Lecture 7: Complexity and Approximation

Outline

Additional references:

C. Papadimitriou: "*Computational Complexity*", 1994.

T. Jech: "*Set Theory*", 1978.

1. COMPLEXITY CONCEPTS IN BRIEF

➤ We shall use *Turing machines* (TM) as models of computation.

➤ A *deterministic* Turing machine (DTM) M is a quadruple $\langle K, \Sigma, \delta, s\rangle$ where

1. K is a set of *states* that includes the *initial* state $s \in K$,
2. Σ is the finite *alphabet* of M which always contains $\sqcup$ and $\triangleright$, the blank and first symbol, respectively, and
3. δ is a *transition function*
$$\delta : K \times \Sigma \to (K \cup \{\mathrm{halt}, \mathrm{yes}, \mathrm{no}\}) \times \Sigma \times \{\rightarrow, \leftarrow, \downarrow\}$$
where halt, yes, and no are halting, accepting, and rejecting states, respectively, and $\rightarrow$, $\leftarrow$, and $\downarrow$ express cursor moves.

➤ In a *nondeterministic* Turing machine (NTM) M, δ is replaced by a *transition relation* for the domain and range in question.

Deterministic Computation

Consider a deterministic Turing machine $M = \langle K, \Sigma, \delta, s\rangle$.

➤ States of computation are described in terms of *configurations* $\langle q, w, u\rangle$ where $q \in K$ is a state and $w, u \in \Sigma^*$ are strings.

➤ The *initial configuration* of M is $\langle s, \triangleright, x\rangle$ where the string $x \in (\Sigma - \{\sqcup\})^*$ or $x = \sqcup$ is the *input* of M.

➤ The *computation* of M on input x is a sequence of configurations
$$\langle q_0, w_0, u_0\rangle \xrightarrow{M} \ldots \xrightarrow{M} \langle q_k, w_k, u_k\rangle$$
where $q_0 = s$, $w_0 = \triangleright$, $u_0 = x$, $k > 0$, and $q_k \in \{\mathrm{halt}, \mathrm{yes}, \mathrm{no}\}$.

➤ The reflexive transitive closure of $\xrightarrow{M}$ is denoted by $\xrightarrow{M^*}$.

➤ The machine M *accepts* / *rejects* its input x iff $q_k = \mathrm{yes}$ / $q_k = \mathrm{no}$.

Deciding Language Membership

➤ Given an input x, an NTM M may exhibit different computations that can be organized as a *computation tree*:

$$\begin{array}{ccc} & \langle s, \triangleright, 0000\rangle & \\ M^* \swarrow & \ldots & \searrow M^* \\ \langle \mathrm{no}, \triangleright 1, 000\rangle & \ldots & \langle \mathrm{yes}, \triangleright 0000, \sqcup\rangle \end{array}$$

➤ An NTM $M = \langle K, \Sigma, \delta, s\rangle$ decides a *language*, i.e., a set of strings $L \subseteq (\Sigma \setminus \{\sqcup\})^*$, if and only if for all strings $x \in (\Sigma \setminus \{\sqcup\})^*$,
$$x \in L \iff \langle s, \triangleright, x\rangle \xrightarrow{M^*} \langle \mathrm{yes}, w, u\rangle \text{ for some } w \text{ and } u.$$

➤ This definition covers DTMs as special cases of NTMs.

Example. The input 0000 is accepted by the rightmost computation.

Decision Problems

- A *decision problem* is a problem whose *instances* have a simple solution: either an answer “yes” or “no”.
- Consider an instance of PRIMES: Is 561 a prime?
- A decision problem is solved using a DTM or an NTM
  1. by encoding problem instances as strings, and
  2. by constructing a machine M which decides the language L corresponding to the “yes”-instances of the problem.

Example. The famous *satisfiability problem* of propositional logic is about deciding whether the given sentence ϕ is satisfiable or not.

$\Longrightarrow$ The problem can be identified with the language of satisfiable sentences—denoted by SAT.

Fundamental Complexity Classes

- The computational complexity of decision problems can be analyzed by setting resource bounds on TMs that solve them.
- A TM M halts in polynomial time if and only if there is a polynomial p so that for any input $x \in (\Sigma - \{\sqcup\})^*$, any computation of M on x comprises at most $p(|x|)$ configurations.
- The two fundamental time complexity classes are
  1. P: languages decidable in polynomial time using a DTM, and
  2. NP: languages decidable in polynomial time using an NTM.
- The class P is a subclass of NP—and likely to be a proper one.

Theorem. PRIMES and SAT belong to P and NP, respectively.

Reductions

Definition. Let L_1 and L_2 be two languages.

The language L_1 is reducible to L_2 iff there is function R—computable by a DTM M in polynomial time—such that for all inputs x,

$$x \in L_1 \iff R(x) \in L_2.$$

Example. Consider a graph $G = \langle N, E \rangle$ where N and $E \subseteq N \times N$ specify its nodes and edges, respectively.

The question whether G is 3-colorable (language 3COL) can be reduced to propositional satisfiability using $R(G) = R(\langle N, E \rangle) =$

$\{r_n \vee g_n \vee b_n \mid n \in N\} \cup \{\neg r_n \vee \neg r_m,\ \neg g_n \vee \neg g_m,\ \neg b_n \vee \neg b_m \mid \langle n, m \rangle \in E\}$.

Proposition. For any finite G, $G \in$ 3COL if and only if $R(G) \in$ SAT.

Completeness

- Consider any class C of languages (such as P or NP).
- The most demanding languages of C are distinguished as follows.

Definition. A language L—not necessarily contained in C—is

1. C-*hard* if and only if every language $L' \in \mathrm{C}$ is reducible to L in polynomial time, and
2. C-*complete* if and only if $L \in \mathrm{C}$ and L is C-hard.

Theorem. SAT is NP-complete (Cook, 1971).

Remark. No general polynomial-time algorithm that would solve an NP-complete decision problem is known to date.

2. COMPLEXITY RESULTS FOR ASP

A number of decision problems are of interest:

1. *Existence* of a stable model:
   Given a normal logic program P, does P have a stable model?
2. *Brave reasoning* with respect to stable models:
   Given a normal logic program P and an atom $a \in \mathrm{Hb}(P)$:
   Is there a stable model $M \in \mathrm{SM}(P)$ such that a is true in M?
3. *Cautious reasoning* with respect to stable models:
   Given a normal logic program P and an atom $a \in \mathrm{Hb}(P)$:
   Is a true in every stable model $M \in \mathrm{SM}(P)$?

Existence of Stable Models

Definition. The language STABLE is the set of finite normal programs P—represented as strings—such that $\mathrm{SM}(P) \neq \emptyset$.

Proposition. STABLE is in NP and NP-hard/complete.

Proof. 1. It is possible to construct an NTM M which

(i) chooses a model candidate $M \subseteq \mathrm{Hb}(P)$ for the input P,

(ii) computes $\mathrm{LM}(P^M)$ in time polynomial with respect to $||P||$, and

(iii) *accepts* P if $M = \mathrm{LM}(P^M)$ and *rejects* it otherwise.

2. For a set S of clauses, let $R(S) = \{f \leftarrow \sim A, \sim \overline{B}, \sim f. \mid A \vee \neg B \in S\}$ $\cup \{a \leftarrow \sim \overline{a}.\ \ \overline{a} \leftarrow \sim a. \mid a \in \mathrm{Hb}(S)\}$ where shorthands $A = \{a_1, \ldots, a_n\}$, $B = \{b_1, \ldots, b_m\}$, and $\overline{B} = \{\overline{b} \mid b \in B\}$ are used.

For a finite set S of clauses, $S \in \mathrm{SAT} \iff R(S) \in \mathrm{STABLE}$. □

Sketch for a Direct Completeness Proof

- Due to NP-completeness, any nondeterministic polynomial time computation can be reduced to computation of stable models.
- More specifically, one may construct for any NTM M, any string x, and any polynomial p, a normal program $P(M,x,p)$ such that

  M accepts x in at most $p(|x|)$ steps

  $\iff$ the program $P(M,x,p)$ has a stable model.
- Such a *polynomial time* reduction $P(M,x,p)$ describes the effects of $n = p(|x|)$ computation steps in terms of
  1. the state of the tape (n cells) in the beginning,
  2. the possible state transitions of M, and
  3. the final condition for an accepting computation.

Complexity of Brave Reasoning

Definition. The language BRAVE consists of pairs $\langle P, a\rangle$ such that P is a finite normal program, $a \in \mathrm{Hb}(P)$, and $a \in M$ for some $M \in \mathrm{SM}(P)$.

Proposition. BRAVE is in NP and NP-hard/complete.

Proof. 1. For a normal program P and an atom $a \in \mathrm{Hb}(P)$,

$$\langle P, a\rangle \in \mathrm{BRAVE} \iff R_1(P,a) = P \cup \{f \leftarrow \sim a, \sim f.\} \in \mathrm{STABLE}$$

where $f \notin \mathrm{Hb}(P)$ is new so that $\mathrm{Hb}(R_1(P,a)) = \mathrm{Hb}(P) \cup \{f\}$.

2. For a normal program P,

$$P \in \mathrm{STABLE} \iff R_2(P) = \langle P \cup \{f.\}, f\rangle \in \mathrm{BRAVE}$$

where $f \notin \mathrm{Hb}(P)$ is new so that $\mathrm{Hb}(P \cup \{f.\}) = \mathrm{Hb}(P) \cup \{f\}$. □

Complexity of Cautious Reasoning

Definition. CAUTIOUS is the language of pairs $\langle P, a\rangle$ such that P is a finite normal program, $a \in \mathrm{Hb}(P)$, and $a \in M$ for every $M \in \mathrm{SM}(P)$.

Proposition. The complement of CAUTIOUS is in NP and and NP-hard/complete which means that CAUTIOUS is *coNP-complete*.

Proof. 1. For a finite normal program P and an atom $a \in \mathrm{Hb}(P)$,

$$\langle P, a\rangle \notin \text{CAUTIOUS} \iff R_1(P,a) = P \cup \{f \leftarrow a, \sim f.\} \in \text{STABLE}$$

where $f \notin \mathrm{Hb}(P)$ is new so that $\mathrm{Hb}(R_1(P,a)) = \mathrm{Hb}(P) \cup \{f\}$.

2. For a finite normal program P,

$$P \in \text{STABLE} \iff R_2(P) = \langle P \cup \{f \leftarrow f.\}, f\rangle \notin \text{CAUTIOUS}$$

where $f \notin \mathrm{Hb}(P)$ is new so that $\mathrm{Hb}(P \cup \{f \leftarrow f.\}) = \mathrm{Hb}(P) \cup \{f\}$. □

Complexity of `smodels` Programs

- The input language of the `smodels` solver is of interest.
- Analogous hardness results follow immediately from the fact that normal rules form a part of the input language.
- The translations presented so far do not provide a polynomial time reduction from `smodels` programs to normal programs.
- However, the membership of STABLE in NP can be proved as in the case of normal programs using a similar NTM.
- For BRAVE and the complement of CAUTIOUS, the reductions $R_1(P,a)$ presented for normal programs apply as such.
- The language of `lparse` is of much higher time complexity.

3. ORDINALS AND TRANSFINITE INDUCTION

The definition of *ordinal numbers*, or *ordinals* for short, will be based on two properties of sets defined as follows:

Definition. A set S is *transitive* if and only if for every $e \in S$, $e \subseteq S$.

Example. For instance, the set $S = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$ is transitive because it holds that $\emptyset \subseteq S$, $\{\emptyset\} \subseteq S$, and $\{\emptyset, \{\emptyset\}\} \subseteq S$.

Definition. A binary relation $< \subseteq S \times S$ is a *linear order* $<$ on S if and only if $<$ is irreflexive, transitive, and connected, i.e., for every $e_1, e_2 \in S$, $e_1 < e_2$, $e_1 = e_2$, or $e_2 > e_1$.

Definition. A set S is *well-ordered* by a linear order $<$ if and only if for every $\emptyset \subset X \subseteq S$, there is the *least element* $x \in X$ with respect to $<$, i.e., for every $e \in X$, $x \leq e$.

Ordinal Numbers

- An *ordinal number* S is a transitive set well-ordered by $\in$.
- Each well-ordered set is isomorphic to some ordinal (or *order type*).
- The class of all ordinals is well-ordered: $\alpha < \beta \iff \alpha \in \beta$.
- If α and β are ordinals, then either $\alpha \subseteq \beta$ or $\beta \subseteq \alpha$.
- The sum $\alpha + \beta$ of two ordinals α and β denotes the concatenation of the respective well-orders.

Example. Natural numbers correspond to *finite* ordinals:

$$0 \mapsto \emptyset,\ 1 = 0+1 \mapsto \{\emptyset\},\ 2 = 1+1 \mapsto \{\emptyset, \{\emptyset\}\},\ \ldots$$

The set of all natural numbers corresponds to the least infinite ordinal $\omega = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}, \ldots\}$.

Ordinals and Cardinals

Definition.

1. The *successor* $\alpha+1$ of an ordinal α is the ordinal $\alpha\cup\{\alpha\}$.
2. If $\alpha=\beta+1$ for some ordinal β, then α is a *successor ordinal*.
3. An ordinal α which is not a successor ordinal is a *limit ordinal*.
4. If $|\alpha|\neq|\beta|$ for every ordinal $\beta<\alpha$, then α is a *cardinal* number.

Examples.

1. The first two limit ordinals are $\emptyset$ and ω.
2. $2+\omega=\omega$ are $\omega+2$ are not isomorphic as well-ordered sets.
3. The ordinals $2=\{\emptyset,\{\emptyset\}\}$ and ω are cardinals but $\omega+2$ is not ($|\omega|=|\omega+2|$).

The Principle of Transfinite Induction

- Let $P(\alpha)$ be some property defined for an ordinal α.
- Proving the property $P(\alpha)$ for all ordinals α using *transfinite induction* consists of the following tree steps:
  1. In the base case $\alpha=0$, it is proved that $P(0)$.
  2. Then $P(\alpha+1)$ is proved for all successor ordinals $\alpha+1$ assuming that $P(\alpha)$ holds by the inductive hypothesis.
  3. Finally, $P(\beta)$ is proved for all limit ordinals β using the inductive hypothesis that $P(\alpha)$ holds for all ordinals $\alpha<\beta$.

Remark. Transfinite induction is the basic method for proving properties of ordinals, or other objects indexed by ordinals.

4. WELL-FOUNDED SEMANTICS

- Since reasoning with stable models is intractable in general, finding techniques that approximate such reasoning tasks is of interest.
- The well-founded semantics [Van Gelder et al., 1988] provides a sound approximation of stable models.
- Each normal program P is assigned a unique *three-valued* model that can be characterized in terms of the operator Γ_P.

Example. Suppose $M\subseteq \mathrm{Hb}(P)$ is a set of atoms which are known to be true for sure (initially this set could be $\emptyset$). Then

1. $\Gamma_P(M)=\mathrm{LM}(P^M)$ gives atoms that are *potentially true*, and
2. $\Gamma_P^2(M)=\Gamma_P(\Gamma_P(M))$ gives atoms that are true for sure, again.

Properties of the Approximation Operator Γ_P^2

The following results are formulated for normal programs P.

Proposition. The operator Γ_P^2 is monotonic.

Proof. Consider any interpretations $M_1\subseteq M_2\subseteq \mathrm{Hb}(P)$. Since Γ_P is antimonotonic, we obtain $\Gamma_P(M_2)\subseteq\Gamma_P(M_1)$ and $\Gamma_P^2(M_1)\subseteq\Gamma_P^2(M_2)$. □

Corollary. The operator Γ_P^2 has the least fixpoint $\mathrm{lfp}(\Gamma_P^2)$.

Proposition. For all $M\in\mathrm{SM}(P)$, $\mathrm{lfp}(\Gamma_P^2)\subseteq M\subseteq\Gamma_P(\mathrm{lfp}(\Gamma_P^2))$.

Proof. Consider any $M\in\mathrm{SM}(P)$. Let $M_0=\emptyset$, $M_{\alpha+1}=\Gamma_P^2(M_\alpha)$ for all successor ordinals $\alpha+1$ and $M_\beta=\bigcup_{\alpha<\beta}M_\alpha$ for all limit ordinals β. Then $M_\alpha\subseteq M\subseteq\Gamma_P(M_\alpha)$ follows by transfinite induction for any α. □

The Well-Founded Model

- The operator Γ_P yields a lower and an upper bound for $\mathrm{SM}(P)$.
- The fixpoint $\mathrm{lfp}(\Gamma_P^2)$ gives rise to a *partial* (three-valued) model, the *well-founded model* of P. Stable models are *total* (two-valued).
- In contrast with $\mathrm{lfp}(\mathrm{T}_P)$, the fixpoint $\mathrm{lfp}(\Gamma_P^2)$ might not be reached with ω applications of Γ_P^2.

Definition. The well-founded model of a normal program P is characterized by $\mathrm{WFM}(P) = \mathrm{lfp}(\Gamma_P^2) \cup \{\sim a \mid a \in \mathrm{Hb}(P) \setminus \Gamma_P(\mathrm{lfp}(\Gamma_P^2))\}$.

Proposition. If $\mathrm{WFM}(P)$ is total, i.e., $\Gamma_P(\mathrm{lfp}(\Gamma_P^2)) \setminus \mathrm{lfp}(\Gamma_P^2) = \emptyset$, it holds that $\mathrm{SM}(P) = \{\mathrm{lfp}(\Gamma_P^2)\}$.

Example. For the normal program $P = \{a \leftarrow \sim a, \sim b.\ \}$, we have $\Gamma_P(\emptyset) = \{a\}$ and $\Gamma_P^2(\emptyset) = \emptyset$. Thus $\mathrm{WFM}(P) = \{\sim b\}$.

Example

Consider the normal program $Q =$

$$\{\ a_1 \leftarrow \sim a_0. \quad a_2 \leftarrow \sim a_1. \quad a_3 \leftarrow \sim a_2. \quad b_1 \leftarrow a_3, \sim b_2. \quad b_2 \leftarrow a_3, \sim b_1. \ \}.$$

The construction of $\mathrm{lfp}(\Gamma_Q^2)$ proceeds as follows:

1. $\Gamma_Q(\emptyset) = \{a_1, a_2, a_3, b_1, b_2\}$ and $\Gamma_Q^2(\emptyset) = \{a_1\}$.
2. $\Gamma_Q(\{a_1\}) = \{a_1, a_3, b_1, b_2\}$ and $\Gamma_Q^2(\{a_1\}) = \{a_1, a_3\}$.
3. $\Gamma_Q(\{a_1, a_3\}) = \{a_1, a_3, b_1, b_2\}$ and $\Gamma_Q^2(\{a_1, a_3\}) = \{a_1, a_3\}$.

Thus $\mathrm{lfp}(\Gamma_Q^2) = \{a_1, a_3\}$ and $\mathrm{WFM}(Q) = \{a_1, a_3, \sim a_0, \sim a_2\}$ which approximates the two stable models in $\mathrm{SM}(Q) = \{\{a_1, a_3, b_1\}, \{a_1, a_3, b_2\}\}$.

Transfinite Case

Example. Consider the infinite normal program $R = \mathrm{Gnd}(P)$ for a normal program P involving variables and function symbols:

$$R = \{a_{i+1} \leftarrow \sim b_i.\ \ b_i \leftarrow \sim a_i. \ \mid i \geq 0\} \cup \{c \leftarrow a_i. \ \mid i \geq 0\} \cup \{e_{i+1} \leftarrow \sim c, \sim d_i.\ \ d_i \leftarrow \sim c, \sim e_i \mid i \geq 0\}.$$

1. $\Gamma_R^2 \uparrow 0 = \emptyset$.
2. $\Gamma_R^2 \uparrow i = \{b_j \mid 0 \leq j < i\}$.
3. $\Gamma_R^2 \uparrow \omega = \{b_j \mid j \geq 0\}$.
4. $\Gamma_R^2 \uparrow \omega + i = \{b_j \mid j \geq 0\} \cup \{d_j \mid 0 \leq j < i\}$.
5. $\Gamma_R^2 \uparrow \omega + \omega = \{b_j \mid j \geq 0\} \cup \{d_j \mid j \geq 0\} = \mathrm{lfp}(\Gamma_R^2)$.

Thus $\mathrm{WFM}(R) = \{\sim a_j \mid j \geq 0\} \cup \{b_j \mid j \geq 0\} \cup \{\sim c\} \cup \{d \mid j \geq 0\} \cup \{\sim e_j \mid j \geq 0\}$.

Complexity of Well-Founded Reasoning

The effects of approximation become also apparent in the computational complexities associated with the main reasoning tasks.

- Since the existence of the well-founded model is guaranteed the respective decision problem can be answered "yes" constantly.
- Moreover, there is no distinction between brave and cautious reasoning because the well-founded model is also unique.

Proposition. BRAVE = CAUTIOUS is in P and P-hard/complete.

Proof. 1. It is possible to construct a DTM M which (i) computes $M = \mathrm{lfp}(\Gamma_P^2)$ for P and (ii) accepts the input $\langle P, a\rangle$ if and only if $a \in M$.

2. For a set of *Horn clauses* S, $S \in \mathrm{SAT} \iff R(S) = \langle \{a \leftarrow B. \mid a \vee \neg B \in S\} \cup \{f \leftarrow B. \mid \neg B \in C\}, f\rangle \in \mathrm{CAUTIOUS}$. □

OBJECTIVES

- You are familiar with the basic concepts of computational complexity theory (classes P and NP, reductions, and completeness).
- You know the computational complexity results associated with the main reasoning tasks of ASP.
- You know the basics of ordinals and the difference of (ordinary) finite induction and transfinite induction.
- You are able to define well-founded models for normal program and prove simple properties about them.
- You can calculate the well-founded model for simple normal logic programs (by applying Γ_P^2 iteratively).

TIME TO PONDER

Reconsider the technique of encoding AI planning problems and how the accepting computations of an NTM M, time-wise bounded by a polynomial p, could be described in terms of normal rules.

- What is the notion of a situation in the context of NTMs?
- Design a set of relation symbols for the description of situations.
- What kind of operators can be identified?
- How the length of a plan is determined?