Complexity Theory

Georg Moser

Institute of Computer Science @ UIBK

Summer 2008

Outline

- Summary of Last Lecture: The Circuit Value Problem
- Exercises
- Monotone Inductive Definitions
- Alternating Turing Machines
- Alternating Complexity Classes

The Circuit Value Problem

Definition (Boolean circuit)

a Boolean circuit is a program that consists of finitely many assignments of form:

$$P_i := 1 \qquad P_i := P_j \land P_k \qquad P_i := \neg P_j$$
$$P_i := 0 \qquad P_i := P_j \lor P_k$$

where $j, k < i$ and every P_i is defined at most once

the value of circuit is the value of P_n, where n is maximal

Definition (CVP)

the circuit value problem (CVP) is defined as:

- given: Boolean circuit (with several inputs)
- question: what is the value of the circuit?

The Cook-Levin Theorem

Theorem

The circuit value problem is $\leqslant_m^{\log}$-complete for P

Theorem

Boolean satisfiability is $\leqslant_m^{\log}$-complete for NP

Observations

- with SAT the input values are not given, but have to be found
- CVP is defined in terms of circuits
- SAT is defined in terms of formulas

Homework

1. Miscellaneous Exercise 11
2. Miscellaneous Exercise 12
3. Miscellaneous Exercise 14
4. Miscellaneous Exercise 15
5. Miscellaneous Exercise 16

Definition — complete lattices

a complete lattice is

1. a set U and
2. a partial order $\leqslant$ (defined on U)

such that any subset A of U has a least upper bound (denoted as $\sup A$)

Definition — operator

- an operator on a complete lattice U is a function $\tau \colon U \to U$
- an operator is monotone if
$$x \leqslant y \implies \tau(x) \leqslant \tau(y)$$
- an operator is chain-continuous if $\forall A$
$$\tau(\sup A) = \sup_{x \in A} \tau(x)$$
where A is a chain in U, i.e., a totally ordered subset of U
- if $U = 2^X$ ordered by $\subseteq$; operator τ is also called set operator

Definition — fixpoint

- a prefixpoint of an operator τ on a complete lattice U is $x \in U$ such that $\tau(x) \leqslant x$
- a fixpoint of τ on U is $x \in U$ such that $\tau(x) = x$
- for set operators $\tau \colon 2^X \to 2^X$ a subset $A \subseteq X$ is called closed if A is a prefixpoint

Definition — $\tau^{\dagger}(\cdot)$

let
$$PF_{\tau}(x) = \{y \in U \mid \tau(y) \leqslant y \wedge x \leqslant y\}$$
denote the set of all prefixpoints of τ above x; define $\tau^{\dagger}(x) = \inf PF_{\tau}(x)$

Lemma

any monotone operator τ has a $\leqslant$-least fixpoint, namely $\tau^{\dagger}(\bot)$

Definition — closure operator

an operator τ is a closure operator if

1. τ is monotone
2. $\forall\, x\ x \leqslant \tau(x)$
3. $\forall\, x\ \tau(\tau(x)) = \tau(x)$

Lemma

for any monotone operator τ, the operator $\tau^{\dagger}$ is a closure operator

Definition — $\tau^{i}(\cdot)$

let τ be a monotone operator on U
$$\tau^0(x) = \bot \qquad \tau^{\omega}(x) = \sup_{i<\lambda} \tau^i(x)$$
$$\tau^{i+1}(x) = \sup\{x, \tau(\tau^{\alpha}(x))\}$$

Theorem — Knaster-Tarski

for a monotone and chain-continuous operator we have
$$\tau^{\dagger}(x) = \tau^*(x) := \sup_{\alpha \leqslant \omega} \tau^{\alpha}(x)$$

Definition — ATM

- an alternating Turing machine is defined like a nondeterministic Turing machine, but includes a function

$$\text{type}\colon Q \to \{\wedge, \vee, \neg\}$$

- a configuration is called $\wedge$-, $\vee$-, or $\neg$-configuration, depending on the type of its state
- all $\neg$-configurations have exactly one successor
- accept and reject states are formalised implicitly

Definition — three valued logic

we write $\perp$ for the truth value "don't know"

$\vee$	1	$\perp$	0
1	1	1	1
$\perp$	1	$\perp$	$\perp$
0	1	$\perp$	0

$\wedge$	1	$\perp$	0
1	1	$\perp$	0
$\perp$	$\perp$	$\perp$	0
0	0	0	0

$\neg$	
1	0
$\perp$	$\perp$
0	1

$\vee$ is supremum and $\wedge$ is infimum in order $0 \leqslant \perp \leqslant 1$

Definition — information order

the information order is defined as $\perp \sqsubseteq 0$, $\perp \sqsubseteq 1$

Definition — labeling ℓ

let $\mathcal{C}$ a set of configuration, a labeling is a map $\ell\colon \mathcal{C} \to \{0, 1, \perp\}$; we define

$$\ell \sqsubseteq \ell' :\Longleftrightarrow \forall \alpha \in \mathcal{C}\ \ell(\alpha) \sqsubseteq \ell'(\alpha)$$

Lemma

the set of labelings together with $\sqsubseteq$ form a complete lattice, i.e., every set of labelings has a supremum

Definition — τ

we define an operator on labels

$$\tau(\ell)(\alpha) := \begin{cases} \bigwedge_{\alpha \to \beta} \ell(\beta) & \alpha \text{ an } \wedge\text{-configuration} \\ \bigvee_{\alpha \to \beta} \ell(\beta) & \alpha \text{ an } \vee\text{-configuration} \\ \neg \ell(\beta) & \alpha \text{ a } \neg\text{-configuration and } \alpha \to \beta \end{cases}$$

define ℓ_* as the $\sqsubseteq$-least fixpoint of τ

Observation

the labeling ℓ_* is the supremum of the chain

$$\ell_0 \sqsubseteq \ell_1 \sqsubseteq \ell_2 \sqsubseteq \dots$$

where $\ell_0 := \lambda\alpha.\perp$ and $\ell_{i+1} := \tau(\ell_i)$

Definition

an ATM accepts its input x if

- $\ell_*(\texttt{start}) = 1$

it rejects if $\ell_*(\texttt{start}) = 0$

Lemma

every ATM with negations can be simulated by an ATM without negations at no extra cost (in space or time)

Definition — dual ATM

the dual of an ATM M is the alternating TM M', defined as M but with exchanged $\wedge$- and $\vee$-states

Proof

- let M be an ATM and M' its dual
- $\forall\ \alpha,\ \alpha'\ \forall\ i\ \ \ell_i(\alpha) = \neg\ell_i'(\alpha')$, hence $\ell_*(\alpha) = \neg\ell_*'(\alpha')$
- form M'' as the (disjoint) union of M and M'
- $\forall\ p\ \neg$-state
  $\forall\ ((p, a), (q, b, d))$ transition of M
  $\forall\ ((p', a), (q', b, d))$ transition of M'

  make p and $\wedge$-state and p' an $\vee$-state

$$((p, a), (q, b, d)) \mapsto ((p, a), (q', b, d))$$
$$((p', a), (q', b, d)) \mapsto ((p', a), (q, b, d))$$

∎

Alternating Complexity Classes

Definition

$$\text{ALOGSPACE} := \text{ASPACE}(\log n) \qquad \text{APTIME} := \text{ATIME}(n^{O(1)})$$

$$\text{APSPACE} := \text{ATIME}(n^{O(1)}) \qquad \text{AEXPTIME} := \text{ATIME}(2^{n^{O(1)}})$$

Theorem

let $T(n) \geqslant n$ and $S(n) \geqslant \log n$

1. $\text{ATIME}(T(n)) \subseteq \text{DSPACE}(T(n))$
2. $\text{DSPACE}(S(n)) \subseteq \text{ATIME}(S(n)^2)$
3. $\text{ASPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$
4. $\text{DTIME}(T(n)) \subseteq \text{ASPACE}(\log T(n))$

Corollary

for $T(n) \geqslant n$ and $S(n) \geqslant \log n$: $\text{ATIME}(T(n)^{O(1)}) = \text{DSPACE}(T(n)^{O(1)})$ and $\text{ASPACE}(S(n)) = \text{DTIME}(2^{O(S(n))})$