)$ is the class of decision problems that can be solved by a NDTM using polynomial space

Non-Deterministic Space Complexity

- $\mathsf{NL} = \mathsf{NSPACE}(\log(n))$ is the class of decision problems that can be solved by a NDTM using logarithmic space
- $\mathsf{NPSPACE} = \bigcup_{k>0} \mathsf{NSPACE}(n^k)$ is the class of decision problems that can be solved by a NDTM using polynomial space

$$\mathsf{NL} \subseteq \mathsf{NPSPACE}$$

PSPACE vs NPSPACE

Proposition

$$\mathrm{PSPACE} = \mathrm{NPSPACE}$$

PSPACE vs NPSPACE

Proposition

$$\text{PSPACE} = \text{NPSPACE}$$

Proof:

- For any $k > 0$,
  $\text{NSPACE}(n^k) \subseteq \text{DSPACE}((n^k)^2) = \text{DSPACE}((n^{2k}) \subseteq \text{PSPACE}$,
  so $\text{NPSPACE} \subseteq \text{PSPACE}$
- For any $k > 0$, $\text{DSPACE}(n^k) \subseteq \text{NSPACE}(n^k) \subseteq \text{NPSPACE}$, so
  $\text{PSPACE} \subseteq \text{NPSPACE}$

Relations Between Time and Space (1/3)

Proposition

For any mapping $f : \mathbb{N} \to \mathbb{N}$, NTIME$(f(n)) \subseteq$ DSPACE$(f(n))$

Relations Between Time and Space (1/3)

Proposition

For any mapping $f : \mathbb{N} \to \mathbb{N}$, NTIME$(f(n)) \subseteq$ DSPACE$(f(n))$

Corollary

$$\text{NP} \subseteq \text{PSPACE}$$

Relations Between Time and Space (2/3)

Polynomial Hierarchy vs Polynomial Space

$$\text{PH} \subseteq \text{PSPACE}$$

Intuition

For each class of the polynomial hierarchy, there is a complete problem $\forall_i$QBF or $\exists_i$QBF. All these problems are special instances of TQBF, so they are in PSPACE.

Relations Between Time and Space (3/3)

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq P \subseteq \begin{matrix}\mathrm{NP} \\ \mathrm{coNP}\end{matrix} \subseteq \Delta_2^{\mathrm{P}} \subseteq \begin{matrix}\Sigma_2^{\mathrm{P}} \\ \Pi_2^{\mathrm{P}}\end{matrix} \subseteq \cdots \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP}$$

We also know that some inclusions are strict:

- NL $\subset$ PSPACE
- P $\subset$ EXP

Outline

Completeness for Space Complexity Classes

The notions of hardness and completeness exist for space complexity classes.

- For most of the complexity classes, the definition is exactly the same as hardness and completeness for time complexity

Completeness for Space Complexity Classes

The notions of hardness and completeness exist for space complexity classes.

- For most of the complexity classes, the definition is exactly the same as hardness and completeness for time complexity
  - A problem $\mathcal{P}$ is C-hard if for every problem $\mathcal{P}$' in C, $\mathcal{P}' \leq_f^{\mathsf{P}} \mathcal{P}$, *i.e.* there is a polynomial reduction from any problem in C to $\mathcal{P}$

Completeness for Space Complexity Classes

The notions of hardness and completeness exist for space complexity classes.

- For most of the complexity classes, the definition is exactly the same as hardness and completeness for time complexity
  - A problem $\mathcal{P}$ is C-hard if for every problem $\mathcal{P}$' in C, $\mathcal{P}' \leq_f^{\mathsf{P}} \mathcal{P}$, *i.e.* there is a polynomial reduction from any problem in C to $\mathcal{P}$
  - A problem is C-complete if it is in C and it is C-hard

Completeness for Space Complexity Classes

The notions of hardness and completeness exist for space complexity classes.

- For most of the complexity classes, the definition is exactly the same as hardness and completeness for time complexity
  - A problem $\mathcal{P}$ is C-hard if for every problem $\mathcal{P}$' in C, $\mathcal{P}' \leq_f^{\mathsf{P}} \mathcal{P}$, *i.e.* there is a polynomial reduction from any problem in C to $\mathcal{P}$
  - A problem is C-complete if it is in C and it is C-hard
  - This definition works for PSPACE-hardness and PSPACE-completeness

Completeness for Space Complexity Classes

The notions of hardness and completeness exist for space complexity classes.

- For most of the complexity classes, the definition is exactly the same as hardness and completeness for time complexity
  - A problem $\mathcal{P}$ is C-hard if for every problem $\mathcal{P}$' in C, $\mathcal{P}' \leq_f^{\mathsf{P}} \mathcal{P}$, *i.e.* there is a polynomial reduction from any problem in C to $\mathcal{P}$
  - A problem is C-complete if it is in C and it is C-hard
  - This definition works for PSPACE-hardness and PSPACE-completeness
  - It does not work for L and NL

Logarithmic Space and Completeness

Logarithmic-Space Functional Reduction

A logarithmic-space functional reduction f is a total computable function from a problem $\mathcal{P}_1$ to a problem $\mathcal{P}_2$ such that, for any instance i of $\mathcal{P}_1$,

- $f(i)$ can be computed in logarithmic-space in the size of i
- i is a positive instance of $\mathcal{P}_1$ iff $f(i)$ is a positive instance of $\mathcal{P}_2$

Notation: $\mathcal{P}_1 \leq_f^{\mathsf{L}} \mathcal{P}_2$

Logarithmic Space and Completeness

Logarithmic-Space Functional Reduction

A logarithmic-space functional reduction f is a total computable function from a problem $\mathcal{P}_1$ to a problem $\mathcal{P}_2$ such that, for any instance i of $\mathcal{P}_1$,

- $f(i)$ can be computed in logarithmic-space in the size of i
- i is a positive instance of $\mathcal{P}_1$ iff $f(i)$ is a positive instance of $\mathcal{P}_2$

Notation: $\mathcal{P}_1 \leq_f^{\mathsf{L}} \mathcal{P}_2$
Observation: If $\mathcal{P}_1 \leq_f^{\mathsf{L}} \mathcal{P}_2$, then $\mathcal{P}_1 \leq_f^{\mathsf{P}} \mathcal{P}_2$

Logarithmic Space and Completeness

Logarithmic-Space Functional Reduction

A logarithmic-space functional reduction f is a total computable function from a problem $\mathcal{P}_1$ to a problem $\mathcal{P}_2$ such that, for any instance i of $\mathcal{P}_1$,

- $f(i)$ can be computed in logarithmic-space in the size of i
- i is a positive instance of $\mathcal{P}_1$ iff $f(i)$ is a positive instance of $\mathcal{P}_2$

Notation: $\mathcal{P}_1 \leq_f^{\mathsf{L}} \mathcal{P}_2$
Observation: If $\mathcal{P}_1 \leq_f^{\mathsf{L}} \mathcal{P}_2$, then $\mathcal{P}_1 \leq_f^{\mathsf{P}} \mathcal{P}_2$

NL-Hardness and NL-Completeness

- A problem $\mathcal{P}$ is NL-hard iff $\forall \mathcal{P}' \in \mathsf{NL}$, $\mathcal{P}' \leq_f^{\mathsf{L}} \mathcal{P}$
- A problem $\mathcal{P}$ is NL-complete iff $\mathcal{P}$ is NL-hard and $\mathcal{P} \in \mathsf{NL}$

What About L-Completeness?

Theorem

Any problem in L is L-complete with respect to $\leq_f^{\mathsf{L}}$!

What About L-Completeness?

Theorem

Any problem in L is L-complete with respect to $\leq_f^{L}$!

To define a meaningful notion of L-completeness, the hardness must be defined with another kind of functional reductions
More details in [Garey and Johnson 1979]

Canonical PSPACE-Complete Problem

Reminder:

- A canonical QBF is a formula $\mathcal{Q}_1\mathcal{X}_1, \mathcal{Q}_2\mathcal{X}_2, \dots \mathcal{Q}_n\mathcal{X}_n, \phi$:
  - $\mathcal{Q}_i \in \{\forall, \exists\}$ and $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$
  - $\mathcal{X}_1, \dots \mathcal{X}_n$ form a partition of the Boolean variables in ϕ
  - ϕ is a propositional formula
- $\exists_n$QBF: is the QBF $\exists\mathcal{X}_1, \forall\mathcal{X}_2, \dots \mathcal{Q}_n\mathcal{X}_n, \phi$ true?
- $\forall_n$QBF: is the QBF $\forall\mathcal{X}_1, \exists\mathcal{X}_2, \dots \mathcal{Q}_n\mathcal{X}_n, \phi$ true?

Canonical PSPACE-Complete Problem

Reminder:

- A canonical QBF is a formula $\mathcal{Q}_1\mathcal{X}_1, \mathcal{Q}_2\mathcal{X}_2, \dots \mathcal{Q}_n\mathcal{X}_n, \phi$:
  - $\mathcal{Q}_i \in \{\forall, \exists\}$ and $\mathcal{Q}_i \neq \mathcal{Q}_{i+1}$
  - $\mathcal{X}_1, \dots \mathcal{X}_n$ form a partition of the Boolean variables in ϕ
  - ϕ is a propositional formula
- $\exists_n$QBF: is the QBF $\exists\mathcal{X}_1, \forall\mathcal{X}_2, \dots \mathcal{Q}_n\mathcal{X}_n, \phi$ true?
- $\forall_n$QBF: is the QBF $\forall\mathcal{X}_1, \exists\mathcal{X}_2, \dots \mathcal{Q}_n\mathcal{X}_n, \phi$ true?

Complexity of TQBF

$$\text{TQBF} = \bigcup_{n \geq 1} (\exists_n\text{QBF} \cup \forall_n\text{QBF})$$

TQBF is PSPACE-complete

PSPACE vs Games
Tic-Tac-Toe

- Classical Tic-Tac-Toe: In a 3×3 grid, two players must draw some marks (X or O). The winner is the first who has placed three marks in a horizontal, vertical or diagonal row
- Generalized Tic-tac-toe ((m, n, k) game): In a $m \times n$ grid, a player wins when he has placed k marks in a row
- A winning strategy at time t for a player is a sequence of moves (drawing a mark in the grid) such that the player wins.

- Classical Tic-Tac-Toe: In a 3×3 grid, two players must draw some marks (X or O). The winner is the first who has placed three marks in a horizontal, vertical or diagonal row
- Generalized Tic-tac-toe ((m, n, k) game): In a $m \times n$ grid, a player wins when he has placed k marks in a row
- A winning strategy at time t for a player is a sequence of moves (drawing a mark in the grid) such that the player wins.

Complexity of (m, n, k)-STRAT

$(m - n - k)$-STRAT is the decision problem: “Is there a winning strategy for player X in the (m, n, k) game?”
$(m - n - k)$-STRAT is PSPACE-complete

PSPACE vs Games

Tic-Tac-Toe: Intuition

- Determining if there is a winning strategy for player X is like asking the question:

  Is there a way to draw an X at step 1, such that for all possible moves of the O player at step 2, there is a way to draw an X at step 3, such that for all possible moves of the O player at step 4, there is ... such that X player wins?

PSPACE vs Games

Tic-Tac-Toe: Intuition

- Determining if there is a winning strategy for player X is like asking the question:

  Is there a way to draw an X at step 1, such that for all possible moves of the O player at step 2, there is a way to draw an X at step 3, such that for all possible moves of the O player at step 4, there is . . . such that X player wins?

- This can be represented as a QBF formula

Generalized Super Mario Bros.

[Demaine *et al.* 2016]

Figure – Super Mario Bros. is PSPACE-complete

Lemmings
[Viglietta 2015]

Figure – Lemmings is PSPACE-complete

NL-Complete Problem in Graphs

Reach

Given a directed graph $G = \langle N, E \rangle$ and two nodes $n_1, n_2 \in N$, is there a path from n_1 to n_2 in G?

Theorem [Papadimitriou 1994]

Reach is NL-complete

Reach is NL-complete: Proof

We will prove that

- Reach $\in$ NL
- Reach is NL-hard

Reach $\in$ NL?

- Let us find a non-deterministic logspace algorithm that solves Reach

Reach $\in$ NL: Proof

Algorithm 1 Reach

Input: $G = \langle N, E \rangle, n_1, n_2 \in N$

$v = n_1$
$cpt = 0$
while $cpt < |N|$ **do**
  Non-deterministically pick v' some neighbour of v
  $v = v'$
  if $v == n_2$ **then**
    return YES
  end if
  $cpt{+}{+}$
end while
return NO

- Space is logarithmic: we only need to store v and cpt

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \mathsf{NL}$, $\mathcal{P} \leq_f^{\mathsf{L}}$ Reach

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \mathsf{NL}$, $\mathcal{P} \leq_f^{\mathsf{L}}$ Reach
- Given $\mathcal{P} \in \mathsf{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \mathsf{NL}$, $\mathcal{P} \leq^{\mathsf{L}}_{f}$ Reach
- Given $\mathcal{P} \in \mathsf{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \rightarrow$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \text{NL}$, $\mathcal{P} \leq^{\text{L}}_{f}$ Reach
- Given $\mathcal{P} \in \text{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \rightarrow$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach
- $f(x) = (G_x = \langle N, E \rangle, n_1, n_2)$ is defined by

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in$ NL, $\mathcal{P} \leq^{\mathsf{L}}_{f}$ Reach
- Given $\mathcal{P} \in$ NL, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \rightarrow$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach
- $f(x) = (G_x = \langle N, E \rangle, n_1, n_2)$ is defined by
  - N is the set of configurations of $\mathcal{M}$

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \mathsf{NL}$, $\mathcal{P} \leq_f^{\mathsf{L}}$ Reach
- Given $\mathcal{P} \in \mathsf{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \rightarrow$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach
- $f(x) = (G_x = \langle N, E \rangle, n_1, n_2)$ is defined by
  - N is the set of configurations of $\mathcal{M}$
  - $(a, b) \in E$ iff there is a step that goes from a to b when $\mathcal{M}$ is executed on x

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \text{NL}$, $\mathcal{P} \leq_f^{\mathsf{L}}$ Reach
- Given $\mathcal{P} \in \text{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \to$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach
- $f(x) = (G_x = \langle N, E \rangle, n_1, n_2)$ is defined by
  - N is the set of configurations of $\mathcal{M}$
  - $(a, b) \in E$ iff there is a step that goes from a to b when $\mathcal{M}$ is executed on x
  - n_1 is the initial configuration and n_2 is the accept configuration

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \mathsf{NL}$, $\mathcal{P} \leq_f^{\mathsf{L}}$ Reach
- Given $\mathcal{P} \in \mathsf{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \to$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach
- $f(x) = (G_x = \langle N, E \rangle, n_1, n_2)$ is defined by
  - N is the set of configurations of $\mathcal{M}$
  - $(a, b) \in E$ iff there is a step that goes from a to b when $\mathcal{M}$ is executed on x
  - n_1 is the initial configuration and n_2 is the accept configuration
  - $\mathcal{M}$ accepts x iff there is a path from n_1 to n_2

Reach is NL-hard: Sketch of proof

- We need to prove that $\forall \mathcal{P} \in \text{NL}$, $\mathcal{P} \leq_f^{\text{L}}$ Reach
- Given $\mathcal{P} \in \text{NL}$, let $\mathcal{M}$ be a NDTM that decides $\mathcal{P}$
- We will define $f : \mathcal{P} \to$ Reach such that $x \in \mathcal{P}$ iff $f(x) \in$ Reach
- $f(x) = (G_x = \langle N, E \rangle, n_1, n_2)$ is defined by
  - N is the set of configurations of $\mathcal{M}$
  - $(a, b) \in E$ iff there is a step that goes from a to b when $\mathcal{M}$ is executed on x
  - n_1 is the initial configuration and n_2 is the accept configuration
  - $\mathcal{M}$ accepts x iff there is a path from n_1 to n_2
  - See [Papadimitriou 1994] for more details

NL-Complete Problem in Logics

Reminder on 2SAT

2SAT is a special case of satisfiability problem with CNF formulas with only clauses of length ≤ 2

Theorem [Papadimitriou 1994]

2SAT is NL-complete

2SAT is NL-complete: Sketch of proof

We will prove that

- 2SAT $\in$ NL
- 2SAT is NL-hard

2SAT is NL-complete: Sketch of proof

We will prove that

- 2SAT $\in$ NL
- 2SAT is NL-hard
- Reminder: From previous theorem [Immerman 1988, Szelepcsényi 1988], coNL = NL

2SAT is NL-complete: Sketch of proof

We will prove that

- 2SAT $\in$ NL
- 2SAT is NL-hard
- Reminder: From previous theorem [Immerman 1988, Szelepcsényi 1988], coNL = NL
- We define Unreach $= \{(G = \langle N, E \rangle, n_1, n_2) \mid G$ is a directed graph, $n_1, n_2 \in N$, and there is no path from n_1 to n_2 in $G\}$
- By definition, Unreach $\in$ coNL = NL

2SAT is NL-complete: Sketch of proof

We will prove that

- 2SAT $\in$ NL
- 2SAT is NL-hard
- Reminder: From previous theorem [Immerman 1988, Szelepcsényi 1988], coNL = NL
- We define Unreach $= \{(G = \langle N, E \rangle, n_1, n_2) \mid G$ is a directed graph, $n_1, n_2 \in N$, and there is no path from n_1 to n_2 in $G\}$
- By definition, Unreach $\in$ coNL = NL
- We will use Reach and Unreach to prove that 2SAT is NL-complete

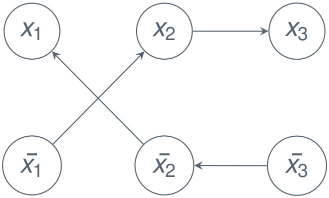
Implication graph of a 2CNF formula

- Given ϕ a 2CNF formula on variables X, we build $G_\phi = \langle N, E \rangle$ a directed graph such that
  - $N = X \cup \{\bar{x} \mid x \in X\}$
  - $(l_1, l_2) \in E$ iff $\neg l_1 \vee l_2$ is a clause in ϕ (*i.e.* $l_1 \Rightarrow l_2$)

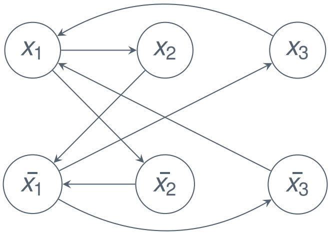
Implication graph of a 2CNF formula

- Given ϕ a 2CNF formula on variables X, we build $G_\phi = \langle N, E \rangle$ a directed graph such that
  - $N = X \cup \{\bar{x} \mid x \in X\}$
  - $(l_1, l_2) \in E$ iff $\neg l_1 \vee l_2$ is a clause in ϕ (*i.e.* $l_1 \Rightarrow l_2$)

$\phi_1 = (x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$



$$\phi_2 = (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_1) \wedge (x_1 \vee x_3) \wedge (\neg x_3 \vee x_1)$$



2SAT and the implication graph

- Lemma: The formula ϕ is unsatisfiable iff $\exists x$ such that there is a path from x to $\bar{x}$ and a path from $\bar{x}$ to x in G_ϕ
- The following non-deterministic algorithm solves 2UNSAT

Algorithm 2 2UNSAT

Input: ϕ
  Build G_ϕ
  Non-deterministically pick x
  return $(G_\phi, x, \bar{x}) \in$ Reach AND $(G_\phi, \bar{x}, x) \in$ Reach

---

- So 2UNSAT$\in$ NL, and 2SAT$\in$ coNL $=$ NL

2SAT is NL-hard: Sketch of proof (1/2)

- We reduce Unreach to 2SAT
- Let $G = \langle N, E \rangle$ be a directed graph, and n_1, n_2 in N. We define $f(G, n_1, n_2) = \phi$ with
  - if $(x, y) \in E$, then $\neg x \vee y$ is a clause in ϕ
  - n_1 is a unit clause in ϕ
  - $\neg n_2$ is a unit clause in ϕ

2SAT is NL-hard: Sketch of proof (1/2)

- We reduce Unreach to 2SAT
- Let $G = \langle N, E \rangle$ be a directed graph, and n_1, n_2 in N. We define $f(G, n_1, n_2) = \phi$ with
  - if $(x, y) \in E$, then $\neg x \vee y$ is a clause in ϕ
  - n_1 is a unit clause in ϕ
  - $\neg n_2$ is a unit clause in ϕ
- if $(G, n_1, n_2) \in$ Unreach, we can make a partition of N: N_1, N_2 s. t. $n_1 \in N_1$ and $n_2 \in N_2$, and there is no edge from N_1 to N_2

2SAT is NL-hard: Sketch of proof (1/2)

- We reduce Unreach to 2SAT
- Let $G = \langle N, E \rangle$ be a directed graph, and n_1, n_2 in N. We define $f(G, n_1, n_2) = \phi$ with
  - if $(x, y) \in E$, then $\neg x \vee y$ is a clause in ϕ
  - n_1 is a unit clause in ϕ
  - $\neg n_2$ is a unit clause in ϕ
- if $(G, n_1, n_2) \in$ Unreach, we can make a partition of N: N_1, N_2 s. t. $n_1 \in N_1$ and $n_2 \in N_2$, and there is no edge from N_1 to N_2
- $\forall x \in N_1$, choose $\omega(x) = 1$, $\forall x \in N_2$, choose $\omega(x) = 0$. $\omega \models \phi$

2SAT is NL-hard: Sketch of proof (1/2)

- We reduce Unreach to 2SAT
- Let $G = \langle N, E \rangle$ be a directed graph, and n_1, n_2 in N. We define $f(G, n_1, n_2) = \phi$ with
  - if $(x, y) \in E$, then $\neg x \vee y$ is a clause in ϕ
  - n_1 is a unit clause in ϕ
  - $\neg n_2$ is a unit clause in ϕ
- if $(G, n_1, n_2) \in$ Unreach, we can make a partition of N: N_1, N_2 s. t. $n_1 \in N_1$ and $n_2 \in N_2$, and there is no edge from N_1 to N_2
- $\forall x \in N_1$, choose $\omega(x) = 1$, $\forall x \in N_2$, choose $\omega(x) = 0$. $\omega \models \phi$
- $(G, n_1, n_2) \in$ Unreach $\Rightarrow \phi \in$ 2SAT

2SAT is NL-hard: Sketch of proof (2/2)

- if $(G, n_1, n_2) \notin$ Unreach, then there is a path $(n_1, x_1, \ldots, x_k, n_2)$ in G. The translation of edges in clauses means that $n_1 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_k \Rightarrow n_2$, so if n_1 is true, then n_2. But ϕ contains both unitary clauses n_1 and $\neg n_2$
- So ϕ is unsatisfiable

2SAT is NL-hard: Sketch of proof (2/2)

- if $(G, n_1, n_2) \notin$ Unreach, then there is a path $(n_1, x_1, \ldots, x_k, n_2)$ in G. The translation of edges in clauses means that $n_1 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_k \Rightarrow n_2$, so if n_1 is true, then n_2. But ϕ contains both unitary clauses n_1 and $\neg n_2$
- So ϕ is unsatisfiable
- $(G, n_1, n_2) \notin$ Unreach $\Rightarrow \phi \notin$ 2SAT

2SAT is NL-hard: Sketch of proof (2/2)

- if $(G, n_1, n_2) \notin$ Unreach, then there is a path $(n_1, x_1, \ldots, x_k, n_2)$ in G. The translation of edges in clauses means that $n_1 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_k \Rightarrow n_2$, so if n_1 is true, then n_2. But ϕ contains both unitary clauses n_1 and $\neg n_2$
- So ϕ is unsatisfiable
- $(G, n_1, n_2) \notin$ Unreach $\Rightarrow \phi \notin$ 2SAT
- We can conclude Unreach $\leq_f^{\mathsf{L}}$ 2SAT

References

[Savitch 1970] W. J. Savitch, *Relationships between nondeterministic and deterministic tape complexities*. Journal of Computer and System Sciences, 177–192, 1970.

[Immerman 1988] N. Immerman, *Nondeterministic space is closed under complementation*. SIAM Journal on Computing 17, 935–938, 1988.

[Szelepcsényi 1988] R. Szelepcsényi, *The Method of Forced Enumeration for Nondeterministic Automata*. Acta Informatica vol. 26, no 3, 279–284, 1988.

References

[Garey and Johnson 1979] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. Section 7.5: Logarithmic Space, 177–181, 1979.

[Papadimitriou 1994] C.H Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.

[Demaine *et al.* 2016] E.D. Demaine, G. Viglietta and A. Williams, *Super Mario Bros. Is Harder/Easier than We Thought*. Proceedings of the 8th International Conference on Fun with Algorithms, 2016.

[Viglietta 2015] G. Viglietta, *Lemmings Is PSPACE-Complete*. Theor. Comput. Sci. 586: 120-134, 2015.