Complexity Theory

VU 181.142, SS 2019

4. Boolean Logic

Reinhard Pichler

Institute of Logic and Computation
DBAI Group
TU Wien

22 October, 2019

Outline

4. Boolean Logic
4.1 Syntax and Semantics
4.2 Satisfiability and Validity
4.3 Normal Forms
4.4 Special cases of SAT
4.5 3-SAT
4.6 HORNSAT
4.7 2-SAT
4.8 Efficient Solution of HORNSAT

Motivation

- Logic is the formal basis of many areas of computer science: digital circuit design, programming language semantics, specification and verification, constraint programming, logic programming, databases, artificial intelligence, knowledge representation, machine learning, ...
- In computational complexity theory: Computational problems from logic are of central importance; they can be used to express computation at various levels.

Syntax

Symbols

The syntax of propositional logic (= Boolean logic) (i.e. the set of well-formed propositional formulae) is based on the following symbols:

- Boolean variables (or atoms): $X = \{x_1, x_2, \dots\}$.
- Boolean connectives: $\vee$, $\wedge$, and $\neg$.

Definition

The set of propositional formulae is the smallest set such that

- all Boolean variables are propositional formulae
- if φ_1 and φ_2 are propositional formulae, so are $\neg\varphi_1$, $(\varphi_1 \wedge \varphi_2)$, and $(\varphi_1 \vee \varphi_2)$.

An expression of the form x_i or $\neg x_i$ is called a literal.

Some notational conventions

- Simplified notation: $(((x_1 \vee \neg x_3) \vee x_2) \vee (x_4 \vee (x_2 \vee x_5)))$ is written as $x_1 \vee \neg x_3 \vee x_2 \vee x_4 \vee x_2 \vee x_5$ or $x_1 \vee \neg x_3 \vee x_2 \vee x_4 \vee x_5$.
- Disjunctions and conjunctions involving n members:
  - $\bigvee_{i=1}^{n} \varphi_i$ stands for $\varphi_1 \vee \cdots \vee \varphi_n$.
  - $\bigwedge_{i=1}^{n} \varphi_i$ stands for $\varphi_1 \wedge \cdots \wedge \varphi_n$.
- Frequently appearing abbreviations:
  - An implication $\varphi_1 \rightarrow \varphi_2$ stands for $\neg\varphi_1 \vee \varphi_2$.
  - An equivalence $\varphi_1 \leftrightarrow \varphi_2$ stands for $(\neg\varphi_1 \vee \varphi_2) \wedge (\neg\varphi_2 \vee \varphi_1)$.
- The *dual* (or *complement*) of a literal α is denoted by $\overline{\alpha}$, i.e., let $\alpha \in X$. Then $\overline{\alpha}$ stands for $\neg\alpha$ and $\overline{\neg\alpha}$ stands for α.

Semantics

Motivation

How to interpret Boolean expressions?
Observation: Boolean expressions are propositions that are either true or false. They speak about a world where certain atomic propositions (Boolean variables) are either true or false.

Definition

- A truth assignment T is a mapping from a finite subset $X' \subset X$ to the set of truth values $\{\mathbf{true}, \mathbf{false}\}$.
- Let $X(\varphi)$ be the set of Boolean variables appearing in φ. A truth assignment $T : X' \rightarrow \{\mathbf{true}, \mathbf{false}\}$ is appropriate to φ if $X(\varphi) \subseteq X'$.

Satisfaction relation

Definition

Let a truth assignment $T : X' \rightarrow \{\mathbf{true}, \mathbf{false}\}$ be appropriate to φ, i.e., $X(\varphi) \subseteq X'$. $T \models \varphi$ (T satisfies φ or φ is true in T) is defined inductively as follows:

- If φ is a variable from X', then $T \models \varphi$ iff $T(\varphi) = \mathbf{true}$.
- If $\varphi = \neg\varphi_1$, then $T \models \varphi$ iff $T \not\models \varphi_1$.
- If $\varphi = \varphi_1 \wedge \varphi_2$, then $T \models \varphi$ iff $T \models \varphi_1$ and $T \models \varphi_2$.
- If $\varphi = \varphi_1 \vee \varphi_2$, then $T \models \varphi$ iff $T \models \varphi_1$ or $T \models \varphi_2$.

Example

Let $T(x_1) = \mathbf{true}$, $T(x_2) = \mathbf{false}$.
Then $T \models x_1 \vee x_2$
but $T \not\models (x_1 \vee \neg x_2) \wedge (\neg x_1 \wedge x_2)$.

Logical equivalence

Definition

Expressions φ_1 and φ_2 are logically equivalent ($\varphi_1 \equiv \varphi_2$) iff for all truth assignments T appropriate to both of them,

$$T \models \varphi_1 \text{ iff } T \models \varphi_2.$$

Example

$(\varphi_1 \vee \varphi_2) \equiv (\varphi_2 \vee \varphi_1)$
$((\varphi_1 \wedge \varphi_2) \wedge \varphi_3) \equiv (\varphi_1 \wedge (\varphi_2 \wedge \varphi_3))$
$\neg\neg\varphi \equiv \varphi$
$((\varphi_1 \wedge \varphi_2) \vee \varphi_3) \equiv ((\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3))$
$\neg(\varphi_1 \wedge \varphi_2) \equiv (\neg\varphi_1 \vee \neg\varphi_2)$
$(\varphi_1 \vee \varphi_1) \equiv \varphi_1$

Satisfiability and Validity

Definition

- A Boolean expression φ is satisfiable iff there is a truth assignment T appropriate to it with $T \models \varphi$.
- A Boolean expression φ is valid/a tautology (denoted by $\models \varphi$) iff for every truth assignment T appropriate to it, $T \models \varphi$.

Proposition

- *The following interconnection between satisfiability and validity holds:* φ *is valid* $\Leftrightarrow \neg\varphi$ *is unsatisfiable.*
- *Moreover, for any Boolean expressions* ψ_1 *and* ψ_2*,*

$$\psi_1 \equiv \psi_2 \text{ iff } \models \psi_1 \leftrightarrow \psi_2 \text{ iff } \neg(\psi_1 \leftrightarrow \psi_2) \text{ is unsatisfiable.}$$

Decision Problems

SAT

INSTANCE: Boolean formula φ.

QUESTION: Is φ satisfiable?

VALIDITY

INSTANCE: Boolean formula φ.

QUESTION: Is φ valid?

Complexity of SAT

Cook-Levin Theorem

SAT is NP-complete.

Proof of the membership

SAT can be decided by the following NP-algorithm:

1. Guess a truth assignment T appropriate to φ.
2. Check that $T \models \varphi$.

Proof sketch of the hardness

Idea. Let L be an arbitrary problem (= language) in NP.
We describe a reduction R which for each string x constructs a Boolean formula $R(x)$ such that $x \in L \Leftrightarrow R(x)$ is satisfiable.

Proof sketch of the hardness (continued)

Let T be a single-string NTM that decides L in $q(|x|)$ for any input x for some polynomial $q(.)$. W.l.o.g., we assume that any computation of T takes exactly $N = q(|x|)$ steps for any input x.

Now let x be an arbitrary instance of problem L. Then we construct a Boolean formula $R(x)$ over the following propositional atoms:

$symbol_\sigma[\tau, \pi]$ for $0 \leq \tau \leq N$, $0 \leq \pi \leq N$ and $\sigma \in \Sigma$.
Intuitive meaning: at instant τ of the computation, cell number π contains symbol σ.

$cursor[\tau, \pi]$ for $0 \leq \tau \leq N$ and $0 \leq \pi \leq N$.
Intuitive meaning: at instant τ, the cursor points to cell number π.

$state_s[\tau]$ for $0 \leq \tau \leq N$ and $s \in K$.
Intuitive meaning: at instant τ, the NTM T is in state s.

Proof sketch (continued)

The formula $R(x)$ contains the following groups of conjuncts:

1. Initialization facts. Let $x = x_1 \ldots x_n$.
Then $R(x)$ contains the following atoms as conjuncts:

$symbol_{\triangleright}[0, 0]$
$symbol_{\sigma}[0, \pi]$ for $1 \leq \pi \leq |x|$, where $x_\pi = \sigma$
$symbol_{\sqcup}[0, \pi]$ for $|x| < \pi \leq N$
$cursor[0, 0]$
$state_{s_0}[0]$

2. Transition rules. For each pair (s, σ) of state s and symbol σ let $\langle s, \sigma, s'_1, \sigma'_1, d_1 \rangle, \ldots, \langle s, \sigma, s'_k, \sigma'_k, d_k \rangle$ denote all possible transitions according to the transition relation Δ (for the cursor movements, we write $d_i \in \{-1, 0, 1\}$ rather than $d_i \in \{\leftarrow, -, \rightarrow\}$).
Then $R(x)$ contains the following conjuncts for each value of τ and π such that $0 \leq \tau < N$ and $0 \leq \pi < N$

Proof sketch (continued)

$state_s[\tau] \wedge symbol_\sigma[\tau, \pi] \wedge cursor[\tau, \pi] \rightarrow$
$[(state_{s'_1}[\tau + 1] \wedge symbol_{\sigma'_1}[\tau + 1, \pi] \wedge cursor[\tau + 1, \pi + d_1]) \vee \cdots \vee$
$(state_{s'_k}[\tau + 1] \wedge symbol_{\sigma'_k}[\tau + 1, \pi] \wedge cursor[\tau + 1, \pi + d_k])]$

3. Uniqueness constraints. Let $K = \{s_0, \ldots, s_m\}$ and $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$.
Then $R(x)$ contains the following formulae for each value of τ and π such that $0 \leq \tau \leq N$, $0 \leq \pi \leq N$, $0 \leq i \leq m$, and $1 \leq j \leq n$.

$state_{s_i}[\tau] \leftrightarrow (\neg state_{s_0}[\tau] \wedge \cdots \wedge \neg state_{s_{i-1}}[\tau] \wedge$
$\wedge \neg state_{s_{i+1}}[\tau] \wedge \cdots \wedge \neg state_{s_m}[\tau])$
$cursor[\tau, \pi] \leftrightarrow (\neg cursor[\tau, 0] \wedge \cdots \wedge \neg cursor[\tau, \pi - 1] \wedge$
$\wedge \neg cursor[\tau, \pi + 1] \wedge \cdots \wedge \neg cursor[\tau, N])$
$symbol_{\sigma_j}[\tau, \pi] \leftrightarrow (\neg symbol_{\sigma_1}[\tau, \pi] \wedge \cdots \wedge \neg symbol_{\sigma_{j-1}}[\tau, \pi] \wedge$
$\wedge \neg symbol_{\sigma_{j+1}}[\tau, \pi] \wedge \cdots \wedge \neg symbol_{\sigma_n}[\tau, \pi])$

Proof sketch (continued)

4. Inertia rules. $R(x)$ contains the following conjuncts for each value τ, π, π', σ, where $0 \leq \tau < N$, $0 \leq \pi < \pi' \leq N$, and $\sigma \in \Sigma$,

$symbol_\sigma[\tau, \pi] \wedge cursor[\tau, \pi'] \rightarrow symbol_\sigma[\tau + 1, \pi]$
$symbol_\sigma[\tau, \pi'] \wedge cursor[\tau, \pi] \rightarrow symbol_\sigma[\tau + 1, \pi']$

5. Acceptance. Let $s_m =$ "yes".
Then $R(x)$ contains the following atom as a conjunct:

$state_{s_m}[N]$.

Correctness and complexity of this reduction. This reduction is clearly feasible in logarithmic space (since we never have to handle different parts of intermediate results simultaneously). Moreover, it is straightforward to show the following equivalence:
$x \in L$ (i.e., there exists an accepting computation of the NTM T on input x) iff $R(x)$ is satisfiable.

Complexity of VALIDITY

Corollary

VALIDITY *is* co-NP-*complete.*

Proof

Recall the following equivalences:
φ is valid $\Leftrightarrow$ $\neg\varphi$ is unsatisfiable and
φ is unsatisfiable $\Leftrightarrow$ $\neg\varphi$ is valid.

Membership. **VALIDITY** can be reduced to the co-**SAT**-problem.
Since **SAT** is in NP, co-**SAT** is in co-NP and so is **VALIDITY**.

Hardness. co-**SAT**can be reduced to the **VALIDITY**-problem.
Since **SAT** is NP-hard, co-**SAT** is co-NP-hard and so is **VALIDITY**.

Normal Forms

Definition

- A formula is in CNF (= Conjunctive Normal Form) if it is of the form $(l_{11} \vee \cdots \vee l_{1n_1}) \wedge \cdots \wedge (l_{m1} \vee \cdots \vee l_{mn_m})$
- A formula is in DNF (= Disjunctive Normal Form) if it is of the form $(l_{11} \wedge \cdots \wedge l_{1n_1}) \vee \cdots \vee (l_{m1} \wedge \cdots \wedge l_{mn_m})$

where each l_{ij} is a literal (i.e., a Boolean variable or its negation).

Definition

- A disjunction $l_1 \vee \cdots \vee l_n$ of literals is called a clause.
- A conjunction $l_1 \wedge \cdots \wedge l_n$ of literals is called an implicant.
- We may assume that normal forms do not have repeated clauses/implicants or repeated literals in clauses/implicants. Example. $(\neg x_1 \vee \neg x_1 \vee x_2) \equiv (\neg x_1 \vee x_2)$.

CNF/DNF transformation

Theorem

Every Boolean expression is equivalent to one in conjunctive (disjunctive) normal form CNF (DNF).

Proof sketch

Transformation into a mixed conjunction and disjunction of literals:

- Remove $\leftrightarrow$ and $\rightarrow$:
  $$\alpha \leftrightarrow \beta \quad \rightsquigarrow \quad (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha) \qquad (1)$$
  $$\alpha \rightarrow \beta \quad \rightsquigarrow \quad \neg\alpha \vee \beta \qquad (2)$$
- Push negations in front of Boolean variables:
  $$\neg\neg\alpha \quad \rightsquigarrow \quad \alpha \qquad (3)$$
  $$\neg(\alpha \vee \beta) \quad \rightsquigarrow \quad \neg\alpha \wedge \neg\beta \qquad (4)$$
  $$\neg(\alpha \wedge \beta) \quad \rightsquigarrow \quad \neg\alpha \vee \neg\beta \qquad (5)$$

CNF/DNF transformation

Proof sketch (continued)

The next phase depends on the normal form being pursued:

- For a CNF, move $\wedge$ connectives outside $\vee$ connectives:
  $$\alpha \vee (\beta \wedge \gamma) \quad \rightsquigarrow \quad (\alpha \vee \beta) \wedge (\alpha \vee \gamma) \qquad (6)$$
  $$(\alpha \wedge \beta) \vee \gamma \quad \rightsquigarrow \quad (\alpha \vee \gamma) \wedge (\beta \vee \gamma) \qquad (7)$$
- For a DNF, move $\vee$ connectives outside $\wedge$ connectives:
  $$\alpha \wedge (\beta \vee \gamma) \quad \rightsquigarrow \quad (\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \qquad (8)$$
  $$(\alpha \vee \beta) \wedge \gamma \quad \rightsquigarrow \quad (\alpha \wedge \gamma) \vee (\beta \wedge \gamma) \qquad (9)$$

Note. In the worst case, an equivalent normal form can be exponentially bigger than the original expression.

Example. Consider deriving a CNF for $(x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$.

Example

Example

Transform $(x_1 \vee x_2) \rightarrow (x_2 \leftrightarrow x_3)$ into CNF.

$(x_1 \vee x_2) \rightarrow (x_2 \leftrightarrow x_3) \quad \rightsquigarrow^{(2)}$

$\neg(x_1 \vee x_2) \vee (x_2 \leftrightarrow x_3) \quad \rightsquigarrow^{(1)}$

$\neg(x_1 \vee x_2) \vee ((\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_2)) \quad \rightsquigarrow^{(4)}$

$(\neg x_1 \wedge \neg x_2) \vee ((\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_2)) \quad \rightsquigarrow^{(7)}$

$(\neg x_1 \vee ((\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_2))) \wedge (\neg x_2 \vee ((\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_2))) \rightsquigarrow^{(6)}$

$((\neg x_1 \vee (\neg x_2 \vee x_3)) \wedge (\neg x_1 \vee (\neg x_3 \vee x_2))) \wedge$
$\quad \wedge (\neg x_2 \vee ((\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_2))) \quad \rightsquigarrow^{(6)}$

$((\neg x_1 \vee (\neg x_2 \vee x_3)) \wedge (\neg x_1 \vee (\neg x_3 \vee x_2))) \wedge$
$\quad \wedge ((\neg x_2 \vee (\neg x_2 \vee x_3)) \wedge (\neg x_2 \vee (\neg x_3 \vee x_2))) \quad \rightsquigarrow^{(\text{simplification})}$

$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_2) \wedge (\neg x_2 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_2)$
$\rightsquigarrow^{(\text{simplification})}$

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

SAT and CNF

Theorem

There exists a log-space reduction which reduces any Boolean expression φ into a sat-equivalent Boolean expression ψ in conjunctive normal form, i.e.: φ is satisfiable $\Leftrightarrow$ ψ is satisfiable.

Proof sketch

- By de Morgan's laws and the equivalence $\neg\neg\alpha \equiv \alpha$, negation can be shifted immediately in front of atoms (cf. rewrite rules (4)+(5) resp. (3) above).
- The CNF can then be obtained from the resulting formula by successive applications of the following rewrite rule
  $(A \wedge B) \vee C \rightsquigarrow (z \vee A) \wedge (z \vee B) \wedge (\neg z \vee C)$
  for some fresh variable z.

Proof sketch (continued)

It can be seen as follows that the above rewrite rule produces a sat-equivalent formula:

- Suppose that $\Phi = (A \wedge B) \vee C$ is satisfiable, i.e., there exists a satisfying truth assignment $\mathcal{I}$. We show that $\mathcal{I}$ can be extended to a satisfying truth assignment $\mathcal{J}$ of $\Psi = (z \vee A) \wedge (z \vee B) \wedge (\neg z \vee C)$
  Case 1: Suppose that $(A \wedge B)$ is true in $\mathcal{I}$. We set $\mathcal{J}(z) =$ **false**. Clearly, all conjuncts of Ψ are true in $\mathcal{J}$.
  Case 2: Suppose that C is true in $\mathcal{I}$. Then we set $\mathcal{J}(z) =$ **true**. Again, all conjuncts of Ψ are true in $\mathcal{J}$.
- Suppose that $\Psi = (z \vee A) \wedge (z \vee B) \wedge (\neg z \vee C)$ is satisfiable, i.e., there exists a satisfying truth assignment $\mathcal{J}$ of Ψ. We claim that then $\mathcal{J}$ is also a satisfying truth assignment of $\Phi = (A \wedge B) \vee C$.
  Case 1: Suppose that $\mathcal{J}(z) =$ **true**. By the conjunct $(\neg z \vee C)$ in Ψ, C (and thus Φ) is true in $\mathcal{J}$.
  Case 2: Suppose that $\mathcal{J}(z) =$ **false**. By the conjuncts $(z \vee A) \wedge (z \vee B)$ in Ψ, both A and B (and thus Φ) are true in $\mathcal{J}$.

SAT and CNF

Theorem

There exists a log-space reduction which reduces any Boolean expression φ into a sat-equivalent Boolean expression ψ in 3-CNF, i.e.: ψ is in CNF and every clause of ψ consists of exactly 3 literals.

Proof sketch

By the above theorem, we may assume that φ is in CNF.

Case 1. If a clause is "too small":
A clause of the form $c = l_1 \vee l_2$ may be replaced by the two clauses $c_1 = z \vee l_1 \vee l_2$ and $c_2 = \neg z \vee l_1 \vee l_2$

Likewise, a clause $c = l_1$ is replaced by 4 clauses $c_1 = x \vee y \vee l_1$, $c_2 = x \vee \neg y \vee l_1$, $c_3 = \neg x \vee y \vee l_1$ and $c_4 = \neg x \vee \neg y \vee l_1$.

Note that the result of this step is an equivalent formula.

SAT and CNF

Proof sketch (continued)

Case 2. If a clause is "too big":
Any clause of the form $c = l_1 \vee l_2 \vee l_3 \vee R$ may be replaced by the two clauses $c_1 = \neg z \vee l_3 \vee R$ and $c_2 = z \vee l_1 \vee l_2$

It remains to show that the result of this step is a sat-equivalent formula. We only show one direction: Suppose that $c = l_1 \vee l_2 \vee l_3 \vee R$ has a satisfying assignment $\mathcal{I}$. Then at least one of the disjuncts in c is true.

If l_1 or l_2 is true in $\mathcal{I}$, then we extend $\mathcal{I}$ to $\mathcal{J}$ with $\mathcal{J}(z) =$ **false**. Obviously, both c_1 and c_2 are true in $\mathcal{J}$.

If l_3 or R is true in $\mathcal{I}$, then we extend $\mathcal{I}$ to $\mathcal{J}$ with $\mathcal{J}(z) =$ **true**. Again, both c_1 and c_2 are true in $\mathcal{J}$.

Special cases of SAT

3-SAT

INSTANCE: Boolean formula φ in 3-CNF.

QUESTION: Is φ satisfiable?

2-SAT

INSTANCE: Boolean formula φ in 2-CNF (i.e., each clause consists of exactly 2 literals).

QUESTION: Is φ satisfiable?

HORNSAT

INSTANCE: Boolean formula φ in CNF, s.t. each clause is in Horn form (i.e., each clause contains at most one positive literal).

QUESTION: Is φ satisfiable?

3-SAT

Theorem

3-SAT *is* NP-*complete.*

Proof

- Membership is clear since **3-SAT** is a special case of **SAT**.
- Hardness follows from the NP-hardness of **SAT** and from the fact that any Boolean expression φ can be reduced into a sat-equivalent Boolean expression ψ in 3-CNF (i.e.: ψ is in CNF and every clause of ψ consists of exactly 3 literals).

Corollary

The **VALIDITY***-problem remains* co-NP-*complete even if the formulae are restricted to 3-DNF (i.e., DNF where each implicant consists of exactly 3 literals).*

HORNSAT

Theorem

HORNSAT *is* P-*complete.*

Proof sketch of the membership

Observation. A Horn clause c is of one of the following forms:

- facts: $c = p$ (i.e., a single positive literal).
- rules: $c = \neg q_1 \vee \cdots \vee \neg q_n \vee p$ with $n \geq 1$, which is usually denoted as $q_1 \wedge \cdots \wedge q_n \rightarrow p$
- goals: $c = \neg q_1 \vee \cdots \vee \neg q_n$

Proof sketch of the membership (continued)

Idea of the SAT-test.

- Compute the set Y of all variables that are logically implied by the facts and rules in φ.
- If there exists a goal $\neg q_1 \vee \cdots \vee \neg q_n$ in φ, s.t. every q_i is in Y, then φ is unsatisfiable.
- Otherwise φ is satisfiable. Indeed, we get a model of φ by setting all propositional variables in Y to **true** and all other variables to **false**.

Computation of the variables Y that are logically implied by φ.

- Initially, let $Y :=$ set of facts in the formula φ.
- Iteratively apply the "immediate consequence operator" to Y, i.e.: If there exists a rule $q_1 \wedge \cdots \wedge q_n \rightarrow p$, s.t. $\{q_1, \ldots, q_n\} \subseteq Y$ but $p \notin Y$ then set $Y := Y \cup \{p\}$.
- The naive implementation of this algorithm requires quadratic time (whenever a variable is added to Y, check if some new rule can now be applied; there are only linearly many variables to be ever added).

Proof sketch of the hardness

We can proceed exactly as in the proof of the Cook-Levin Theorem. We only have to make sure that the conjuncts in the resulting formula $R(x)$ are (transformed into) Horn clauses.

1. Initialization facts. No changes required. Conjuncts in $R(x)$:

$symbol_{\triangleright}[0,0]$, $symbol_{\sigma}[0,\pi]$, $symbol_{\sqcup}[0,\pi]$, $cursor[0,0]$, $state_{s_0}[0]$

2. Transition rules. Now T is a deterministic TM. For each pair (s,σ) of state s and symbol σ there exists exactly one possible transition $\langle s,\sigma,s',\sigma',d\rangle$ according to the transition function δ. Conjuncts in $R(x)$:

$state_s[\tau] \wedge symbol_{\sigma}[\tau,\pi] \wedge cursor[\tau,\pi] \rightarrow state_{s'}[\tau+1]$
$state_s[\tau] \wedge symbol_{\sigma}[\tau,\pi] \wedge cursor[\tau,\pi] \rightarrow symbol_{\sigma'}[\tau+1,\pi]$
$state_s[\tau] \wedge symbol_{\sigma}[\tau,\pi] \wedge cursor[\tau,\pi] \rightarrow cursor[\tau+1,\pi+d]$

Proof sketch of the hardness (continued)

3. Uniqueness constraints. It suffices to express the condition that, at any time instant τ, the machine is in *at most* one state, the cursor is in *at most* one position, and each tape cell contains *at most* one symbol, e.g.: for all $i \neq j$, $\pi \neq \pi'$, $R(x)$ contains conjuncts of the following form:

$\neg state_{s_i}[\tau] \vee \neg state_{s_j}[\tau]$
$\neg cursor[\tau,\pi] \vee \neg cursor[\tau,\pi']$
$\neg symbol_{\sigma_i}[\tau,\pi] \vee \neg symbol_{\sigma_j}[\tau,\pi]$

4. Inertia rules. No changes required. Conjuncts in $R(x)$:

$symbol_{\sigma}[\tau,\pi], cursor[\tau,\pi'] \rightarrow symbol_{\sigma}[\tau+1,\pi]$
$symbol_{\sigma}[\tau,\pi'], cursor[\tau,\pi] \rightarrow symbol_{\sigma}[\tau+1,\pi']$

5. Acceptance. No changes required. Conjunct in $R(x)$:

$state_{s_m}[N]$, where $s_m =$ "yes".

2-SAT

Theorem

2-SAT *is* co-NL-*complete.*

Proof sketch of the membership

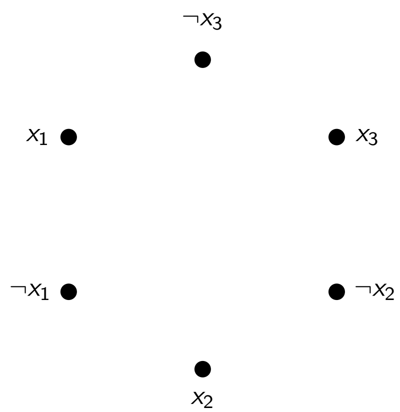
Recall the **2-SAT**-algorithm from the "Formale Methoden" lecture. Given an arbitrary instance φ of **2-SAT**, we define the graph $G(\varphi)$ as follows:

- The variables of φ and their negations form the vertices of $G(\varphi)$.
- There is an arc (α,β) iff there is a clause $\overline{\alpha} \vee \beta$ or $\beta \vee \overline{\alpha}$ in φ, where $\overline{\alpha}$ is the complement of α, i.e.: If α is true in some satisfying assignment $\mathcal{I}$ of φ, then β must also be true in $\mathcal{I}$.
- It can be shown that φ is unsatisfiable iff there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in $G(\varphi)$.

NL-algorithm for the unsatisfiability of φ. Guess a variable x and check that x is reachable from $\neg x$ and $\neg x$ is reachable from x in $G(\varphi)$.

Example

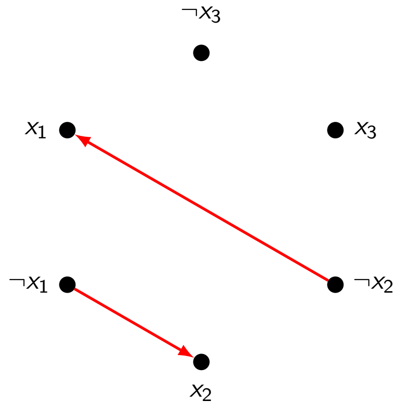
$$\psi = (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_2 \vee x_3)$$



Example

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

$$\begin{array}{ccc} \neg x_1 & \Rightarrow & x_2 \\ \neg x_2 & \Rightarrow & x_1 \end{array}$$

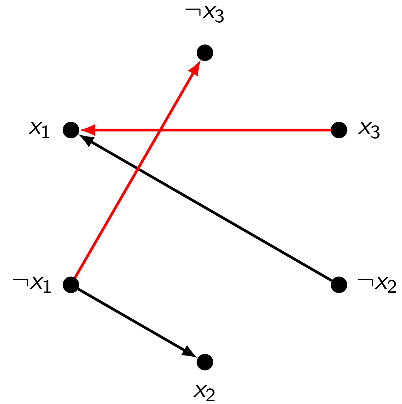


Complexity Theory | 4. Boolean Logic | 4.7. 2-SAT

Example

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

$$\begin{array}{ccc} \neg x_1 & \Rightarrow & \neg x_3 \\ x_3 & \Rightarrow & x_1 \end{array}$$



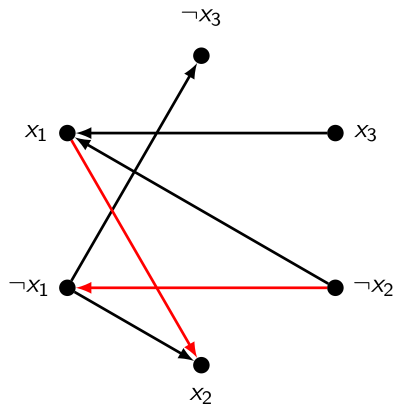
Reinhard Pichler | 22 October, 2019 | Page 32

Complexity Theory | 4. Boolean Logic | 4.7. 2-SAT

Example

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

$$\begin{array}{ccc} x_1 & \Rightarrow & x_2 \\ \neg x_2 & \Rightarrow & \neg x_1 \end{array}$$



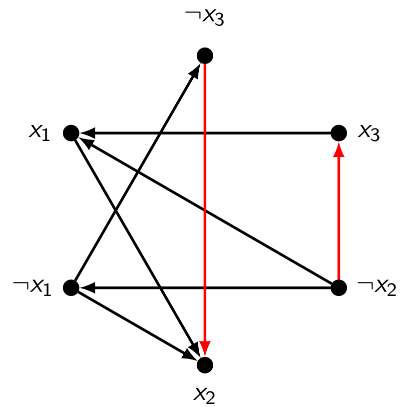
Reinhard Pichler | 22 October, 2019 | Page 32

Complexity Theory | 4. Boolean Logic | 4.7. 2-SAT

Example

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

$$\begin{array}{ccc} \neg x_2 & \Rightarrow & x_3 \\ \neg x_3 & \Rightarrow & x_2 \end{array}$$



Reinhard Pichler | 22 October, 2019 | Page 32

Proof sketch of the hardness

We reduce **REACHABILITY** to the **co-2-SAT**-problem. Let (G, s, t) be an arbitrary instance of **REACHABILITY**, where $G = (V, E)$ is a graph and s, t are nodes in V. W.l.o.g., we may assume that G contains no isolated nodes (i.e., every node in V is adjacent to at least one edge).

We construct the following instance $R(G, s, t) = \varphi$ of **co-2-SAT**:

- The set of variables in φ is V.
- For every edge (a, b) in G, the formula φ contains the clause $\neg a \vee b$ (or, equivalently $a \rightarrow b$).
- Finally, the formula φ also contains the unit clauses s and $\neg t$.

Clearly, this reduction is feasible in log-space. It remains to prove its correctness.

Lemma

Consider the problem reduction from **REACHABILITY** *to* **co-2-SAT** *described above. Let ψ denote the subformula of φ, s.t. ψ consists of the rules only, i.e., $\varphi = \psi \wedge s \wedge \neg t$. Then, for every $v \in V$, the following equivalence holds: v is reachable from s $\Leftrightarrow$ $(\psi \wedge s) \models v$ (i.e.: every model of $\psi \wedge s$ is a model of v.)*

Proof sketch of the hardness (continued)

It remains to show: t is reachable from s in G $\Leftrightarrow$ φ is unsatisfiable

"$\Rightarrow$" Suppose that t is reachable from s. Then (by the above lemma), every model of $(\psi \wedge s)$ is a model of t. Hence, there exists no interpretation in which both $(\psi \wedge s)$ and $\neg t$ are true, i.e.: φ is unsatisfiable.

"$\Leftarrow$" Suppose that t is *not* reachable from s. Then (by the lemma), t is *not* implied by $(\psi \wedge s)$. Hence, there exists an interpretation $\mathcal{I}$ in which $(\psi \wedge s)$ is true while t is false, i.e., $\mathcal{I}$ is a model of $\varphi = (\psi \wedge s) \wedge \neg t$, i.e.: φ is satisfiable.

Remarks

- By the Immerman-Szelepscényi Theorem we know that co-NL = NL.
- Hence, the co-NL-completeness of **2-SAT** immediately implies that **2-SAT** is also NL-complete.
- Three of the most fundamental complexity classes (NL, P, NP) have thus been shown to contain natural variants of **SAT** as complete problems, namely **2-SAT**, **HORNSAT**, and **3-SAT**, respectively.

Efficient Solution of HORNSAT

Basic Step

INPUT: a set X of variables, set Π of rules (a prop. logic program)
OUTPUT: Compute the least fixed-point (denoted as X^+) of the immediate consequence operator w.r.t. rules Π applied to X.

Remarks

- X^+ contains all variables derivable by Π from X.
- Clearly, for every variable z, we have $z \in X^+$ iff $X \cup \Pi \models z$.

Motivation

The basic step occurs (in different terminology) in several areas of computer science, like deriving functional dependencies in a relational schema, graph reachability, reachability in a CFG, etc.

Computing the least fixed-point X^+

Proposition

Let Π be a set of rules over variables V and let $X \subseteq V$. The least fixed-point X^+ of X can be computed in polynomial time.

Proof

A straightforward polynomial-time algorithm works as follows:

$Y := X$;
while $\exists r \in \Pi$, s.t. $body(r) \subseteq Y$ and $head(r) \notin Y$ **do**
  $Y := Y \cup \{head(r)\}$;
endwhile;
return Y;

Complexity of Computing X^+

Motivation

Complexity of computing X^+ for some subset $X \subseteq V$

- Straightforward algorithm: works in time $O(|V|^2 \cdot |\Pi|)$.
  - number of iterations: $|V|$
  - in each iteration, scan through all rules $r \in \Pi$ once $\Rightarrow |\Pi|$ upper bound.
  - Check if $body(r) \subseteq Y$ holds $\Rightarrow |V|$ upper bound.
- In (Beeri/Bernstein, 1979), it was shown (for the corresponding problem on functional dependencies) that X^+ can be computed in linear time. More precisely, the algorithm works in time $O(|V| + \| \Pi \|)$.

Algorithm of Beeri and Bernstein

Data structures

Input: Set of rules Π over V, variable set $X \subseteq V$
Output: Least fixed point X^+ w.r.t. immediate consquence operator
Auxiliary data structures:
  count: array of integers, index: each $r \in \Pi$,
  L: array of lists of rules, index: each $z \in V$

Initialization

unmark all members of V;
for each $z \in V$ **do** $L(z)$:= empty-list;
for each $r \in \Pi$ **do**
  count$(r) := |body(r)|$;
  for each $z \in body(r)$ **do** add r to the list $L(z)$;
end for;

Algorithm of Beeri and Bernstein (continued)

Computation of X^+

$Y := X$;
while Y contains an unmarked element z **do**
  mark z;
  for each $r \in L(z)$ **do**
    count(r) := count(r) - 1;
    if count$(r) = 0$ **then** $Y := Y \cup \{head(r)\}$;
  end for;
end while;
return Y;

Algorithm of Beeri and Bernstein (continued)

Correctness of the algorithm (rough proof sketch)

The proof goes via the following loop invariant of the while-loop:

- All marked elements are in Y,
- $Y \subseteq X^+$,
- For all $r \in \Pi$, we have count$(r) = |\{z \in body(r) \mid z \text{ is not marked}\}|$.

Upper bound $\mathrm{O}(|V| + \| \Pi \|)$ on time complexity

- Initialization: takes time $\mathrm{O}(|V| + \| \Pi \|)$
- while-loop:
  - Each element is marked at most once.
  - Altogether, the counts are decremented at most $\| \Pi \|$ times.
  - The innermost loop goes through all the heads of rules once. Hence, once more $\mathrm{O}(\| \Pi \|)$ is needed.

Decision Procedure for HORNSAT

HORNSAT

INSTANCE: Boolean formula φ in CNF over the propositional variables V, s.t. each clause is in Horn form (i.e., has at most one positive literal).
QUESTION: Is φ satisfiable?

Decision Procedure for HORNSAT

Related closure computation problem:
$X = \{p \mid p \text{ is a fact in } \varphi\}$.
$\Pi = \{q_1 \wedge \cdots \wedge q_k \to p \mid \neg q_1 \vee \cdots \vee \neg q_k \vee p \text{ is a rule in } \varphi\} \cup$
$\{q_1 \wedge \cdots \wedge q_k \to \bot \mid \neg q_1 \vee \cdots \vee \neg q_k \text{ is a goal in } \varphi\}$.

Critierion for the Satisfiability of φ:
φ is satisfiable, iff $\bot \notin X^+$ holds.

Related problems

REACHABILITY (in a directed graph)

INSTANCE: directed graph $G = (V, E)$, vertices s, t
QUESTION: Is there a path in G from s to t?

Related closure computation problem:
V as above, $\Pi = \{A \to B \mid (A, B) \in E\}$, $X = \{s\}$.
QUESTION: Does $t \in X^+$ hold?

Functional dependencies in a relational schema

INSTANCE: relational schema (R, F), set of attributes $X \subseteq R$
QUESTION: Which attributes from R are functionally dependent on X?

Related closure computation problem:
$V = R$, $\Pi = \{A_1 \wedge \cdots \wedge A_k \to B \mid (A_1 \ldots A_k \to B) \in F\}$..
QUESTION: Compute X^+.

Corollary

All these problems can be solved in linear time.

Learning Objectives

- Recapitulation of the syntax and semantics of Boolean expressions (= propositional formulae).
- Satisfiability and validity of Boolean expressions.
- Normal forms of Boolean expressions: CNF, DNF, 3-CNF, 2-CNF.
- Difference between equivalence and sat-equivalence.
- Two fundamental NP-complete decision problems: **SAT** and **3-SAT**.
- Two tractable special cases of **SAT**: **HORNSAT** and **2-SAT**.
- Linear-time algorithm for **HORNSAT** and related problems.