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: $\lor$, $\land$, 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: \(((x_1 \lor \neg x_2) \lor x_3) \lor (x_4 \lor (x_5 \lor x_6))\) is written as \(x_1 \lor \neg x_3 \lor x_2 \lor x_4 \lor x_2 \lor x_5 \lor x_6\).

- Disjunctions and conjunctions involving \(n\) members:
  - \(\lor_{i=1}^{n} \phi_i\) stands for \(\phi_1 \lor \cdots \lor \phi_n\).
  - \(\land_{i=1}^{n} \phi_i\) stands for \(\phi_1 \land \cdots \land \phi_n\).

- Frequently appearing abbreviations:
  - An implication \(\phi_1 \rightarrow \phi_2\) stands for \(\neg \phi_1 \lor \phi_2\).
  - An equivalence \(\phi_1 \leftrightarrow \phi_2\) stands for \((\neg \phi_1 \lor \phi_2) \land (\neg \phi_2 \lor \phi_1)\).

- The dual (or complement) of a literal \(\alpha\) is denoted by \(\neg \alpha\), i.e., let \(\alpha \in X\). Then \(\neg \alpha\) stands for \(\neg \alpha\) and \(\neg \neg \alpha\) stands for \(\alpha\).


Satisfaction relation

**Definition**
Let a truth assignment \(T : X' \rightarrow \{\text{true}, \text{false}\}\) be appropriate to \(\varphi\), i.e., \(X(\varphi) \subseteq X'\). \(T \models \varphi\) (or \(T\) 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 \(\varphi = \neg \phi_1\), then \(T \models \varphi\) iff \(T \not\models \phi_1\).
- If \(\varphi = \phi_1 \land \phi_2\), then \(T \models \varphi\) iff \(T \models \phi_1\) and \(T \models \phi_2\).
- If \(\varphi = \phi_1 \lor \phi_2\), then \(T \models \varphi\) iff \(T \models \phi_1\) or \(T \models \phi_2\).

**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,

\(T \models \varphi_1\) iff \(T \models \varphi_2\).

**Example**
\((\varphi_1 \lor \varphi_2) \equiv (\varphi_2 \lor \varphi_1)\)
\(((\varphi_1 \land \varphi_2) \land \varphi_3) \equiv (\varphi_1 \land (\varphi_2 \land \varphi_3))\)
\((\neg \varphi) \equiv \varphi\)
\(((\varphi_1 \lor \varphi_2) \lor \varphi_3) \equiv ((\varphi_1 \lor \varphi_3) \lor (\varphi_2 \lor \varphi_3))\)
\((\neg (\varphi_1 \land \varphi_2)) \equiv (\neg \varphi_1 \lor \neg \varphi_2)\)
\((\varphi_1 \lor \varphi_1) \equiv \varphi_1\)
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 \( \psi_2 \),
\( \psi_1 \equiv \psi_2 \iff \models \psi_1 \iff \neg (\psi_1 \iff \psi_2) \) is unsatisfiable.

Decision Problems

SAT
INSTANCE: Boolean formula \( \varphi \).
QUESTION: Is \( \varphi \) satisfiable?

VALIDITY
INSTANCE: Boolean formula \( \varphi \).
QUESTION: Is \( \varphi \) valid?

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 \( \sigma \in \Sigma \).
- \( \text{cursor}[\tau, \pi] \) for \( 0 \leq \tau \leq N \) and \( 0 \leq \pi \leq N \).
- \( \text{state}_s[\tau] \) for \( 0 \leq \tau \leq N \) and \( s \in K \).

Intuitive meaning:
- At instant \( \tau \) of the computation, cell number \( \pi \) contains symbol \( \sigma \).
- At instant \( \tau \), the cursor points to cell number \( \pi \).
- 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:
   
   $\text{symbol}_0[0, 0]$
   $\text{symbol}_0[0, \pi]$ for $1 \leq \pi \leq \vert x \vert$, where $x_\pi = \sigma$
   $\text{symbol}_\pi[0, \pi]$ for $\vert x \vert < \pi \leq N$
   $\text{cursor}[0, 0]$
   $\text{state}_0[0]$

2. Transition rules. For each pair $(s, \sigma)$ of state $s$ and symbol $\sigma$ let $(s, \sigma, s'_1, d_1), \ldots, (s, \sigma, s'_k, d_k)$ denote all possible transitions according to the transition relation $\Delta$ (for the cursor movements, we write $d_i \in \{-1, 0, 1\}$ rather than $d_i \in \{\leftarrow, \rightarrow, \downarrow\}$).
   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)

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 $\sigma \in \Sigma$,

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

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

\[
\text{state}_m[N]
\]

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 $x$) if and only if $R(x)$ 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
  \((l_1 \lor \cdots \lor l_n) \land \cdots \land (l_m \lor \cdots \lor l_{mn})\)
- A formula is in DNF (= Disjunctive Normal Form) if it is of the form
  \((l_1 \land \cdots \land l_n) \lor \cdots \lor (l_m \land \cdots \land l_{mn})\)

where each \(l_i\) 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

Proof sketch (continued)

The next phase depends on the normal form being pursued:

- For a CNF, move \(\land\) connectives outside \(\lor\) connectives:
  \[\alpha \lor (\beta \land \gamma) \rightarrow (\alpha \lor \beta) \land (\alpha \lor \gamma)\]
  \[\alpha \land (\beta \lor \gamma) \rightarrow (\alpha \land \beta) \lor (\alpha \land \gamma)\]

- For a DNF, move \(\lor\) connectives outside \(\land\) connectives:
  \[\alpha \land (\beta \lor \gamma) \rightarrow (\alpha \lor \beta) \lor (\alpha \lor \gamma)\]

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

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

Example

Transform \((x_1 \lor x_2) \rightarrow (x_2 \leftrightarrow x_3)\) into CNF.
\[(x_1 \lor x_2) \rightarrow (x_2 \leftrightarrow x_3) \rightarrow (x_1 \lor x_2) \rightarrow (x_2 \leftrightarrow x_3) \rightarrow (x_1 \lor x_2) \lor (x_2 \land \neg x_3) \lor (\neg x_1 \land x_3)\]

\[(\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \rightarrow (x_1 \lor x_2) \lor (x_2 \land \neg x_3) \lor (\neg x_1 \land x_3) \rightarrow (x_1 \lor x_2) \lor (x_2 \land \neg x_3) \lor (\neg x_1 \land x_3) \lor (\neg x_2 \lor x_3)\]

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

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

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

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

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

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

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

\[(\neg x_2 \lor x_3) \lor (\neg x_1 \land x_3) \lor (\neg x_2 \lor x_3) \lor (\neg x_2 \lor x_3) \lor (\neg x_2 \lor x_3) \lor (\neg x_2 \lor x_3) \lor (\neg x_2 \lor x_3) \lor (\neg x_2 \lor x_3) \lor (\neg x_2 \lor x_3)\]
SAT and CNF

Theorem
There exists a log-space reduction which reduces any Boolean expression \( \varphi \) into a \textit{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:
\[
(A \wedge B) \vee C \rightsquigarrow (z \vee A) \wedge (z \vee B) \wedge (\neg z \vee C)
\]
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. Then at least one of the disjuncts in \( \Phi \) is true in \( J \). Thus both \( C \) and \( \Phi \) are true in \( J \).

- Suppose that \( \Psi = (z \vee A) \wedge (z \vee B) \wedge (\neg z \vee C) \) is satisfiable. Then at least one of the conjuncts in \( \Psi \) is true in \( J \). Thus both \( A \) and \( B \) (and thus \( \Phi \)) are true in \( J \).

SAT and CNF

Theorem
There exists a log-space reduction which reduces any Boolean expression \( \varphi \) into a \textit{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 \( c = h_1 \vee h_2 \) may be replaced by the two clauses \( c_1 = z \vee h_1 \vee h_2 \) and \( c_2 = \neg z \vee h_1 \vee h_2 \).
Likewise, a clause \( c = h_1 \) is replaced by 4 clauses \( c_1 = x \vee y \vee h_1 \), \( c_2 = x \vee \neg y \vee h_1 \), \( c_3 = \neg x \vee y \vee h_1 \) and \( c_4 = \neg x \vee \neg y \vee h_1 \).

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

Case 2. If a clause is “too big”:
Any clause of the form \( c = h_1 \vee h_2 \vee h_3 \vee R \) may be replaced by the two clauses \( c_1 = z \vee h_3 \vee R \) and \( c_2 = z \vee h_1 \vee h_2 \).

It remains to show that the result of this step is a sat-equivalent formula. We only show one direction: Suppose that \( c = h_1 \vee h_2 \vee h_3 \vee R \) has a satisfying assignment \( I \). Then at least one of the disjuncts in \( c \) is true.

If \( h_1 \) or \( h_2 \) is true in \( I \), then we extend \( I \) to \( J \) with \( J(z) = false \).

If \( h_3 \) or \( R \) is true in \( I \), then we extend \( I \) to \( J \) with \( J(z) = true \).

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).

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 true and all other variables to 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 “immediate consequence operator” 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 $R(x)$:
   
   \[ \text{symbol}_0[0, 0], \text{symbol}_0[0, \pi], \text{symbol}_1[0, \pi], \text{cursor}[0, 0], \text{state}_0[0] \]

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 $(s, \sigma, s', \sigma', d)$ according to the transition function $\delta$. Conjuncts in $R(x)$:
   
   \[
   \begin{align*}
   \text{state}_0[r] \land \text{symbol}_0[r, \pi] \land \text{cursor}[r, \pi] &\implies \text{state}_1[r + 1] \\
   \text{state}_0[r] \land \text{symbol}_0[r, \pi] \land \text{cursor}[r, \pi] &\implies \text{symbol}_1[r + 1, \pi] \\
   \text{state}_0[r] \land \text{symbol}_0[r, \pi] \land \text{cursor}[r, \pi] &\implies \text{cursor}[r + 1, \pi + d]
   \end{align*}
   \]

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 $\alpha \vee \beta$ or $\beta \vee \alpha$ in $\varphi$, where $\alpha$ is the complement of $\alpha$, i.e.: If $\alpha$ is true in some satisfying assignment $I$ of $\varphi$, then $\beta$ must also be true in $I$.
- 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

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

\[ \neg x_1 \Rightarrow x_2 \\
\neg x_2 \Rightarrow \neg x_1 \\
x_1 \Rightarrow x_3 \\
\neg x_2 \Rightarrow \neg x_1 \\
x_3 \Rightarrow x_2 \\
\neg x_3 \Rightarrow x_2 \]

Example

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

\[ \neg x_2 \Rightarrow x_3 \\
\neg x_3 \Rightarrow x_2 \\
x_1 \Rightarrow x_3 \\
\neg x_2 \Rightarrow \neg x_1 \\
x_3 \Rightarrow x_2 \\
\neg x_3 \Rightarrow x_2 \]
Proof sketch of the hardness

We reduce \textsc{Reachability} to the \textsc{co-2-SAT}-problem. Let \((G, s, t)\) be an arbitrary instance of \textsc{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 \textsc{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 \textsc{Reachability} to \textsc{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, i.e.: \(\varphi\) is unsatisfiable.

\(\Leftarrow\) 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.: \(\varphi\) is satisfiable.

Efficient Solution of \textsc{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:

$Y := X$;
while $\exists r \in \Pi$, s.t. $\text{body}(r) \subseteq Y$ and $\text{head}(r) \not\in Y$ do
  $Y := Y \cup \{\text{head}(r)\}$;
endwhile;
return $Y$;

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 $L(z) := \text{empty-list}$;
for each $r \in \Pi$ do
  count($r$) := $|\text{body}(r)|$;
  for each $z \in \text{body}(r)$ do add $r$ to the list $L(z)$;
end for;

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 (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 $\bot \not\in X^+$ holds.

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.