Résolution efficace des modèles logiques

IA303

Alexandre Chapoutot

ENSTA Paris

2019-2020

Course Outline

Main goals of the course:

- To be able to model decision problems using logical formulas combined with other theories.
- Know the solving algorithms for SAT/SMT problems

Remarks:

- consider unquantified logical formula
- consider semantic methods based on exhaustivity

General information

- Teacher: Alexandre Chapoutot
- Mail: `alexandre.chapoutot@ensta-paris.fr`
- Office: R216 at U2IS, ENSTA Paris, 828 bd des maréchaux, Palaiseau
- Class: Tuesday morning (9h-12h)
- Course grade: TBD
- Webpage:

  `http://perso.ensta-paris.fr/~chapoutot/sat-smt/`

Syllabus

Nov. 12, 2019 Propositional logic

Nov. 26, 2019 SAT solver algorithms

Dec. 03, 2019 First-order theories and SMT solver algorithms

Dec. 10, 2019 Theory of uninterpreted functions

Dec. 17, 2019 Theory of linear real expressions

Jan. 07, 2020 Theory of nonlinear real expressions

Jan. 14, 2020 Combination of theories

Jan. 21, 2020 Exam: TBD

Propositional logic

Syntax of PL

The Backus-Naur Form (BNF) of the grammar PL is:

$$P ::= \top \mid \bot \mid \text{id} \mid \neg P \mid P_1 \wedge P_2 \mid P_1 \vee P_2 \mid P_1 \implies P_2 \mid P_1 \iff P_2$$

We denote by $\mathcal{PL}$ the set of all well written PL formula.
Vocabulary

Atom is of two kinds

- **truth symbols** $\bot$ (false) or $\top$ (true)
- **ident** or **propositional variables** p, q, r, s, ... taken into a countable set $\mathcal{V}$

Literal is an atom α or its negation $\neg\alpha$

Formula are made of atom associated to **logical connectors**: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\implies$ (implication), and $\iff$ (if and only if).

Syntax of PL – 2

We define the relative precedence of the logical operators from highest to lowest as follows

$$\neg, \wedge, \vee, \implies, \iff$$

Moreover, $\implies$ and $\iff$ are associative to the right, so

$$p \implies q \implies r \quad \equiv \quad p \implies (q \implies r)$$

Example

- $F : p \wedge q \implies p \vee \neg q \equiv (p \wedge q) \implies (p \vee (\neg q))$

Semantics of PL

An **interpretation** I is a (partial) mapping from $\mathcal{V}$ to truth value $\{0, 1\} = \mathbb{B}$.

Example

$$I : \{p \mapsto 0; q \mapsto 1; r \mapsto 1; \ldots\}$$

Definition

The semantics or the meaning of a PL formula F is its truth value with respect to a given interpretation I

We will denote by $[\![\cdot]\!]_{\mathsf{PL}}$ the semantic function of PL formula, *i.e.*,

$$[\![\cdot]\!]_{\mathsf{PL}} : \mathcal{PL} \to (\mathcal{V} \to \mathbb{B}) \to \mathbb{B}$$

Inductive definition of $[\![]\!]_{\mathsf{PL}}$

Following the structure of a PL formula F, the semantic function $[\![]\!]_{\mathsf{PL}}$ is defined as followed

- if $F \equiv \bot$ then $[\![F]\!]_{\mathsf{PL}}(I) = 0$ for all I
- if $F \equiv \top$ then $[\![F]\!]_{\mathsf{PL}}(I) = 1$ for all I
- if $F \equiv \mathrm{id} \in \mathcal{V}$ then $[\![F]\!]_{\mathsf{PL}}(I) = I(\mathrm{id})$
- if $F \equiv \neg F_1$ then $[\![F]\!]_{\mathsf{PL}}(I) = \text{not } [\![F_1]\!]_{\mathsf{PL}}(I)$
- if $F \equiv F_1 \wedge F_2$ then $[\![F]\!]_{\mathsf{PL}}(I) = [\![F_1]\!]_{\mathsf{PL}}(I)$ and $[\![F_2]\!]_{\mathsf{PL}}(I)$
- if $F \equiv F_1 \vee F_2$ then $[\![F]\!]_{\mathsf{PL}}(I) = [\![F_1]\!]_{\mathsf{PL}}(I)$ or $[\![F_2]\!]_{\mathsf{PL}}(I)$
- if $F \equiv F_1 \implies F_2$ then $[\![F]\!]_{\mathsf{PL}}(I) = [\![\neg F_1 \vee F_2]\!]_{\mathsf{PL}}(I)$
- if $F \equiv F_1 \iff F_2$ then $[\![F]\!]_{\mathsf{PL}}(I) = [\![(F_1 \implies F_2) \wedge (F_2 \implies F_1)]\!]_{\mathsf{PL}}(I)$

Remark there is a difference between syntax and semantics notations.

Semantics of Boolean operator

The semantics of PL formula is based on the semantics of Boolean operator. It is given by the **Truth Tables**
p, q are propositional variables

p	q	not p	p and q	p or q
0	0	1	0	0
0	1	1	0	1
1	1	0	1	1
1	0	0	0	1

And we use rules as

- $p \implies q \equiv \neg p \vee q$
- $p \iff q \equiv (p \implies q) \wedge (q \implies p)$

to define the Truth value of $\implies$ and $\iff$

Example

Inputs:

- PL formula $F : (p \implies q) \wedge p$
- Interpretation $I : \{p \mapsto 1; q \mapsto 0\}$

Output: Truth value

$$
\begin{aligned}
[\![F]\!]_{\mathrm{PL}}(I) &= [\![(p \implies q) \wedge p]\!]_{\mathrm{PL}}(I) \\
&= [\![(p \implies q)]\!]_{\mathrm{PL}}(I) \text{ or } [\![p]\!]_{\mathrm{PL}}(I) \\
&= [\![(\neg p \wedge q)]\!]_{\mathrm{PL}}(I) \text{ or } 1 \\
&= ([\![\neg p]\!]_{\mathrm{PL}}(I) \text{ or } [\![q]\!]_{\mathrm{PL}}(I)) \text{ or } 1 \\
&= (\text{not } [\![p]\!]_{\mathrm{PL}}(I) \text{ or } 0) \text{ or } 1 \\
&= (\text{not } 1) \text{ or } 0 \text{ or } 1 \\
&= 0 \text{ or } 0 \text{ or } 1 \\
&= 1
\end{aligned}
$$

Satisfiability, Validity

Definition: Satisfiability

A PL formula F is **satisfiable** iff $\exists I$ such that $[\![F]\!]_{\mathsf{PL}}(I) = 1$. F is **unsatisfiable** iff $\forall I$, $[\![F]\!]_{\mathsf{PL}}(I) = 0$ or $\nexists I$, $[\![F]\!]_{\mathsf{PL}}(I) = 1$

Definition: Validity

F is **valid** or a **theorem**, iff $\forall I$, $[\![F]\!]_{\mathsf{PL}}(I) = 1$

A strong link between validity and satisfiability

$$F \text{ is valid iff } \neg F \text{ is unsatisfiable}$$

Definition: Equivalence

F_1 and F_2 are equivalent, *i.e.*, $F_1 \equiv F_2$
iff $\forall I$, $[\![F_1]\!]_{\mathsf{PL}}(I) = [\![F_2]\!]_{\mathsf{PL}}(I)$

Vocabulary

A Clause

is a disjunction of litterals, *e.g.*,

$$K \equiv a \vee \neg b \vee c \vee d$$

which is also denoted by $K = \{a, \neg b, c, d\}$

A cube

is a conjunction of litterals, *e.g.*,

$$C \equiv \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_n$$

also denoted by $C = \{\ell_1, \ell_2, \ldots, \ell_n\}$

Normal forms – NNF

Negative Normal Form (NNF)

Negation connectors only appear in front of literals and $\neg$, $\vee$, and $\wedge$ are the only available connectors.

Construction rules:

- $\neg\neg F \rightsquigarrow F, \quad \neg\top \rightsquigarrow \bot, \quad \neg\bot \rightsquigarrow \top$
- $\neg(F_1 \vee F_2) \rightsquigarrow \neg F_1 \wedge \neg F_2$
- $\neg(F_1 \wedge F_2) \rightsquigarrow \neg F_1 \vee \neg F_2$
- $F_1 \implies F_2 \rightsquigarrow \neg F_1 \vee F_2$
- $F_1 \iff F_2 \rightsquigarrow (F_1 \implies F_2) \wedge (F_2 \implies F_1)$

Example of NNF transformation

Transformation of $F : \neg(P \implies \neg(P \wedge Q))$ in NNF

- $F' : \neg(\neg P \vee \neg(P \wedge Q))$
- $F'' : \neg\neg P \wedge \neg\neg(P \wedge Q)$
- $F''' : P \wedge P \wedge Q$

Hence F''' is equivalent to F but in NNF

Normal forms – DNF

Disjunctive normal form (DNF)

Disjunction of conjunctions of literals

$$\bigvee_i \bigwedge_j \ell_{ij} \quad \text{forall literals} \quad \ell_{ij}$$

Constructions rules:

- Transform F into NNF
- $(F_1 \vee F_2) \wedge F_3 \rightsquigarrow (F_1 \wedge F_3) \vee (F_2 \wedge F_3)$
- $F_1 \wedge (F_2 \vee F_3) \rightsquigarrow (F_1 \wedge F_2) \vee (F_1 \wedge F_3)$

Example of DNF transformation

Transformation of $F : (Q_1 \vee \neg\neg Q_2) \wedge (\neg R_1 \implies R_2)$ in DNF

- $F' : (Q_1 \vee Q_2) \wedge (R_1 \vee R_2)$
- $F'' : (Q_1 \wedge (R_1 \vee R_2)) \vee (Q_2 \wedge (R_1 \vee R_2))$
- $F''' : (Q_1 \wedge R_1) \vee (Q_1 \wedge R_2) \vee (Q_2 \wedge R_1) \vee (Q_2 \wedge R_2)$

Hence F''' is equivalent to F but in DNF

Normal forms – CNF

Conjunctive normal form (CNF)

Conjunction of disjunctions of literals

$$\bigwedge_i \bigvee_j \ell_{ij} \quad \text{forall literals} \quad \ell_{ij}$$

Constructions rules:

- Transform F into NNF
- $(F_1 \wedge F_2) \vee F_3 \rightsquigarrow (F_1 \vee F_3) \wedge (F_2 \vee F_3)$
- $F_1 \vee (F_2 \wedge F_3) \rightsquigarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3)$

Remark

This produces an equivalent CNF formula but its size can be exponentially bigger!

Example of CNF transformation

Transformation of $F : (Q_1 \wedge \neg\neg Q_2) \vee (\neg R_1 \implies R_2)$ in DNF

- $F' : (Q_1 \wedge Q_2) \vee (R_1 \vee R_2)$
- $F'' : (Q_1 \vee (R_1 \vee R_2)) \vee (Q_2 \vee (R_1 \vee R_2))$

Hence F'' is equivalent to F but in CNF

Equisatifiable formula

Tseitin transformation produces a CNF formula F' from F which grows linearly but it will be only equisatifiable, *i.e.*, F is satisfiable iff F' is satisfiable.

The idea is to add new variables:

- $A \wedge B \rightsquigarrow X$ and the CNF formula is produced $(\neg A \vee \neg B \vee X) \wedge (A \vee \neg X) \wedge (B \vee \neg X)$
- $A \vee B \rightsquigarrow X$ and the CNF formula is produced $(A \vee B \vee \neg X) \wedge (\neg A \vee X) \wedge (\neg B \vee X)$

Remark: hence we can focus on CNF formula

SAT problem

- It is algorithmic problem to decide if a CNF PL formula is valid or not.
- It is a NP-hard problem for a 3-SAT problem (clauses have at most 3 literals)

A simple decision procedure

Based on Truth Table a simple algorithm to decide if a PL formula F is satisfiable can be given.

```
let rec sat f =
    if f = ⊤ then true
    else if f = ⊥ then false
    else
      let p = choose(vars(f)) in
      (sat f{p ↦ ⊤}) or (sat f{p ↦ ⊥})
```

See Lecture 2 for an improved algorithm (CDCL)

Problem solving with SAT

Design/Prove flow

$$\text{Problem} \rightarrow \text{PL formula} \rightarrow \text{CNF} \rightarrow \text{SAT Solver} \rightarrow \text{Yes/No}$$

- Transformation from Problem to PL formula is called the *encoding* into SAT