Decision Procedures

Martin Blicha

Charles University

26.2.2018

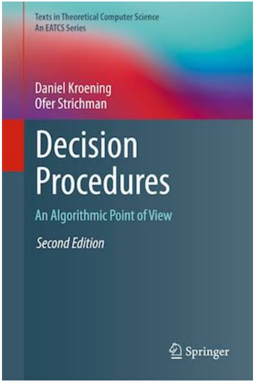
What is the course about?

- Preliminaries
- Satisfiability of boolean formulas
  - Modern SAT solvers
  - Local algorithms
  - BDDs
  - QBF
- SMT
- Decision procedures for theories
  - Equality and uninterpreted functions
  - Linear arithmetic
  - Bit vectors
  - Arrays, memory, pointers
- Combination of theories
- ...

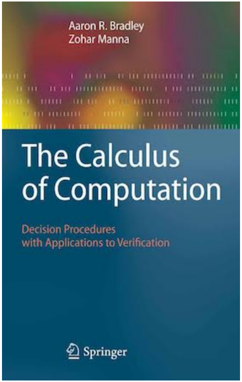
How to pass the course

- Oral exam
- Write your own SAT solver (will be explained at seminar)

Materials

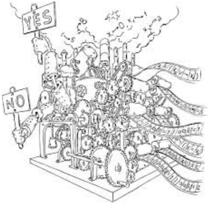


- Kroening D., Strichman O.: Decision Procedures. Second edition. Springer, 2016.



- Bradley A., Manna Z.: The Calculus of Computation. Springer, 2007.

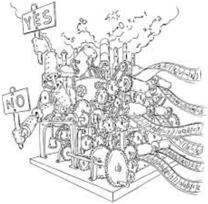
What is a decision procedure?



Intuition

Decision procedure is an algoritm that takes a logical formula as input and decides its satisfiability.

What is a decision procedure?

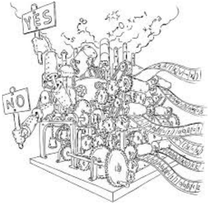


Intuition

Decision procedure is an algoritm that takes a logical formula as input and decides its satisfiability.

- Satisfiable $\rightarrow$ satisfying assignment (model)

What is a decision procedure?



Intuition

Decision procedure is an algoritm that takes a logical formula as input and decides its satisfiability.

- Satisfiable $\rightarrow$ satisfying assignment (model)
- (Unsatisfiable $\rightarrow$ proof of unsatisfiability)

Motivation

Used everywhere where logic is the primary modelling language.

- Hardware verification
  - Verifying designs of electronic circuits
- Software verification
  - Verifying that an assertion in code cannot be violated
- Compiler optimizations
  - Correctness of transformations
- Chemical reaction networks

Language of propositional logic

- Countable set of proposition variables $\{p_0, p_1, \dots\}$
- Logical connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$

Definition (Propositional formula)

1. Every propositional variable is a formula.
2. If φ, ψ are propositional formulas then also $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi$ are propositional formulas.
3. Nothing else is a propositional formula.

Basic definitions

Definition (assignment)

Assignment maps propositional variables to *true* or *false* (1 or 0, $\top$ or $\bot$)

Definition (satisfying assignment)

Formula φ is satisfied under assignment α (of its variables) if it evaluates to *true* under α. We write $\alpha \vDash \varphi$ to denote that φ is satisfied by α (α satisfies φ, α is a model of φ).

Definition (satisfiability and validity)

- A formula is satisfiable if there exists an assignment that satisfies it.
- A formula is a contradiction if it is not satisfiable.
- A formula is valid (tautology) if it is satisfied under all assignments.

Satisfiability and validity

Corollary

φ is valid iff $\neg\varphi$ is not satisfiable.

Satisfiability and validity

Corollary

φ *is valid iff* $\neg\varphi$ *is not satisfiable.*

If we have an algorithm for deciding satisfiability, we have an algorithm also for deciding validity (and vice versa).

Normal forms

Definition (literal, term, clause)

- *Literal* is either a propositional variable or its negation. We say literal is positive or negative, respectively.
- *Term* is a conjunction of literals.
- *Clause* is a disjunction of literals.

Normal forms

Definition (literal, term, clause)

- *Literal* is either a propositional variable or its negation. We say literal is positive or negative, respectively.
- *Term* is a conjunction of literals.
- *Clause* is a disjunction of literals.

Example

Let p, q be propositional variables. Then $p, \neg p, q, \neg q$ are literals, $p \wedge \neg q$ is a term, $\neg p \vee \neg q$ is a clause.

Normal forms

Definition (NNF)

A formula is in negation normal form (NNF), if it contains only $\wedge, \vee, \neg$ as connectives and negation occurs only in front of variables.

Definition (DNF)

A formula is in disjunctive normal form (DNF) if it is a disjunction of terms.

Definition (CNF)

A formula is in conjunctive normal form (CNF) if it is a conjunction of clauses.

Conversion to CNF

Lemma

For every formula there exists an equivalent formula in CNF.

Conversion to CNF

Lemma

For every formula there exists an equivalent formula in CNF.

Idea of an constructive proof

- Convert to NNF.
  - Rewrite connectives using only $\wedge, \vee, \neg$.
  - Apply De Morgan's law to propagate negation inward.
  - Apply double negation rule to eliminate double negations.
- Apply distribution law to propagate disjunction over conjunction.

Conversion to CNF

Lemma

For every formula there exists an equivalent formula in CNF.

Idea of an constructive proof

- Convert to NNF.
  - Rewrite connectives using only $\wedge, \vee, \neg$.
  - Apply De Morgan's law to propagate negation inward.
  - Apply double negation rule to eliminate double negations.
- Apply distribution law to propagate disjunction over conjunction.

The equivalent formula can be *exponentially* larger.

Tseitin's encoding

Lemma (Tseitin)

Every formula can be converted to an equisatisfiable formula in CNF which is larger only by a constant factor.

Tseitin's encoding

Lemma (Tseitin)

Every formula can be converted to an equisatisfiable formula in CNF which is larger only by a constant factor.

Idea

Introduce fresh variables to encode subformulas. Encode the meaning of these fresh variables with clauses. Avoids duplicating whole subformulas.

Tseitin's encoding

1. Build a derivation tree of φ with variables as leaves.

Tseitin's encoding

1. Build a derivation tree of φ with variables as leaves.
2. Introduce a fresh variable for every inner node (Representants).

Tseitin's encoding

1. Build a derivation tree of φ with variables as leaves.
2. Introduce a fresh variable for every inner node (Representants).
3. Encode the meaning of the fresh variables (with clauses).

Tseitin's encoding

1. Build a derivation tree of φ with variables as leaves.
2. Introduce a fresh variable for every inner node (Representants).
3. Encode the meaning of the fresh variables (with clauses).
4. Equisatisfiable formula in CNF is the representant of the whole formula (the root) together with all the encoding clauses.

Tseitin's encoding

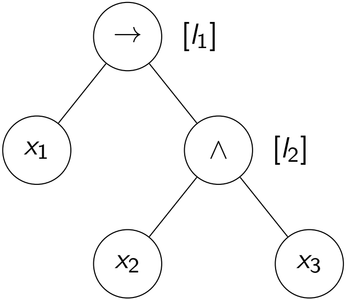
1. Build a derivation tree of φ with variables as leaves.
2. Introduce a fresh variable for every inner node (Representants).
3. Encode the meaning of the fresh variables (with clauses).
4. Equisatisfiable formula in CNF is the representant of the whole formula (the root) together with all the encoding clauses.

Example: $x_1 \to x_2 \wedge x_3$

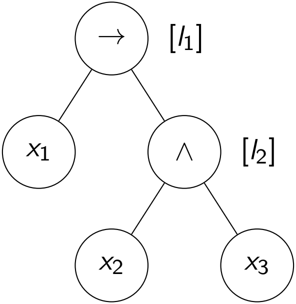
Tseitin's encoding

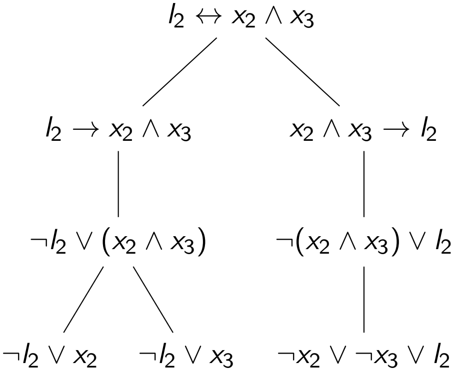
1. Build a derivation tree of φ with variables as leaves.
2. Introduce a fresh variable for every inner node (Representants).
3. Encode the meaning of the fresh variables (with clauses).
4. Equisatisfiable formula in CNF is the representant of the whole formula (the root) together with all the encoding clauses.

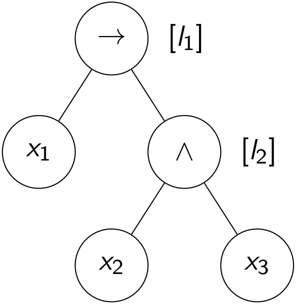
Example: $x_1 \rightarrow x_2 \wedge x_3$



Tseitin's encoding







$$l_1 \leftrightarrow x_1 \rightarrow l_2$$

$$l_1 \rightarrow (x_1 \rightarrow l_2) \qquad (x_1 \rightarrow l_2) \rightarrow l_1$$

$$\neg l_1 \vee (x_1 \rightarrow l_2) \qquad (x_1 \wedge \neg l_2) \vee l_1$$

$$\neg l_1 \vee \neg x_1 \vee l_2 \qquad l_1 \vee x_1 \qquad l_1 \vee \neg l_2$$

Tseitin's encoding (formally)

Definition

Let φ be a formula and let *Repr* be a mapping of subformulas of φ to propositional variables (representants) such that:

- $Repr(p) = p$ for p a propositional variable
- $Repr(\psi) = l_\psi$ is a new unique propositional variable for non-atomic subformula.

Tseitin's encoding (formally)

Definition

Let φ be a formula and let *Repr* be a mapping of subformulas of φ to propositional variables (representants) such that:

- $Repr(p) = p$ for p a propositional variable
- $Repr(\psi) = l_\psi$ is a new unique propositional variable for non-atomic subformula.

Let *Enc* be a mapping of subformulas to CNF formulas defined as follows:

- p a propositional variable. $Enc(p) = true$
- $\psi = \psi_1 \wedge \psi_2$. $Enc(\psi) = (\neg l_\psi \vee l_{\psi_1}) \wedge (\neg l_\psi \vee l_{\psi_2}) \wedge (l_\psi \vee \neg l_{\psi_1} \vee \neg l_{\psi_2})$
- $\psi = \psi_1 \vee \psi_2$. $Enc(\psi) = (l_\psi \vee \neg l_{\psi_1}) \wedge (l_\psi \vee \neg l_{\psi_2}) \wedge (\neg l_\psi \vee l_{\psi_1} \vee l_{\psi_2})$
- $\psi = \psi_1 \rightarrow \psi_2$. $Enc(\psi) = (l_\psi \vee l_{\psi_1}) \wedge (l_\psi \vee \neg l_{\psi_2}) \wedge (\neg l_\psi \vee \neg l_{\psi_1} \vee l_{\psi_2})$
- $\psi = \neg\psi_1$. $Enc(\psi) = (l_\psi \vee l_{\psi_1}) \wedge (\neg l_\psi \vee \neg l_{\psi_1})$

Tseitin's encoding formally

Lemma

Let φ *be a formula and let* $\varphi' = Repr(\varphi) \wedge \bigwedge_{\psi} Enc(\psi)$ *for every* ψ *a subformula of* φ*. Then* φ *and* φ' *are equisatisfiable and* $|\varphi'| = O(|\varphi|)$

Tseitin's encoding formally

Lemma

Let φ *be a formula and let* $\varphi' = Repr(\varphi) \wedge \bigwedge_{\psi} Enc(\psi)$ *for every* ψ *a subformula of* φ*. Then* φ *and* φ' *are equisatisfiable and* $|\varphi'| = O(|\varphi|)$

Proof.

Idea:

- Satisfying assignment of φ' restricted to the original variables is a satisfying assignment of φ
- Satisfying assignment α of φ can be extended to a satisfying assignment α' of φ' by assigning for each introduced variable l_ψ: $\alpha'(l_\psi) = \alpha(\psi)$.

□

Optimizing Tseitin's encoding

- Do not encode negative literals

Optimizing Tseitin's encoding

- Do not encode negative literals
- Consider n-ary conjunctions and disjunctions.

Optimizing Tseitin's encoding

- Do not encode negative literals
- Consider n-ary conjunctions and disjunctions.
- Use one-sided Tseitin's encoding for formulas in NNF.