Computability and Complexity

Lecture 14

PSPACE-completeness
Summary of the Course

given by Jiri Srba

PSPACE-Completeness

Definition (PSPACE-Completeness)

A language B is PSPACE-complete iff

1. $B \in \text{PSPACE}$ (containment in PSPACE), and
2. for every $A \in \text{PSPACE}$ we have $A \leq_P B$ (PSPACE-hardness).

PSPACE-Completeness

Definition (PSPACE-Completeness)

A language B is PSPACE-complete iff

1. $B \in \text{PSPACE}$ (containment in PSPACE), and
2. for every $A \in \text{PSPACE}$ we have $A \leq_P B$ (PSPACE-hardness).

Theorem

If B is PSPACE-complete, $B \leq_P C$, and $C \in \text{PSPACE}$, then C is PSPACE-complete.

Proof: Because $\leq_P$ is transitive.

A_{LBA} Is PSPACE-Complete

$$A_{LBA} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid \ M \text{ is an LBA such that } M \text{ accepts } w\ \}$$

Theorem

A_{LBA} is PSPACE-complete.

A_{LBA} Is PSPACE-Complete

$A_{LBA} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is an LBA such that } M \text{ accepts } w\}$

Theorem

A_{LBA} is PSPACE-complete.

Proof:

- Containment in PSPACE: "On input $\langle M, w\rangle$: simulate M on w". This takes only linear space, so it belongs to PSPACE.

A_{LBA} Is PSPACE-Complete

$A_{LBA} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is an LBA such that } M \text{ accepts } w\}$

Theorem

A_{LBA} is PSPACE-complete.

Proof:

- Containment in PSPACE: "On input $\langle M, w\rangle$: simulate M on w". This takes only linear space, so it belongs to PSPACE.
- PSPACE-hardness: Let $L \in$ PSPACE. We show $L \leq_P A_{LBA}$.
  - Because $L \in$ PSPACE then there is a decider M running in space n^k such that $L(M) = L$.
  - Poly-time reduction: On input w: output $\langle M, w(\sqcup)^{|w|^k}\rangle$.
  - Clearly, $w \in L$ iff M accepts $w(\sqcup)^{|w|^k}$.
  - M runs in space n^k, so M on input $w(\sqcup)^{|w|^k}$ acts as LBA. $\square$

A_{LBA} Is PSPACE-Complete

$A_{LBA} \stackrel{\text{def}}{=} \{\langle M, w\rangle \mid M \text{ is an LBA such that } M \text{ accepts } w\}$

Theorem

A_{LBA} is PSPACE-complete.

Proof:

- Containment in PSPACE: "On input $\langle M, w\rangle$: simulate M on w". This takes only linear space, so it belongs to PSPACE.
- PSPACE-hardness: Let $L \in$ PSPACE. We show $L \leq_P A_{LBA}$.
  - Because $L \in$ PSPACE then there is a decider M running in space n^k such that $L(M) = L$.
  - Poly-time reduction: On input w: output $\langle M, w(\sqcup)^{|w|^k}\rangle$.
  - Clearly, $w \in L$ iff M accepts $w(\sqcup)^{|w|^k}$.
  - M runs in space n^k, so M on input $w(\sqcup)^{|w|^k}$ acts as LBA. $\square$

The technique used in this reduction is called padding.

TQBF Is PSPACE-Complete

Quantified Boolean formula (QBF):

$$\psi \stackrel{\text{def}}{=} \forall x_1 \exists x_2 \forall x_3 \exists x_4 \ldots \forall x_{k-1} \exists x_k . \phi$$

where ϕ is a Boolean formula over the variables $x_1, \ldots, x_k$.

Any given QBF ψ is either true or false.

$$TQBF \stackrel{\text{def}}{=} \{ \langle \psi \rangle \mid \ \psi \text{ is a true QBF } \}$$

Theorem

TQBF is PSPACE-complete.

Proof: See the book (not part of the syllabus).

ALL_{NFA} Is PSPACE-Complete

Problem: "Does a given NFA accept all strings from Σ^*?"

Theorem

ALL_{NFA} is PSPACE-complete.

Proof: Not part of the syllabus.

Summary of the Course

Computability theory (decidability of problems).

Complexity theory (time and space requirements needed to solve problems).

Model of computation = Turing machine

Decision Problems

Decision Problem

"Given an instance of the problem, is it a positive instance or a negative instance?"

Language Formulation

$$L = \{\langle P \rangle \mid \ P \text{ is a positive instance of the problem } \}$$

Decision Problems

Decision Problem

"Given an instance of the problem, is it a positive instance or a negative instance?"

Language Formulation

$$L = \{\langle P \rangle \mid \; P \text{ is a positive instance of the problem } \}$$

Questions:

- Computability Theory: Is a given problem algorithmically solvable? (Is the corresponding language decidable?)
- Complexity Theory: How difficult is to solve a given problem? (What is the time/space complexity of the corresponding language?)

Turing Machine — A Model of a Computer

Church-Turing Thesis

"The Turing machine model captures exactly the informal notion of algorithm."

Polynomial Time Equivalence of Deterministic Models (Thesis)

"All reasonable deterministic models of computation are polynomial time equivalent to deterministic single-tape Turing machine."

Variants of TMs and Time vs. Space Complexity

Let $t(n)$ be a function s.t. $t(n) \geq n$.

Theorem (Multi-Tape TM)

Every k-tape TM M has an equivalent 1-tape TM M'.

If M is running in time $t(n)$ then M' is running in time $O(t^2(n))$.

Variants of TMs and Time vs. Space Complexity

Let $t(n)$ be a function s.t. $t(n) \geq n$.

Theorem (Multi-Tape TM)

Every k-tape TM M has an equivalent 1-tape TM M'.

If M is running in time $t(n)$ then M' is running in time $O(t^2(n))$.

Theorem (Nondeterministic TM)

Every nondeterministic TM M has an equivalent determin. TM M'.

If M is running in time $t(n)$ then M' is running in time $2^{O(t(n))}$.

If M is running in space $t(n)$ then M' is running in space $O(t^2(n))$.

Variants of TMs and Time vs. Space Complexity

Let $t(n)$ be a function s.t. $t(n) \geq n$.

Theorem (Multi-Tape TM)

Every k-tape TM M has an equivalent 1-tape TM M'.

If M is running in time $t(n)$ then M' is running in time $O(t^2(n))$.

Theorem (Nondeterministic TM)

Every nondeterministic TM M has an equivalent determin. TM M'.

If M is running in time $t(n)$ then M' is running in time $2^{O(t(n))}$.

If M is running in space $t(n)$ then M' is running in space $O(t^2(n))$.

Theorem (Time vs. Space Complexity)

Every TM running in time $t(n)$ is running in $O(t(n))$ space.

Every TM running in space $t(n)$ is running in time $2^{O(t(n))}$.

Classes of Languages

- P: class of all languages decidable in polynomial time on deterministic TMs.
- NP: class of all languages decidable in polynomial time on nondeterministic TMs.
- co-NP: class of all languages which complements belong to NP.
- PSPACE: class of all languages decidable in polynomial space on (deterministic or nondeterministic) TM.
- EXPTIME: class of all languages decidable in exponential time on deterministic TMs.
- Decidable: class of all languages that are recognized by TMs which are deciders.

Classes of Languages

- P: class of all languages decidable in polynomial time on deterministic TMs.
- NP: class of all languages decidable in polynomial time on nondeterministic TMs.
- co-NP: class of all languages which complements belong to NP.
- PSPACE: class of all languages decidable in polynomial space on (deterministic or nondeterministic) TM.
- EXPTIME: class of all languages decidable in exponential time on deterministic TMs.
- Decidable: class of all languages that are recognized by TMs which are deciders.

- Recognizable: class of all languages that are recognized by TMs.
- Co-recognizable: class of all languages which complements are recognized by TMs.

Crucial Results

Theorem (Turing — Undecidability of A_{TM})

The acceptance problem of a Turing machine is undecidable.

Theorem (Cook-Levin — NP-Completeness of SAT)

The satisfiability problem for Boolean formulae is NP-complete.

Crucial Results

Theorem (Turing — Undecidability of A_{TM})

The acceptance problem of a Turing machine is undecidable.

Theorem (Cook-Levin — NP-Completeness of SAT)

The satisfiability problem for Boolean formulae is NP-complete.

Other undecidable or computationally hard problems were derived using reductions:

- for undecidability we used mapping reductions, and
- for NP-hardness we used polynomial time reductions.

Languages Studied in Computability Theory

- Decidable:
  A_{DFA}, A_{NFA}, E_{NFA}, EQ_{NFA}, A_{CFG}, E_{CFG}, A_{LBA}.
- Recognizable but not decidable:
  A_{TM}, $HALT_{TM}$, $\overline{E_{TM}}$, $\overline{E_{LBA}}$, $\overline{ALL_{CFG}}$, $\overline{EQ_{CFG}}$, PCP, $BPCP$.
- Co-recognizable but not decidable:
  Complements of all languages from the above category.
- Neither recognizable nor co-recognizable:
  EQ_{TM}, $REGULAR_{TM}$, $TOTAL_{TM}$.

Languages Studied in Complexity Theory

- In P:
  PATH, *RELPRIME*, any context-free language.
- NP-complete:
  SAT, *CNF-SAT*, *3SAT*, *HAMPATH*, *UHAMPATH*, *CLIQUE*, *SUBSET-SUM*, *VERTEX-COVER*.
- PSPACE-complete:
  A_{LBA}, $TQBF$, ALL_{NFA}.

Closure Properties

Class of Languages	$\cap$	$\cup$	$\circ$	$*$	$^-$
decidable	YES	YES	YES	YES	YES
recognizable	YES	YES	YES	YES	NO
P	YES	YES	YES	YES	YES
NP	YES	YES	YES	YES	???
PSPACE	YES	YES	YES	YES	YES
EXPTIME	YES	YES	YES	YES	YES

- Intersection: Run two Turing machines in sequence.
- Union: Run two Turing machines in sequence, in parallel, or nondeterministically choose one.
- Concatenation: Try all possible splitting points, or guess the point nondeterministically.
- Kleene star: Try all possible splittings (exponentially many, or use dynamic programming), or guess them.
- Complement: Swap accept and reject state (works only for deterministic TMs that never loop).