universität innsbruck

SS 2024 lecture 1

Constraint Solving

René Thiemann and Fabian Mitterwallner

based on a previous course by Aart Middeldorp

Outline

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

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)

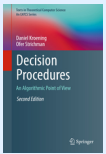
Grading — Proseminar

- 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

Literature

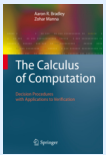
Daniel Kröning and Ofer Strichman
Decision Procedures – An Algorithmic Point of View (Second Edition)
Springer, 2016 online version via Ebook Central

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

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)

Outline

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 $p, q, r, p_1, p_2, \ldots$
  - top,bottom $\top, \bot$
  - 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 $\varphi ::= p \mid \bot \mid \top \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$
- 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 $v\colon \{p \mid p \text{ is atom}\} \to \{\mathsf{T}, \mathsf{F}\}$
  truth values
- extension to formulas:
  - $v(\top) = \mathsf{T}$
  - $v(\bot) = \mathsf{F}$
  - $v(\neg\varphi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = \mathsf{F} \\ \mathsf{F} & \text{otherwise} \end{cases}$
  - $v(\varphi \wedge \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) = \mathsf{T} \\ \mathsf{F} & \text{otherwise} \end{cases}$
  - $v(\varphi \vee \psi) = \begin{cases} \mathsf{F} & \text{if } v(\varphi) = v(\psi) = \mathsf{F} \\ \mathsf{T} & \text{otherwise} \end{cases}$
  - $v(\varphi \to \psi) = \begin{cases} \mathsf{F} & \text{if } v(\varphi) = \mathsf{T} \text{ and } v(\psi) = \mathsf{F} \\ \mathsf{T} & \text{otherwise} \end{cases}$
  - $v(\varphi \leftrightarrow \psi) = \begin{cases} \mathsf{T} & \text{if } v(\varphi) = v(\psi) \\ \mathsf{F} & \text{otherwise} \end{cases}$

Definitions

- semantic entailment

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

  if $v(\psi) = \mathsf{T}$ whenever $v(\varphi_1) = v(\varphi_2) = \cdots = v(\varphi_n) = \mathsf{T}$, for every valuation v
- formula φ is valid if $v(\varphi) = \mathsf{T}$ for every valuation v
- formula φ is satisfiable if $v(\varphi) = \mathsf{T}$ 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 φ $\exists$ CNF ψ $\exists$ 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
1 -2 4 0
-1 2 -3 -4 0
3 -2 0
```

4 atoms and 3 clauses

$x_1 \vee \neg x_2 \vee x_4$

$\neg x_1 \vee x_2 \vee \neg x_3 \vee \neg x_4$

$x_3 \vee \neg x_2$

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

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 $\leftrightarrow$

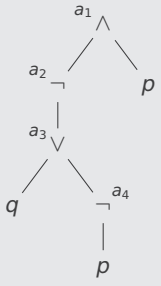
Example (Tseitin's Transformation)

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

  $a_1 \quad \neg(q \vee \neg p) \wedge p \qquad a_3 \quad q \vee \neg p$

  $a_2 \quad \neg(q \vee \neg p) \qquad\quad a_4 \quad \neg p$

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



Lemma

1. $(\varphi \leftrightarrow \neg\psi) \equiv (\varphi \vee \psi) \wedge (\neg\varphi \vee \neg\psi)$
2. $(\varphi \leftrightarrow \psi \wedge \chi) \equiv (\neg\varphi \vee \psi) \wedge (\neg\varphi \vee \chi) \wedge (\varphi \vee \neg\psi \vee \neg\chi)$
3. $(\varphi \leftrightarrow \psi \vee \chi) \equiv (\varphi \vee \neg\psi) \wedge (\varphi \vee \neg\chi) \wedge (\neg\varphi \vee \psi \vee \chi)$

Example (cont'd)

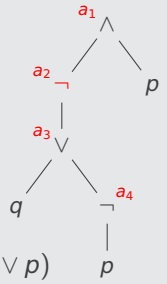
$$
\begin{aligned}
\varphi &\approx a_1 \wedge (a_1 \leftrightarrow a_2 \wedge p) \wedge (a_2 \leftrightarrow \neg a_3) \wedge (a_3 \leftrightarrow q \vee a_4) \wedge (a_4 \leftrightarrow \neg p) \\
&\equiv a_1 \wedge (\neg a_1 \vee a_2) \wedge (\neg a_1 \vee p) \wedge (a_1 \vee \neg a_2 \vee \neg p)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

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

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

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

Example

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

	$\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		
$\Longrightarrow$	1^{d} $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		decide
$\Longrightarrow$	$1^{d}\ \neg 2$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		unit propagate
$\Longrightarrow$	$1^{d}\ \neg 2\ 3$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		unit propagate
$\Longrightarrow$	$1^{d}\ \neg 2\ 3\ 4$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		unit propagate
$\Longrightarrow$	$\neg 1$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		backtrack
$\Longrightarrow$	$\neg 1\ 4$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		unit propagate
$\Longrightarrow$	$\neg 1\ 4\ \neg 3^{d}$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		decide
$\Longrightarrow$	$\neg 1\ 4\ \neg 3^{d}\ 2$ $\parallel \neg 1 \vee \neg 2,\ 2 \vee 3,\ \neg 1 \vee \neg 3 \vee 4,\ 2 \vee \neg 3 \vee \neg 4,\ 1 \vee 4$		unit propagate

Definition (Transition Rules)

- unit propagate $\quad M \parallel F, C \vee l \implies M\, l \parallel F, C \vee l$

  if $M \models \neg C$ and l is undefined in M $\qquad$ unit clause

- pure literal $\quad M \parallel F \implies M\, l \parallel F$

  if l occurs in F and l^c (complement of l) does not occur in F and l is undefined in M

- decide $\quad M \parallel F \implies M \overset{d}{l} \parallel F$

  if l or l^c occurs in F and l is undefined in M

- fail $\quad M \parallel F, C \implies$ fail-state

  if $M \models \neg C$ and M contains no decision literals

- backtrack $\quad M \overset{d}{l} N \parallel F, C \implies M\, l^c \parallel F, C$

  if $M \overset{d}{l} N \models \neg C$ and N contains no decision literals

Outline

Example

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

$$
\begin{array}{rrll}
 & \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \\
\Longrightarrow & \overset{d}{1} \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{decide} \\
\Longrightarrow & \overset{d}{1}\ 2 \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{unit propagate} \\
\Longrightarrow & \overset{d}{1}\ 2\ \overset{d}{3} \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{decide} \\
\Longrightarrow & \overset{d}{1}\ 2\ \overset{d}{3}\ 4 \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{unit propagate} \\
\Longrightarrow & \overset{d}{1}\ 2\ \overset{d}{3}\ 4\ \overset{d}{5} \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{decide} \\
\Longrightarrow & \overset{d}{1}\ 2\ \overset{d}{3}\ 4\ \overset{d}{5}\ \neg 6 \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{unit propagate} \\
\Longrightarrow & \overset{d}{1}\ 2\ \neg 5 \ \| \ \neg 1 \vee 2,\ \neg 3 \vee 4,\ \neg 5 \vee \neg 6,\ 6 \vee \neg 5 \vee \neg 2 & & \text{backjump}
\end{array}
$$

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

Definitions

- backtrack $M \overset{d}{l} N \parallel F, C \implies M\, l^c \parallel F, C$

  if $M \overset{d}{l} N \models \neg C$ and N contains no decision literals

- backjump $M \overset{d}{l} N \parallel F, C \implies M\, l' \parallel F, C$

  if $M \overset{d}{l} N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that

  backjump clause

  - $F, C \models C' \vee l'$
  - $M \models \neg C'$
  - l' is undefined in M
  - l' or l'^c occurs in F or in $M \overset{d}{l} N$

Example (cont'd)

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

Definition

basic DPLL $\mathcal{B}$ consists of transition rules

- unit propagate $\quad M \parallel F, C \vee l \implies M\ l \parallel F, C \vee l$

  if $M \models \neg C$ and l is undefined in M

- decide $\quad M \parallel F \implies M \overset{d}{l} \parallel F$

  if l or l^c occurs in F and l is undefined in M

- fail $\quad M \parallel F, C \implies$ fail-state

  if $M \models \neg C$ and M contains no decision literals

- backjump $\quad M \overset{d}{l} N \parallel F, C \implies M\ l' \parallel F, C$

  if $M \overset{d}{l} N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that

  - $F, C \models C' \vee l'$ and $M \models \neg C'$
  - l' is undefined in M and l' or l'^c occurs in F or in $M \overset{d}{l} N$

Theorem

there are no infinite derivations $\| F \Longrightarrow_{\mathcal{B}} S_1 \Longrightarrow_{\mathcal{B}} S_2 \Longrightarrow_{\mathcal{B}} \cdots$

Proof

- for list of distinct literals M, $|M|$ is length of M
- measure state $M_0 \overset{d}{l}_1 M_1 \overset{d}{l}_2 M_2 \ldots \overset{d}{l}_k M_k \| F$ where $M_0, \ldots, M_k$ contain no decision literals by tuple $(|M_0|, |M_1|, \ldots, |M_k|)$
- compare tuples lexicographically using standard order on $\mathbb{N}$
- every transition step strictly increases measure
- measure is bounded by $(n+1)$-tuple $(n, \ldots, n)$ where n is total number of atoms

Example

$$
\begin{array}{lrlll}
\| \varphi = (\neg 1 \vee 2) \wedge (\neg 3 \vee 4) \wedge (\neg 5 \vee \neg 6) \wedge (6 \vee \neg 5 \vee \neg 2) & & & & (0) \\
\Longrightarrow & \overset{d}{1} \| \varphi & \text{decide} & & (0,0) \\
\Longrightarrow & 1\ \overset{d}{2} \| \varphi & \text{unit propagate} & & (0,1) \\
\Longrightarrow & 1\ 2\ \overset{d}{3} \| \varphi & \text{decide} & & (0,1,0) \\
\Longrightarrow & 1\ 2\ 3\ \overset{d}{4} \| \varphi & \text{unit propagate} & & (0,1,1) \\
\Longrightarrow & 1\ 2\ 3\ 4\ \overset{d}{5} \| \varphi & \text{decide} & & (0,1,1,0) \\
\Longrightarrow & 1\ 2\ 3\ 4\ 5\ \overset{d}{\neg 6} \| \varphi & \text{unit propagate} & & (0,1,1,1) \\
\Longrightarrow & 1\ \overset{d}{2}\ \neg 5 \| \varphi & \text{backjump} & & (0,2)
\end{array}
$$

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

Lemma

1. if $\| F \Longrightarrow^*_{\mathcal{B}} M \| F'$ then
   - $F = F'$
   - M does not contain complementary literals
   - M consists of distinct literals

2. if $\| F \Longrightarrow^*_{\mathcal{B}} M_0 \overset{d}{l}_1 M_1 \overset{d}{l}_2 M_2 \cdots \overset{d}{l}_k M_k \| F$ with no decision literals in $M_0, \ldots, M_k$
   then $F, l_1, \ldots, l_i \vDash M_i$ for all $0 \leqslant i \leqslant k$

Theorem

if $\| F \Longrightarrow_{\mathcal{B}} S_1 \Longrightarrow_{\mathcal{B}} \cdots \Longrightarrow_{\mathcal{B}} S_n \not\Longrightarrow_{\mathcal{B}}$ then

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

Proof

1. (only if) $\| F \Longrightarrow^*_{\mathcal{B}} M \| F \Longrightarrow_{\text{fail}} \text{fail-state}$
   - 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. $\| F \Longrightarrow^*_{\mathcal{B}} M \| F' \not\Longrightarrow_{\mathcal{B}}$
   - $F = F'$ 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

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