



Constraint Solving

René Thiemann and Fabian Mitterwallner
based on a previous course by Aart Middeldorp

Important Information

- LVA 703304 (VO 2) + 703305 (PS 2)
- <http://cl-informatik.uibk.ac.at/teaching/ss24/cs>
- Please register for both VO and PS
- OLAT link for VO (contains recordings)

Time and Place

VO	Tuesday	8:15–10:00	SR 13	RT
PS	Friday	8:30–10:00	3W04	FM

Outline

- 1. Introduction**
Organisation
- 2. Propositional Logic – Review**
- 3. Tseitin's Transformation**
- 4. DPLL**
- 5. Further Reading**

Consultation Hours

René Thiemann	3M09	Tuesday 10:15–11:15
Fabian Mitterwallner	3M03	Thursday 10:30–12:00

Schedule – one deviation: PS in week 11 on June 3, 17:15–18:45

week 1	05.03 & 08.03	week 6	23.04 & 26.04	week 11	28.05 & 03.06 at 17:15
week 2	12.03 & 15.03	week 7	30.04 & 03.05	week 12	04.06 & 07.06
week 3	19.03 & 22.03	week 8	07.05 & 10.05	week 13	11.06 & 14.06
week 4	09.04 & 12.04	week 9	14.05 & 17.05	week 14	18.06 & 21.06 Q & A
week 5	16.04 & 19.04	week 10	21.05 & 24.05	week 15	25.06 first exam

Grading — Vorlesung

- first exam on June 25
- registration starts 5 weeks before exam and ends 1 week before exam
- de-registration is possible until 23:59 on June 23
- second and third exams in September and February (on demand)

Grading — Proseminar

- solved exercises must be marked in OLAT before 8 am on Friday
- 10 points per PS
- attendance is compulsory; unexcused absence is allowed twice (email solutions to get some points in such cases)
- grading: 80 % for marked exercises, 20 % for presentation of solutions

Topics

- [propositional logic \(SAT\)](#)
cardinality constraints, [DPLL](#), maxSAT, NP-completeness, unsatisfiable cores
- [satisfiability modulo theories \(SMT\)](#)
DPLL(T), Nelson–Oppen combination method
- [equality logic and uninterpreted functions \(EUF\)](#)
congruence closure, graph-based reduction
- [linear arithmetic \(LIA and LRA\)](#)
branch and bound, cubes and equalities, difference logic, simplex algorithm, small model property
- [bit vectors \(BV\) and floating points](#)
bit-vector arithmetic, bounded model checking, fixed-point arithmetic, flattening
- [arrays and pointers](#)
array properties, lazy encoding, pointer logic
- [quantified formulas](#)
Cooper’s method, Ferrante–Rackoff’s method, PSPACE-completeness, quantified boolean formulas (QBF)

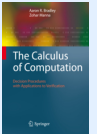
Literature

- Daniel Kroning and Ofer Strichman
Decision Procedures – An Algorithmic Point of View (Second Edition)
Springer, 2016 [online version via Ebook Central](#)



<http://www.decision-procedures.org>

- Aaron Bradley and Zohar Manna
The Calculus of Computation – Decision Procedures with Applications to Verification
Springer, 2007 [online version via Ebook Central](#)



<http://theory.stanford.edu/~arbrad/book.html>

Online Material

slides and additional reading material are available from uibk.ac.at domain

Outline

1. Introduction
2. Propositional Logic – Review
3. Tseitin’s Transformation
4. DPLL
5. Further Reading

Concepts of Propositional Logic

- formula
- assignment
- satisfiability
- validity
- negation normal form (NNF)
- conjunctive normal form (CNF)
- disjunctive normal form (DNF)
- literal

Definition (Propositional Logic: Syntax)

- propositional **formulas** are built from

- **atoms** p, q, r, p_1, p_2, \dots
- **top, bottom** \top, \perp
- **negation** \neg $\neg p$ "not p "
- **conjunction** \wedge $p \wedge q$ " p and q "
- **disjunction** \vee $p \vee q$ " p or q "
- **implication** \rightarrow $p \rightarrow q$ "if p then q "
- **equivalence** \leftrightarrow $p \leftrightarrow q$ " p if and only if q "

according to **BNF grammar** $\varphi ::= p \mid \perp \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$

- notational conventions:

- **binding precedence** $\neg > \wedge > \vee > \rightarrow, \leftrightarrow$ omit outer parentheses
- $\rightarrow, \wedge, \vee$ are **right-associative**: $p \rightarrow q \rightarrow r$ denotes $p \rightarrow (q \rightarrow r)$

Definition (Propositional Logic: Semantics)

- **valuation** (truth **assignment**) is mapping $v: \{p \mid p \text{ is atom}\} \rightarrow \{T, F\}$
- extension to formulas: **truth values**

- $v(\top) = T$
- $v(\perp) = F$
- $v(\neg \varphi) = \begin{cases} T & \text{if } v(\varphi) = F \\ F & \text{otherwise} \end{cases}$
- $v(\varphi \wedge \psi) = \begin{cases} T & \text{if } v(\varphi) = v(\psi) = T \\ F & \text{otherwise} \end{cases}$
- $v(\varphi \vee \psi) = \begin{cases} F & \text{if } v(\varphi) = v(\psi) = F \\ T & \text{otherwise} \end{cases}$
- $v(\varphi \rightarrow \psi) = \begin{cases} F & \text{if } v(\varphi) = T \text{ and } v(\psi) = F \\ T & \text{otherwise} \end{cases}$
- $v(\varphi \leftrightarrow \psi) = \begin{cases} T & \text{if } v(\varphi) = v(\psi) \\ F & \text{otherwise} \end{cases}$

Definitions

- **semantic entailment**

$$\varphi_1, \varphi_2, \dots, \varphi_n \models \psi$$

- if $v(\psi) = T$ whenever $v(\varphi_1) = v(\varphi_2) = \dots = v(\varphi_n) = T$, for every valuation v
- formula φ is **valid** if $v(\varphi) = T$ for every valuation v
- formula φ is **satisfiable** if $v(\varphi) = T$ for some valuation v
- formulas φ and ψ are **equivalent** ($\varphi \equiv \psi$) if $v(\varphi) = v(\psi)$ for every valuation v
- formulas φ and ψ are **equisatisfiable** ($\varphi \approx \psi$) if

$$\varphi \text{ is satisfiable} \iff \psi \text{ is satisfiable}$$

Theorem

- formula φ is valid $\iff \neg \varphi$ is unsatisfiable
- validity and satisfiability are **decidable**

Definitions

- **negation normal form (NNF)** is formula without implication and equivalence, and with negation only applied to atoms
- **literal** is atom p or negation $\neg p$ of atom
- **clause** is disjunction of literals
- **conjunctive normal form (CNF)** is conjunction of clauses
- **disjunctive normal form (DNF)** is disjunction of conjunctions of literals

Theorem

\forall formula $\varphi \exists$ CNF $\psi \exists$ DNF χ such that $\varphi \equiv \psi \equiv \chi$

Satisfiability (SAT)

instance: (propositional) formula φ

question: is φ satisfiable?

Theorem

SAT is NP-complete, even for CNF formulas

Remark

most SAT solvers require CNF as input

DIMACS Input Format

```
c
c comments
c
p cnf 4 3          4 atoms and 3 clauses
1 -2 4 0          x1 ∨ ¬x2 ∨ x4
-1 2 -3 -4 0      ¬x1 ∨ x2 ∨ ¬x3 ∨ ¬x4
3 -2 0            x3 ∨ ¬x2
```

SAT Applications

Applications of SAT

- Encoding games
<http://cl-informatik.uibk.ac.at/software/puzzles/>
- Strategies and configurations
- Cryptanalysis
- Many graph problems
- Component of reasoning in more complex logics

Outline

1. Introduction
2. Propositional Logic – Review
- 3. Tseitin's Transformation**
4. DPLL
5. Further Reading

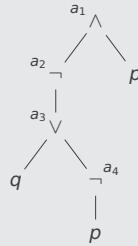
Remarks

- translation from arbitrary formula to equivalent CNF is expensive
- Tseitin's transformation** is linear-time translation to **equisatisfiable** CNF
- in lecture: only consider formulas without \rightarrow and \leftrightarrow

Example (Tseitin's Transformation)

- $\varphi = \neg(q \vee \neg p) \wedge p$
- introduce new variable for each propositional connective:

a_1	$\neg(q \vee \neg p) \wedge p$	a_3	$q \vee \neg p$
a_2	$\neg(q \vee \neg p)$	a_4	$\neg p$
- $\varphi \approx a_1 \wedge (a_1 \leftrightarrow a_2 \wedge p) \wedge (a_2 \leftrightarrow \neg a_3) \wedge (a_3 \leftrightarrow q \vee a_4) \wedge (a_4 \leftrightarrow \neg p)$



Lemma

- $(\varphi \leftrightarrow \neg\psi) \equiv (\varphi \vee \psi) \wedge (\neg\varphi \vee \neg\psi)$
- $(\varphi \leftrightarrow \psi \wedge \chi) \equiv (\neg\varphi \vee \psi) \wedge (\neg\varphi \vee \chi) \wedge (\varphi \vee \neg\psi \vee \neg\chi)$
- $(\varphi \leftrightarrow \psi \vee \chi) \equiv (\varphi \vee \neg\psi) \wedge (\varphi \vee \neg\chi) \wedge (\neg\varphi \vee \psi \vee \chi)$

Example (cont'd)

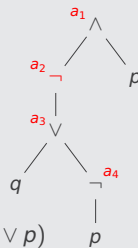
$$\begin{aligned} \varphi &\approx a_1 \wedge (a_1 \leftrightarrow a_2 \wedge p) \wedge (a_2 \leftrightarrow \neg a_3) \wedge (a_3 \leftrightarrow q \vee a_4) \wedge (a_4 \leftrightarrow \neg p) \\ &\equiv a_1 \wedge (\neg a_1 \vee a_2) \wedge (\neg a_1 \vee p) \wedge (a_1 \vee \neg a_2 \vee \neg p) \wedge (a_2 \vee a_3) \wedge (\neg a_2 \vee \neg a_3) \\ &\quad \wedge (a_3 \vee \neg q) \wedge (a_3 \vee \neg a_4) \wedge (\neg a_3 \vee q \vee a_4) \wedge (a_4 \vee p) \wedge (\neg a_4 \vee \neg p) \end{aligned}$$

Improvement (Plaisted & Greenbaum 1986)

replace equivalence (\leftrightarrow) by implication (\rightarrow or \leftarrow) based on **polarity** of subformulas

Example (cont'd)

- $\varphi = \neg(q \vee \neg p) \wedge p$
- $\varphi \approx a_1 \wedge (a_1 \rightarrow a_2 \wedge p) \wedge (a_2 \rightarrow \neg a_3) \wedge (a_3 \leftarrow q \vee a_4) \wedge (a_4 \leftarrow \neg p)$
- $a_1 \rightarrow a_2 \wedge p \equiv (\neg a_1 \vee a_2) \wedge (\neg a_1 \vee p) \wedge (a_1 \vee \neg a_2 \vee \neg p)$
- $a_2 \rightarrow \neg a_3 \equiv (a_2 \vee a_3) \wedge (\neg a_2 \vee \neg a_3)$
- $a_3 \leftarrow q \vee a_4 \equiv (a_3 \vee \neg q) \wedge (a_3 \vee \neg a_4) \wedge (\neg a_3 \vee q \vee a_4)$
- $a_4 \leftarrow \neg p \equiv (a_4 \vee p) \wedge (\neg a_4 \vee \neg p)$
- $\varphi \approx a_1 \wedge (\neg a_1 \vee a_2) \wedge (\neg a_1 \vee p) \wedge (\neg a_2 \vee \neg a_3) \wedge (a_3 \vee \neg q) \wedge (a_3 \vee \neg a_4) \wedge (a_4 \vee p)$



replace $a \leftrightarrow \psi$ by $a \rightarrow \psi$ if ψ occurs only positively, and by $a \leftarrow \psi$ if ψ never occurs positively

Definition

subformula ψ occurs **positively** in formula φ if number of negations on path from root of φ to root of ψ in parse tree of φ is even

Outline

- Introduction
- Propositional Logic - Review
- Tseitin's Transformation
- DPLL
- Further Reading

Remarks

- most state-of-the-art SAT solvers are based on variations of **Davis–Putnam–Logemann–Loveland** (DPLL) procedure (1960, 1962)
- abstract version** of DPLL described in JACM paper of Nieuwenhuis, Oliveras, Tinelli (2006)

Definition (Abstract DPLL)

- states $M \parallel F$ consist of
 - list M of (possibly annotated) non-complementary literals
 - CNF F
- transition rules

$$M \parallel F \implies M' \parallel F' \text{ or fail-state}$$

Example

$$\begin{aligned} \varphi &= (\neg 1 \vee \neg 2) \wedge (2 \vee 3) \wedge (\neg 1 \vee \neg 3 \vee 4) \wedge (2 \vee \neg 3 \vee \neg 4) \wedge (1 \vee 4) \\ \implies & \quad \quad \quad \overset{d}{1} \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{decide} \\ \implies & \quad \quad \quad \overset{d}{1} \neg 2 \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{unit propagate} \\ \implies & \quad \quad \quad \overset{d}{1} \neg 2 \ 3 \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{unit propagate} \\ \implies & \quad \quad \quad \overset{d}{1} \neg 2 \ 3 \ 4 \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{unit propagate} \\ \implies & \quad \quad \quad \neg 1 \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{backtrack} \\ \implies & \quad \quad \quad \neg 1 \ 4 \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{unit propagate} \\ \implies & \quad \quad \quad \neg 1 \ 4 \ \overset{d}{\neg 3} \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{decide} \\ \implies & \quad \quad \quad \neg 1 \ 4 \ \overset{d}{\neg 3} \ 2 \parallel \neg 1 \vee \neg 2, 2 \vee 3, \neg 1 \vee \neg 3 \vee 4, 2 \vee \neg 3 \vee \neg 4, 1 \vee 4 && \text{unit propagate} \end{aligned}$$

Definition (Transition Rules)

- unit propagate** $M \parallel F, C \vee I \implies M I \parallel F, C \vee I$
if $M \models \neg C$ and I is undefined in M **unit clause**
- pure literal** $M \parallel F \implies M I \parallel F$
if I occurs in F and I^c (complement of I) does not occur in F and I is undefined in M
- decide** $M \parallel F \implies M \overset{d}{I} \parallel F$
if I or I^c occurs in F and I is undefined in M
- fail** $M \parallel F, C \implies \text{fail-state}$
if $M \models \neg C$ and M contains no decision literals
- backtrack** $M \overset{d}{I} N \parallel F, C \implies M I^c \parallel F, C$
if $M \overset{d}{I} N \models \neg C$ and N contains no decision literals

Outline

- Introduction
- Propositional Logic – Review
- Tseitin’s Transformation
- DPLL**
Backjumping
- Further Reading

Example

$$\varphi = (\neg 1 \vee 2) \wedge (\neg 3 \vee 4) \wedge (\neg 5 \vee \neg 6) \wedge (6 \vee \neg 5 \vee \neg 2)$$

=>		$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	
=>	$\overset{d}{1}$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	decide
=>	$\overset{d}{1} \overset{d}{2}$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	unit propagate
=>	$\overset{d}{1} \overset{d}{2} \overset{d}{3}$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	decide
=>	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4}$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	unit propagate
=>	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4} \overset{d}{5}$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	decide
=>	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4} \overset{d}{5} \neg 6$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	unit propagate
=>	$\overset{d}{1} \overset{d}{2} \neg 5$	$\parallel \neg 1 \vee 2, \neg 3 \vee 4, \neg 5 \vee \neg 6, 6 \vee \neg 5 \vee \neg 2$	backjump

conflict is due to $\overset{d}{1} \overset{d}{2}$ and $\overset{d}{5} \neg 6$ hence $\neg 1 \vee \neg 5$ can be inferred

Definitions

- **backtrack** $M \overset{d}{I} N \parallel F, C \implies M I^c \parallel F, C$
if $M \overset{d}{I} N \models \neg C$ and N contains no decision literals
- **backjump** $M \overset{d}{I} N \parallel F, C \implies M I' \parallel F, C$
if $M \overset{d}{I} N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - $F, C \models C' \vee I'$ **backjump clause**
 - $M \models \neg C'$
 - I' is undefined in M
 - I' or I'^c occurs in F or in $M \overset{d}{I} N$

Example (cont'd)

$\neg 1 \vee \neg 5$ and $\neg 2 \vee \neg 5$ are backjump clauses with respect to $\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4} \overset{d}{5} \neg 6 \parallel \varphi$

Definition

basic DPLL \mathcal{B} consists of transition rules

- **unit propagate** $M \parallel F, C \vee I \implies M I \parallel F, C \vee I$
if $M \models \neg C$ and I is undefined in M
- **decide** $M \parallel F \implies M \overset{d}{I} \parallel F$
if I or I^c occurs in F and I is undefined in M
- **fail** $M \parallel F, C \implies \text{fail-state}$
if $M \models \neg C$ and M contains no decision literals
- **backjump** $M \overset{d}{I} N \parallel F, C \implies M I' \parallel F, C$
if $M \overset{d}{I} N \models \neg C$ and \exists clause $C' \vee I'$ such that
 - $F, C \models C' \vee I'$ and $M \models \neg C'$
 - I' is undefined in M and I' or I'^c occurs in F or in $M \overset{d}{I} N$

Theorem

there are no infinite derivations $\parallel F \implies_B S_1 \implies_B S_2 \implies_B \dots$

Proof

- for list of distinct literals M , $|M|$ is length of M
- measure state $M_0 \overset{d}{I_1} M_1 \overset{d}{I_2} M_2 \dots \overset{d}{I_k} M_k \parallel F$ where M_0, \dots, M_k contain no decision literals by tuple $(|M_0|, |M_1|, \dots, |M_k|)$
- compare tuples **lexicographically** using standard order on \mathbb{N}
- every transition step **strictly increases** measure
- measure is **bounded** by $(n+1)$ -tuple (n, \dots, n) where n is total number of atoms

Example

$\ \varphi = (\neg 1 \vee 2) \wedge (\neg 3 \vee 4) \wedge (\neg 5 \vee \neg 6) \wedge (6 \vee \neg 5 \vee \neg 2)$	(0)
\Rightarrow	$\overset{d}{1} \ \varphi$ decide (0, 0)
\Rightarrow	$\overset{d}{1} \overset{d}{2} \ \varphi$ unit propagate (0, 1)
\Rightarrow	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \ \varphi$ decide (0, 1, 0)
\Rightarrow	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4} \ \varphi$ unit propagate (0, 1, 1)
\Rightarrow	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4} \overset{d}{5} \ \varphi$ decide (0, 1, 1, 0)
\Rightarrow	$\overset{d}{1} \overset{d}{2} \overset{d}{3} \overset{d}{4} \overset{d}{5} \neg 6 \ \varphi$ unit propagate (0, 1, 1, 1)
\Rightarrow	$\overset{d}{1} \overset{d}{2} \neg 5 \ \varphi$ backjump (0, 2)

- **decide** $(m_0, \dots, m_i) <_{\text{lex}} (m_0, \dots, m_i, 0)$
- **unit propagate** $(m_0, \dots, m_i) <_{\text{lex}} (m_0, \dots, m_i + 1)$
- **backjump** $(m_0, \dots, m_i) <_{\text{lex}} (m_0, \dots, m_j + 1)$ with $j < i$

Lemma

- 1 if $\|\varphi \Rightarrow_B^* M \|\varphi'$ then
 - $F = F'$
 - M does not contain complementary literals
 - M consists of distinct literals
- 2 if $\|\varphi \Rightarrow_B^* M_0 \overset{d}{l_1} M_1 \overset{d}{l_2} M_2 \dots \overset{d}{l_k} M_k \|\varphi$ with no decision literals in M_0, \dots, M_k then $F, l_1, \dots, l_i \models M_i$ for all $0 \leq i \leq k$

Theorem

if $\|\varphi \Rightarrow_B S_1 \Rightarrow_B \dots \Rightarrow_B S_n \not\Rightarrow_B$ then

- 1 $S_n = \text{fail-state}$ if and only if F is unsatisfiable
- 2 $S_n = M \|\varphi'$ only if F is satisfiable and $M \models F$

Proof

- 1 (only if) $\|\varphi \Rightarrow_B^* M \|\varphi \Rightarrow_{\text{fail}}$ fail-state
 - M contains no decision literals and $M \models \neg C$ for some C in F
 - $F \models C$ and $F \models M$ and thus $F \models \neg C$ and thus F is unsatisfiable
- 2 $\|\varphi \Rightarrow_B^* M \|\varphi \not\Rightarrow_B$
 - $F = F'$ and all literals in F are defined in M , otherwise **decide** is applicable
 - F contains no clause C such that $M \models \neg C$, otherwise **backjump** or **fail** is applicable
 - $M \models F$ and thus F is satisfiable

Remarks

- finding backjump clauses is done via **conflict graphs**
- extension of DPLL: Conflict Driven Clause Learning (CDCL)
 - backjump clauses can be learned and are added to formula
 - restart rule erases current list M , but keeps learned formulas
- restarts do not compromise completeness if number of steps between consecutive restarts strictly increases
- modern SAT solvers additionally incorporate
 - heuristics for selecting next decision literal
 - special data structures that allow for efficient unit propagation

Outline

1. Introduction
2. Propositional Logic – Review
3. Tseitin’s Transformation
4. DPLL
5. Further Reading

Kröning and Strichmann

- Chapter 1
- Chapter 2

Further Reading

- David A. Plaisted and Steven Greenbaum
A Structure-Preserving Clause Form Translation
Journal of Symbolic Computation 2(3), pp. 293–304, 1986
- Martin Davis and Hilary Putnam
A Computing Procedure for Quantification Theory
Journal of the ACM 7(3), pp. 201–215, 1960
- Martin Davis, George Logemann, and Donald Loveland
A Machine Program for Theorem-Proving
Communications of the ACM 5(7), pp. 394–397, 1962

Further Reading (cont’d)

- Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli
Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T)
Journal of the ACM 53(6), pp. 937–977, 2006
- Moshe Y. Vardi
Boolean Satisfiability: Theory and Engineering
Communications of the ACM 57(3), editor’s letter, 2014

Important Concepts

- | | | |
|---------------------------|---------------------------|----------------------------|
| • abstract DPLL | • disjunction | • semantic entailment |
| • atom | • disjunctive normal form | • semantic equivalence |
| • basic DPLL | • equisatisfiability | • Tseitin’s transformation |
| • backjump | • fail-state | • tautology |
| • backtrack | • implication | • top |
| • bottom | • literal | • truth table |
| • clause | • negation | • truth values |
| • complementary literals | • polarity | • unit propagation |
| • conflict graph | • pure literal | • validity |
| • conjunction | • restart | • valuation |
| • conjunctive normal form | • right-associativity | |
| • decide | • satisfiability | |