Complexity Theory
VU 181.142, SS 2017

4. Boolean Logic

Reinhard Pichler

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

28 March, 2017
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): \( X = \{x_1, x_2, \ldots \} \).
- Boolean connectives: \( \lor, \land, \text{ and } \neg \).

Definition

The set of propositional formulae is the smallest set such that

- all Boolean variables are propositional formulae
- if \( \varphi_1 \) and \( \varphi_2 \) are propositional formulae, so are \( \neg \varphi_1, (\varphi_1 \land \varphi_2) \), and \( (\varphi_1 \lor \varphi_2) \).

An expression of the form \( x_i \) or \( \neg x_i \) is called a literal.
Some notational conventions

- Simplified notation: \(((x_1 \lor \neg x_3) \lor x_2) \lor (x_4 \lor (x_2 \lor x_5))\) is written as \(x_1 \lor \neg x_3 \lor x_2 \lor x_4 \lor x_2 \lor x_5\) or \(x_1 \lor \neg x_3 \lor x_2 \lor x_4 \lor x_5\).

- Disjunctions and conjunctions involving \(n\) members:
  - \(\bigvee_{i=1}^{n} \varphi_i\) stands for \(\varphi_1 \lor \cdots \lor \varphi_n\).
  - \(\bigwedge_{i=1}^{n} \varphi_i\) stands for \(\varphi_1 \land \cdots \land \varphi_n\).

- Frequently appearing abbreviations:
  - An implication \(\varphi_1 \rightarrow \varphi_2\) stands for \(\neg \varphi_1 \lor \varphi_2\).
  - An equivalence \(\varphi_1 \leftrightarrow \varphi_2\) stands for \((\neg \varphi_1 \lor \varphi_2) \land (\neg \varphi_2 \lor \varphi_1)\).

- The dual (or complement) of a literal \(\alpha\) is denoted by \(\overline{\alpha}\), i.e., let \(\alpha \in X\). Then \(\overline{\alpha}\) stands for \(\neg \alpha\) and \(\overline{\overline{\alpha}}\) stands for \(\alpha\).
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 \( \{ \text{true}, \text{false} \} \).
- Let \( X(\varphi) \) be the set of Boolean variables appearing in \( \varphi \). A truth assignment \( T : X' \rightarrow \{ \text{true}, \text{false} \} \) is appropriate to \( \varphi \) if \( X(\varphi) \subseteq X' \).
Satisfaction relation

Definition

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

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

Example

Let $T(x_1) = \text{true}$, $T(x_2) = \text{false}$. Then $T \models x_1 \lor x_2$ but $T \not\models (x_1 \lor \neg x_2) \land (\neg x_1 \land x_2)$. 
Logical equivalence

**Definition**

Expressions $\varphi_1$ and $\varphi_2$ 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**

$$
(\varphi_1 \lor \varphi_2) \equiv (\varphi_2 \lor \varphi_1)
$$

$$
((\varphi_1 \land \varphi_2) \land \varphi_3) \equiv (\varphi_1 \land (\varphi_2 \land \varphi_3))
$$

$$
\neg\neg\varphi \equiv \varphi
$$

$$
((\varphi_1 \land \varphi_2) \lor \varphi_3) \equiv ((\varphi_1 \lor \varphi_3) \land (\varphi_2 \lor \varphi_3))
$$

$$
\neg(\varphi_1 \land \varphi_2) \equiv (\neg\varphi_1 \lor \neg\varphi_2)
$$

$$
(\varphi_1 \lor \varphi_1) \equiv \varphi_1
$$
Satisfiability and Validity

Definition

- A Boolean expression $\varphi$ is satisfiable iff there is a truth assignment $T$ appropriate to it with $T \models \varphi$.
- A Boolean expression $\varphi$ 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: $\varphi$ is valid $\iff \neg \varphi$ is unsatisfiable.
- Moreover, for any Boolean expressions $\psi_1$ and $\psi_2$,
  \[ \psi_1 \equiv \psi_2 \iff \models \psi_1 \leftrightarrow \psi_2 \iff \neg(\psi_1 \leftrightarrow \psi_2) \text{ is unsatisfiable.} \]
Decision Problems

SAT

INSTANCE: Boolean formula $\varphi$.
QUESTION: Is $\varphi$ satisfiable?

VALIDITY

INSTANCE: Boolean formula $\varphi$.
QUESTION: Is $\varphi$ 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 $\varphi$.
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 $x \in L \iff R(x)$ 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:

- $\text{symbol}_\sigma[\tau, \pi]$ for $0 \leq \tau \leq N$, $0 \leq \pi \leq N$ and $\sigma \in \Sigma$.
  **Intuitive meaning**: at instant $\tau$ of the computation, cell number $\pi$ contains symbol $\sigma$.

- $\text{cursor}[\tau, \pi]$ for $0 \leq \tau \leq N$ and $0 \leq \pi \leq N$.
  **Intuitive meaning**: at instant $\tau$, the cursor points to cell number $\pi$.

- $\text{state}_s[\tau]$ for $0 \leq \tau \leq N$ and $s \in K$.
  **Intuitive meaning**: at instant $\tau$, 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:

- \( symbol_{\succ} [0, 0] \)
- \( symbol_{\sigma} [0, \pi] \) 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 \((s, \sigma)\) of state \(s\) and symbol \(\sigma\) 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 \(\Delta\) (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 \(\tau\) and \(\pi\) such that \(0 \leq \tau < N\) and \(0 \leq \pi < N\).
Proof sketch (continued)

\[\text{state}_s[\tau] \land \text{symbol}_\sigma[\tau, \pi] \land \text{cursor}[\tau, \pi] \rightarrow \\
\left( (\text{state}_{s_i}'[\tau + 1] \land \text{symbol}_{\sigma_i}'[\tau + 1, \pi] \land \text{cursor}[\tau + 1, \pi + d_1]) \lor \cdots \lor \\
(\text{state}_{s_k}'[\tau + 1] \land \text{symbol}_{\sigma_k}'[\tau + 1, \pi] \land \text{cursor}[\tau + 1, \pi + d_k]) \right) \]

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 \( \tau \) and \( \pi \) such that \( 0 \leq \tau \leq N \), \( 0 \leq \pi \leq N \), \( 0 \leq i \leq m \), and \( 1 \leq j \leq n \).

\[\text{state}_{s_i}[\tau] \leftrightarrow (\neg \text{state}_{s_0}[\tau] \land \cdots \land \neg \text{state}_{s_{i-1}}[\tau] \land \\
\neg \text{state}_{s_{i+1}}[\tau] \land \cdots \land \neg \text{state}_{s_m}[\tau]) \]

\[\text{cursor}[\tau, \pi] \leftrightarrow (\neg \text{cursor}[\tau, 0] \land \cdots \land \neg \text{cursor}[\tau, \pi - 1] \land \\
\neg \text{cursor}[\tau, \pi + 1] \land \cdots \land \neg \text{cursor}[\tau, N]) \]

\[\text{symbol}_{\sigma_j}[\tau, \pi] \leftrightarrow (\neg \text{symbol}_{\sigma_1}[\tau, \pi] \land \cdots \land \neg \text{symbol}_{\sigma_{j-1}}[\tau, \pi] \land \\
\neg \text{symbol}_{\sigma_{j+1}}[\tau, \pi] \land \cdots \land \neg \text{symbol}_{\sigma_n}[\tau, \pi]) \]
Proof sketch (continued)

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

\[
\text{symbol}_\sigma[\tau, \pi] \land \text{cursor}[\tau, \pi'] \rightarrow \text{symbol}_\sigma[\tau + 1, \pi]
\]

\[
\text{symbol}_\sigma[\tau, \pi'] \land \text{cursor}[\tau, \pi] \rightarrow \text{symbol}_\sigma[\tau + 1, \pi']
\]

5. Acceptance. Let $s_m =$ “yes”.

Then $R(x)$ contains the following atom as a conjunct:

\[
\text{state}_{s_m}[N].
\]

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:
\( \varphi \) is valid \( \iff \neg \varphi \) is unsatisfiable and
\( \varphi \) is unsatisfiable \( \iff \neg \varphi \) 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 
  \[(l_{11} \lor \cdots \lor l_{1n_1}) \land \cdots \land (l_{m1} \lor \cdots \lor l_{mn_m})\]
- A formula is in **DNF** (= Disjunctive Normal Form) if it is of the form 
  \[(l_{11} \land \cdots \land l_{1n_1}) \lor \cdots \lor (l_{m1} \land \cdots \land l_{mn_m})\]

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

Definition

- A disjunction \(l_1 \lor \cdots \lor l_n\) of literals is called a **clause**.
- A conjunction \(l_1 \land \cdots \land l_n\) 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.** \((\neg x_1 \lor \neg x_1 \lor x_2) \equiv (\neg x_1 \lor x_2)\).
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 $\rightarrow$:
  - $\alpha \leftrightarrow \beta \rightsquigarrow (\neg\alpha \lor \beta) \land (\neg\beta \lor \alpha) \quad (1)$
  - $\alpha \rightarrow \beta \rightsquigarrow \neg\alpha \lor \beta \quad (2)$

- Push negations in front of Boolean variables:
  - $\neg\neg\alpha \rightsquigarrow \alpha \quad (3)$
  - $\neg(\alpha \lor \beta) \rightsquigarrow \neg\alpha \land \neg\beta \quad (4)$
  - $\neg(\alpha \land \beta) \rightsquigarrow \neg\alpha \lor \neg\beta \quad (5)$
CNF/DNF transformation

Proof sketch (continued)

The next phase depends on the normal form being pursued:

- For a CNF, move $\land$ connectives outside $\lor$ connectives:
  \[
  \alpha \lor (\beta \land \gamma) \quad \leadsto \quad (\alpha \lor \beta) \land (\alpha \lor \gamma) \quad (6)
  
  (\alpha \land \beta) \lor \gamma \quad \leadsto \quad (\alpha \lor \gamma) \land (\beta \lor \gamma) \quad (7)
  \]

- For a DNF, move $\lor$ connectives outside $\land$ connectives:
  \[
  \alpha \land (\beta \lor \gamma) \quad \leadsto \quad (\alpha \land \beta) \lor (\alpha \land \gamma) \quad (8)
  
  (\alpha \lor \beta) \land \gamma \quad \leadsto \quad (\alpha \land \gamma) \lor (\beta \land \gamma) \quad (9)
  \]

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

**Example.** Consider deriving a CNF for $(x_1 \land y_1) \lor \ldots \lor (x_n \land y_n)$. 
Example

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

\[(x_1 \lor x_2) \rightarrow (x_2 \leftrightarrow x_3) \quad \leadsto (2)\]
\[-(x_1 \lor x_2) \lor (x_2 \leftrightarrow x_3) \quad \leadsto (1)\]
\[-(x_1 \lor x_2) \lor ((\neg x_2 \lor x_3) \land (\neg x_3 \lor x_2)) \quad \leadsto (4)\]
\[(-x_1 \land \neg x_2) \lor ((\neg x_2 \lor x_3) \land (\neg x_3 \lor x_2)) \quad \leadsto (7)\]
\[(-x_1 \lor ((\neg x_2 \lor x_3) \land (\neg x_3 \lor x_2))) \land (-x_2 \lor ((\neg x_2 \lor x_3) \land (\neg x_3 \lor x_2))) \quad \leadsto (6)\]
\[((\neg x_1 \lor (\neg x_2 \lor x_3)) \land (\neg x_1 \lor (\neg x_3 \lor x_2)) \land (-x_2 \lor ((\neg x_2 \lor x_3) \land (\neg x_3 \lor x_2)))\]  
\[\leadsto (6)\]
\[((\neg x_1 \lor (\neg x_2 \lor x_3)) \land (\neg x_1 \lor (\neg x_3 \lor x_2)) \land (-x_2 \lor ((\neg x_2 \lor x_3) \land (\neg x_3 \lor x_2)))\]  
\[\leadsto (simplification)\]
\[(-x_1 \lor \neg x_2 \lor x_3) \land (-x_1 \lor \neg x_3 \lor x_2) \land (-x_2 \lor \neg x_2 \lor x_3) \land (-x_2 \lor \neg x_3 \lor x_2)\]
SAT and CNF

**Theorem**

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

  $$(A \land B) \lor C \rightsquigarrow (z \lor A) \land (z \lor B) \land (\neg z \lor C)$$

  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 $\Phi = (A \land B) \lor C$ is satisfiable, i.e., there exists a satisfying truth assignment $I$. We show that $I$ can be extended to a satisfying truth assignment $J$ of $\Psi = (z \lor A) \land (z \lor B) \land (\neg z \lor C)$.
  - Case 1: Suppose that $(A \land B)$ is true in $I$. We set $J(z) = \text{false}$. Clearly, all conjuncts of $\Psi$ are true in $J$.
  - Case 2: Suppose that $C$ is true in $I$. Then we set $J(z) = \text{true}$. Again, all conjuncts of $\Psi$ are true in $J$.

- Suppose that $\Psi = (z \lor A) \land (z \lor B) \land (\neg z \lor C)$ is satisfiable, i.e., there exists a satisfying truth assignment $J$ of $\Psi$. We claim that then $J$ is also a satisfying truth assignment of $\Phi = (A \land B) \lor C$.
  - Case 1: Suppose that $J(z) = \text{true}$. By the conjunct $(\neg z \lor C)$ in $\Psi$, $C$ (and thus $\Phi$) is true in $J$.
  - Case 2: Suppose that $J(z) = \text{false}$. By the conjuncts $(z \lor A) \land (z \lor B)$ in $\Psi$, both $A$ and $B$ (and thus $\Phi$) are true in $J$. 

Reinhard Pichler
28 March, 2017
SAT and CNF

Theorem

*There exists a log-space reduction which reduces any Boolean expression \( \varphi \) into a *sat-equivalent* Boolean expression \( \psi \) in 3-CNF, i.e.: \( \psi \) is in CNF and every clause of \( \psi \) consists of exactly 3 literals.*

Proof sketch

By the above theorem, we may assume that \( \varphi \) is in CNF.

**Case 1.** If a clause is “too small”:
A clause of the form \( c = l_1 \lor l_2 \) may be replaced by the two clauses \( c_1 = z \lor l_1 \lor l_2 \) and \( c_2 = \neg z \lor l_1 \lor l_2 \)
Likewise, a clause \( c = l_1 \) is replaced by 4 clauses \( c_1 = x \lor y \lor l_1 \), \( c_2 = x \lor \neg y \lor l_1 \), \( c_3 = \neg x \lor y \lor l_1 \) and \( c_4 = \neg x \lor \neg y \lor l_1 \).

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 $c = l_1 \lor l_2 \lor l_3 \lor R$ may be replaced by the two clauses $c_1 = z \lor l_3 \lor R$ and $c_2 = \neg z \lor l_1 \lor l_2$

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

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

If $l_3$ or $R$ is true in $I$, then we extend $I$ to $J$ with $J(z) = \text{false}$. Again, both $c_1$ and $c_2$ are true in $J$. 
## Special cases of SAT

### 3-SAT

**INSTANCE:** Boolean formula $\varphi$ in 3-CNF.

**QUESTION:** Is $\varphi$ satisfiable?

### 2-SAT

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

**QUESTION:** Is $\varphi$ satisfiable?

### HORNSAT

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

**QUESTION:** Is $\varphi$ 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 \( \varphi \) can be reduced into a sat-equivalent Boolean expression \( \psi \) in 3-CNF (i.e.: \( \psi \) is in CNF and every clause of \( \psi \) 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).
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:** $c = \neg q_1 \lor \cdots \lor \neg q_n \lor p$ with $n \geq 1$, which is usually denoted as $q_1 \land \cdots \land q_n \rightarrow p$
- **goals:** $c = \neg q_1 \lor \cdots \lor \neg q_n$
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 $\varphi$.
- If there exists a goal $\neg q_1 \lor \cdots \lor \neg q_n$ in $\varphi$, s.t. every $q_i$ is in $Y$, then $\varphi$ is unsatisfiable.
- Otherwise $\varphi$ is satisfiable. Indeed, we get a model of $\varphi$ by setting all propositional variables in $Y$ to \textbf{true} and all other variables to \textbf{false}.

Computation of the variables $Y$ that are logically implied by $\varphi$.

- Initially, let $Y := \text{set of facts in the formula } \varphi$.
- Iteratively apply the \textquotedblleft immediate consequence operator\textquotedblright{} to $Y$, i.e.: If there exists a rule $q_1 \land \cdots \land q_n \rightarrow p$, s.t. $\{q_1, \ldots, q_n\} \subseteq Y$ but $p \not\in Y$ 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 $R(x)$:
   $\text{symbol}_{\triangleright}[0,0], \text{symbol}_{\sigma}[0,\pi], \text{symbol}_{\sqcup}[0,\pi], \text{cursor}[0,0], \text{state}_{s_0}[0]$

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

   $\text{state}_s[\tau] \land \text{symbol}_\sigma[\tau,\pi] \land \text{cursor}[\tau,\pi] \rightarrow \text{state}_s'[\tau + 1]$
   
   $\text{state}_s[\tau] \land \text{symbol}_\sigma[\tau,\pi] \land \text{cursor}[\tau,\pi] \rightarrow \text{symbol}_{\sigma'}[\tau + 1,\pi]$
   
   $\text{state}_s[\tau] \land \text{symbol}_\sigma[\tau,\pi] \land \text{cursor}[\tau,\pi] \rightarrow \text{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 $\tau$, 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:

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

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

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

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

$state_{s_m}[N]$, where $s_m = \text{“yes”}$. 
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 \( \varphi \) of 2-SAT, we define the graph \( G(\varphi) \) as follows:

- The variables of \( \varphi \) and their negations form the vertices of \( G(\varphi) \).
- There is an arc \((\alpha, \beta)\) iff there is a clause \( \overline{\alpha} \lor \beta \) or \( \beta \lor \overline{\alpha} \) in \( \varphi \), where \( \overline{\alpha} \) is the complement of \( \alpha \), i.e.: If \( \alpha \) is true in some satisfying assignment \( \mathcal{I} \) of \( \varphi \), then \( \beta \) must also be true in \( \mathcal{I} \).
- It can be shown that \( \varphi \) 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 \( \varphi \). Guess a variable \( x \) and check that \( x \) is reachable from \( \neg x \) and \( \neg x \) is reachable from \( x \) in \( G(\varphi) \).
Example

\[ \psi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2) \land (x_2 \lor x_3) \]
Example

$$\psi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2) \land (x_2 \lor x_3)$$

- $\neg x_1 \Rightarrow x_2$
- $\neg x_2 \Rightarrow x_1$
- $\neg x_3$
- $x_1$
- $x_3$
- $\neg x_2$
- $x_2$

Reinhard Pichler
28 March, 2017
Example

\[ \psi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2) \land (x_2 \lor x_3) \]

\[ \neg x_1 \Rightarrow \neg x_3 \]

\[ x_3 \Rightarrow x_1 \]
Example

\[ \psi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2) \land (x_2 \lor x_3) \]

\[
\begin{aligned}
&x_1 \implies x_2 \\
&\neg x_2 \implies \neg x_1
\end{aligned}
\]
Example

\[ \psi = (x_1 \lor x_2) \land (x_1 \lor \neg x_3) \land (\neg x_1 \lor x_2) \land (x_2 \lor x_3) \]

\[ \neg x_2 \Rightarrow x_3 \]
\[ \neg x_3 \Rightarrow x_2 \]
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 \(\varphi\) is \(V\).
- For every edge \((a, b)\) in \(G\), the formula \(\varphi\) contains the clause \(\neg a \lor b\) (or, equivalently \(a \rightarrow b\)).
- Finally, the formula \(\varphi\) 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 $\psi$ denote the subformula of $\varphi$, s.t. $\psi$ consists of the rules only, i.e., $\varphi = \psi \land s \land \neg t$. Then, for every $v \in V$, the following equivalence holds: $v$ is reachable from $s \iff (\psi \land s) \models v$ (i.e.: every model of $\psi \land s$ is a model of $v$.)

Proof sketch of the hardness (continued)

It remains to show: $t$ is reachable from $s$ in $G \iff \varphi$ is unsatisfiable

“$\Rightarrow$” Suppose that $t$ is reachable from $s$. Then (by the above lemma), every model of $(\psi \land s)$ is a model of $t$. Hence, there exists no interpretation in which both $(\psi \land s)$ and $\neg t$ are true.

“$\Leftarrow$” Suppose that $t$ is not reachable from $s$. Then (by the lemma), $t$ is not implied by $(\psi \land s)$. Hence, there exists an interpretation $I$ in which $(\psi \land s)$ is true while $t$ is false, i.e., $I$ is a model of $\varphi = (\psi \land s) \land \neg t$, i.e.: $(\psi \land s) \not\models v$. 
Remarks

- By the Immerman-Szelepescénei 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 $\Pi$ of rules (a prop. logic program)
OUTPUT: Compute the least fixed-point (denoted as $X^+$) of the immediate consequence operator w.r.t. rules $\Pi$ applied to $X$.

Remarks

- $X^+$ contains all variables derivable by $\Pi$ 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 $\Pi$ 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:

\[
Y := X; \\
\textbf{while } \exists r \in \Pi, \text{ s.t. } body(r) \subseteq Y \text{ and } head(r) \not\in Y \textbf{ do} \\
\quad Y := Y \cup \{head(r)\}; \\
\textbf{endwhile}; \\
\textbf{return } Y;
\]
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 $\text{body}(r) \subseteq Y$ 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 $\Pi$ over $V$, variable set $X \subseteq V$
Output: Least fixed point $X^+$ w.r.t. immediate consequence 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 $X^+$

\[
Y := X; \\
\text{while } Y \text{ contains an unmarked element } z \text{ do} \\
\hspace{1em} \text{mark } z; \\
\hspace{1em} \text{for each } r \in L(z) \text{ do} \\
\hspace{2em} \text{count}(r) := \text{count}(r) - 1; \\
\hspace{2em} \text{if count}(r) = 0 \text{ then } Y := Y \cup \{\text{head}(r)\}; \\
\hspace{1em} \text{end for}; \\
\text{end while}; \\
\text{return } Y; 
\]
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 $\text{count}(r) = |\{z \in \text{body}(r) \mid z \text{ is not marked}\}|$.

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

- Initialization: takes time $O(|V| + ||\Pi||)$
- 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 $O(||\Pi||)$ is needed.
Decision Procedure for HORNSAT

HORNSAT

INSTANCE: Boolean formula $\varphi$ 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 $\varphi$ satisfiable?

Decision Procedure for HORNSAT

Related closure computation problem:

$X = \{ p \mid p \text{ is a fact in } \varphi \}$.

$\Pi = \{ q_1 \land \cdots \land q_k \rightarrow p \mid \neg q_1 \lor \cdots \lor \neg q_k \lor p \text{ is a rule in } \varphi \} \cup \{ q_1 \land \cdots \land q_k \rightarrow \bot \mid \neg q_1 \lor \cdots \lor \neg q_k \text{ is a goal in } \varphi \}$.

Criterion for the Satisfiability of $\varphi$:

$\varphi$ 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 HORNSAT problem:

- $V$ as above, $\Pi = \{A \rightarrow B \mid (A, B) \in E\}$, $X = \{s\}$.
- 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 HORNSAT problem:

- $V = R$, $\Pi = \{A_1 \land \cdots \land A_k \rightarrow B \mid (A_1 \ldots A_k \rightarrow B) \in F\}$.
- 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.