Logical reasoning and programming

SAT solving (cont'd), FOL, and SMT

Karel Chvalovský

CIIRC CTU

Recap

We deal with formulae in conjunctive normal form (CNF)

$$(\cdots \vee \cdots \vee \cdots) \wedge \cdots \wedge (\cdots \vee \cdots \vee \cdots)$$

and we represent them using

$$\{\{\ldots\}, \ldots, \{\ldots\}\}.$$

Our problem, given a formula φ in CNF, is $\varphi \in \texttt{SAT}$?

We have discussed various approaches how to solve this problem that work very well in practice, like CDCL.

How to encode common problems into SAT?

Planning

In classical planning we want to produce a sequence of actions that translate an initial state into a goal state.

The plan existence problem is known to be PSPACE-complete. Hence it is not (assuming $\mathrm{NP} \neq \mathrm{PSPACE}$) easily solvable using SAT. However, if we consider only plans up to some length, then it is solvable by SAT, because the lengths of plans are usually polynomially bounded.

Planning as a SAT problem

We encode as a CNF formula "there exists a plan of length k", denoted φ_k, and search iterativelly.

- If $\varphi_k \in \mathtt{SAT}$, then we extract a plan from a satisfying assignment.
- If $\varphi_k \notin \mathtt{SAT}$, then we continue with φ_{k+1}.

Classical planning (recap)

We have a set of state variables $X = \{x_1, \ldots, x_n\}$ that are assigned values from a finite set. A state s is such an assignment for X, we write $\{x_1 = v_1, \ldots, x_n = v_n\}$. A set of conditions is a subset of a state.

We have

- an initial state,
- a set of goal conditions—a goal state is such a state that satisfies all the goal conditions.

Moreover, we have a set of actions A where every $a \in A$ has preconditions and effects which are both sets of conditions.

Example

We have packages in a depot and we want to deliver them in the next 24 hours using a truck.

There exists a plan of length k in SAT

We introduce propositional variables for

- actions—p_a^t meaning the action a is used in the step t,
- states—$q_{x=v}^t$ meaning $x = v$ holds in the step t,

for every $a \in A$, $x \in X$, possible value v of x, and step $t \leq k$.

Then we describe all the required properties of a valid plan by a conjunction of clauses:

- the initial state,
- the goal conditions are satisfied after k steps,
- state variables are assigned exactly one value,
- exactly one action is performed in one step,
- the values of state variables change only by actions,
- an applied action must satisfy preconditions and effects.

Note this is similar to the proof of Cook–Levin theorem.

Planning using SAT

We can do various improvements, e.g.,

- perform more actions in a step if they are non-conflicting,
- introduce variables for transitions instead of assignments,
- symmetry breaking.

Incremental SAT solving

Instead of solving a new problem for every k, we can observe that many parts remain the same—we solve a sequence of similar SAT problems. We want to add and remove clauses, but keep learned clauses and variable scores.

Note that in our problem we only add clauses and change the goal conditions, which are described by unit clauses, when we go from φ_k to φ_{k+1}.

Assumptions

Clearly, adding clauses is possible in CDCL, but removing clauses can lead to various problems. However, we have

- a formula φ and
- assumptions $l_1, \ldots, l_n$, where l_i are literals.

The question is whether $\varphi \wedge l_1 \wedge \cdots \wedge l_n \in \mathtt{SAT}$. It is incremental, because we can change assumptions and add new clauses.

We can select all the assumptions as decision variables and continue as always. Hence we can keep all learned clauses from CDCL!

Bounded model checking

It is very similar to planning. We want to verify a property of an automaton with transition states, an initial state, and a given property P that has to be valid at each step.

Bounded model checking as a SAT problem

We bound the number of steps to k and try to reach in k steps a state where P fails. Hence φ_k means "there is a state reachable in k steps where P fails".

- If $\varphi_k \in \mathtt{SAT}$, then we extract a bug from a satisfying assignment.
- If $\varphi_k \notin \mathtt{SAT}$, then we continue with φ_{k+1}.

How to encode typical constraints

We want to express

$$p_1 + p_2 + \cdots + p_n \bowtie k,$$

where $\bowtie \in \{\leq, \geq, =\}$, k is a positive integer, and $\sum_{1 \leq i \leq n} p_i$ is equal to the number of true p_is.

- $=$ is expressed as both $\leq$ and $\geq$,
- ≥ 1 is $\{p_1, \ldots, p_n\}$,
- ≤ 1 is
  - pairwise—$\mathcal{O}(n^2)$ clauses by $\{\, \{\overline{p_i}, \overline{p_j}\} : 1 \leq i < j \leq n \,\}$,
  - sequential counter—$\mathcal{O}(n)$ clauses and $\mathcal{O}(n)$ new variables,
  - bitwise encoding—$\mathcal{O}(n \log n)$ clauses and $\mathcal{O}(\log n)$ new variables,
- $\geq k$ is no more than $n - k$ literals can be false,
- $\leq k$ use generalized pairwise, sequential counters, BDDs, sorting networks, (pairwise) cardinality networks, ...

Or use a pseudo-Boolean (PB) solver for $\sum a_i p_i \bowtie k$.

Consistency and arc-consistency

A very nice property of encodings, e.g., for an encoding of constraints. We say that an encoding is

consistent
: if any partial assignment that cannot be extended to a satisfying assignment (is inconsistent) leads to a conflict by unit propagation,

arc-consistent
: if consistent and unit propagation eliminates values that are inconsistent.

Example

For ≤ 1 we have

consistency
: if two variables are true, then unit propagation produces a conflict,

arc-consistency
: if a variable is true, then unit propagation assigns false to all other variables. (+consistency)

Finite-domain encoding

We encode that a variable x takes one of the values $\{1, \ldots, n\}$.

One-hot encoding

- We use x_i for x takes value i (n variables),
- we need x has exactly one value,
- easy to use constraints and other rules.

Unary encoding (order encoding)

- $\underbrace{1 \ldots 1}_{i-1}\underbrace{0 \ldots 0}_{n-i}$ for x takes value i ($n-1$ variables),
- we need $\{\overline{x_{j+1}}, x_j\}$ for $1 \leq j < n-1$,
- easy to express ranges, e.g. $3 \leq x < 10$.

Binary encoding

- We encode i as a binary number ($\lceil \log n \rceil$ variables),
- if $n \neq 2^k$ some values are not valid,
- using constraints and other rules can be non-trivial.

MaxSAT

There are various variants of SAT. For example, many problems in computer science are expressible as the maximum satisfiability problem—what is the maximum number of clauses that can be satisfied simultaneously.

We usually have (weighted) partial MaxSAT with two types of clauses:

- hard—must be satisfied,
- soft—desirable to be satisfied (possibly with weights)

and we want to maximize the sum of the weights of satisfied soft clauses.

You can check benchmark results at MaxSAT Evaluation 2023. For example RC2 (Python), MaxHS (also MIP solvers), Open-WBO (incomplete solver) and its extensions. There exists the standard WCNF format so we can experiment.

MaxSAT via SAT (initial idea)

We can replace all soft clauses

$$c_1, \ldots, c_n$$

by

$$c_1 \vee r_1, \ldots, c_n \vee r_n,$$

where $r_1, \ldots, r_n$ are fresh variables, called relaxation variables.

And we express that at most m clauses are not satisfied by adding

$$r_1 + \cdots + r_n \leq m.$$

By an iterative calling of a SAT solver, we can solve the MaxSAT problem (minimize m).

For more details on MaxSAT, check this tutorial and this chapter from Biere, M. Heule, et al. 2021.

Unsatisfiable cores

Let φ and ψ be unsatisfiable formulae in CNF such that $\varphi \subseteq \psi$. We say that

- φ is an *unsatisfiable core* of ψ,
- φ is a *minimal unsatisfiable core* of ψ, if every proper subset of φ is satisfiable.

A very important (and hard) practical problem is to extract minimal unsatisfiable cores. Used, for example, in MaxSAT and formal verification.

Core-guided MaxSAT

- Start with all hard and soft clauses and repeat until SAT:
  - Find an unsatisfiable core and relax only soft clauses (many variants how to do that) in the unsatisfiable core.
- Very good for practical (industrial) problems.

What should you use to solve problems in NP?

You have seen many approaches how to solve such problems in Combinatorial Optimization, for example,

- Integer Linear Programming and
- Constraint Programming (MiniZinc Challenge 2023 Results).

Here we have discussed (Max)SAT and we will discuss other approaches later on.

What should you use?

A useful rule of thumb is to select a method based on the language suitable for your problem, see, for example, Which solver should I use? at Google OR-Tools.

See also, for example, LP/CP Programming Contest 2023.

SAT solving summary

SAT solvers are very powerful, among other things, thanks to

- small representations in CNFs,
- preprocessing, (inprocessing),
  - subsumption,
  - variable elimination, (variable addition),
  - symmetry breaking,
- unit propagation,
  - good data structures for backtracking,
- clause learning and back-jumping,
  - restarts,
  - deletion of learned clauses,
  - learned clause minimization,
- fast decision heuristics,
- local search,
- and much much more techniques we did not mention.

We do clever tricks, but first and foremost they have to be fast!

Satisfiability Modulo Theories (SMT)—example

We would like to know whether the following formula

$$f(a) = b \wedge f(a) \neq b$$

is satisfiable; that is loosely speaking that there exists an interpretation satisfying simultaneously both conjuncts.

Satisfiability Modulo Theories (SMT)—example

We would like to know whether the following formula

$$f(a) = b \land f(a) \neq b$$

is satisfiable; that is loosely speaking that there exists an interpretation satisfying simultaneously both conjuncts.

Clearly, it is unsatisfiable regardless of the meaning of symbols in it, because it has a propositional form

$$p \land \neg p,$$

where p is $f(a) = b$.

Satisfiability Modulo Theories (SMT)—example

We would like to know whether the following formula

$$(a \times (f(b) + f(c)) = d) \wedge (b \times (f(a) + f(c)) \neq d) \wedge (a = b)$$

is satisfiable.

Do we need to know the interpretations of $\times$, $+$, and f?

Satisfiability Modulo Theories (SMT)—example

We would like to know whether the following formula

$$(a \times (f(b) + f(c)) = d) \wedge (b \times (f(a) + f(c)) \neq d) \wedge (a = b)$$

is satisfiable.

Do we need to know the interpretations of $\times$, $+$, and f?
No, it is unsatisfiable modulo the theory of equality.

Satisfiability Modulo Theories (SMT)—example

We would like to know whether the following formula

$$(x = 0 \vee x = 1) \wedge (x + y + z \neq 0) \wedge (f(y) > f(z))$$

is satisfiable.

Satisfiability Modulo Theories (SMT)—example

We would like to know whether the following formula

$$(x = 0 \lor x = 1) \land (x + y + z \neq 0) \land (f(y) > f(z))$$

is satisfiable.

It is satisfiable in a union of theories called unquantified linear real arithmetic with uninterpreted sort and function symbols (QF_UFLRA):

- quantifier free (QF),
- linear real arithmetic (LRA),
- uninterpreted functions (UF).

First-Order Logic (recap)

We moved to First-Order Logic (FOL) and hence we have

- logical symbols
  - variables—an infinite (countable) set denoted Var
  - quantifier symbols $\forall$ and $\exists$
  - logical connectives
  - auxiliary symbols — parentheses, punctuation symbols, ...
- non-logical symbols accompanied by their arity (the number of arguments)
  - function symbols
    - nullary functions are called constants
  - predicate (ralation) symbols
    - nullary predicate symbols are essentially propositional variables

The logical symbols are fixed, but the non-logical symbols form a language L.

Variants of FOL

There are various variants of FOL, e.g., they differ in:

Equality

Some symbols like $=$ can be either logical (FOL with equality), or non-logical (FOL without equality). We will discuss both variants.

Many-sorted language

Sometimes it is convenient (and common in SMT) to talk about different types of objects. Hence variables, function and predicate symbols can be accompanied by simple types.

We can "easily" simulate finitely many sorts by introducing new predicates so to simplify things, we will use sorts only *implicitly*.

Example

Instead of having different types of variables for real numbers and integers, we can say $\forall X \in \mathbb{R}(\exists Y \in \mathbb{Z}(\dots)$ that is a shortcut for $\forall X(X \in \mathbb{R} \rightarrow (\exists Y(Y \in \mathbb{N} \wedge (\dots))))$. The exact translation depends on (implicit) quantifiers!

Terms

The set of all terms in a language L, denoted $Term_L$, is the smallest set satisfying

- every variable (an element of Var) is a term in L,
- if f is an n-ary function in L and $t_1, \ldots, t_n$ are terms in L, then $f(t_1, \ldots, t_n)$ is a a term in L.

A term s is a subterm of term t if s is a substring of t.

Example

X, c, and $g(a, g(Y))$ are all terms where X and Y are variables, $c/0$ and $a/0$ are nullary functions (constants), $g/2$ is a binary function, and $g/1$ is a unary function.

Formulae

Let p be an n-ary predicate symbol in L and $t_1, \ldots, t_n$ be terms in L, then $p(t_1, \ldots, t_n)$ is an atomic formula (or atom) in L.

In FOL with equality, let t_1 and t_2 be terms in L, then $(t_1 = t_2)$ is also an atomic formula in L.

The set of all formulae in a language L, denoted Fml_L, is the smallest set such that

- every atomic formula in L is a formula in L,
- if φ and ψ are formulae in L, X is a variable, then $(\forall X \varphi)$, $(\exists X \varphi)$, $(\neg \varphi)$, $(\varphi \to \psi)$, $(\varphi \wedge \psi)$, and $(\varphi \vee \psi)$ are formulae[1] in L.

A formula ψ is a subformula of φ if ψ is a substring of φ.

We usually write only parentheses that are necessary for unambiguous reading.

[1] $\varphi \leftrightarrow \psi$ is a shortcut for $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

Semantics

We have formulae (syntactic objects) and we want to assign a meaning (semantics) to them.

The meaning of

- logical symbols is fixed,
- non-logical symbols is problem specific.

Example

We want to show that

$$p(X, f(Y)) \rightarrow \exists Z(p(X, Z) \wedge p(Z, f(Y)))$$

does not hold if

- everything is interpreted over $\mathbb{Z}$,
- $p/2$ is interpreted as $<$ on $\mathbb{Z}$,
- $f/1$ is interpreted as multiplication by -1, and
- the evaluation of X and Y is 1 and -2, respectively.

Semantics

Model (or interpretation)

A model (or interpretation) for a language L, denoted $\mathcal{M} = (D, i)$, consists of a non-empty set D (domain) and a function i (interpretation) on D such that

- if f is an n-ary function symbol in L, then $i(f)\colon D^n \to D$,
- if p is an n-ary preditace symbol in L, then $i(p) \subseteq D^n$.

Evaluation

Let $\mathcal{M} = (D, i)$ be a model for L, an evaluation in $\mathcal{M}$ is any function $e\colon \mathit{Var} \to D$. Note that $e(X \mapsto a)$, for $X \in \mathit{Var}$ and $a \in D$, is the same as e, but gives a to X.

The value of a term t under an evaluation e in $\mathcal{M} = (D, i)$, denoted $t^{\mathcal{M}}[e]$, is defined recursively

- $X^{\mathcal{M}}[e] = e(X)$, if $X \in \mathit{Var}$,
- $f(t_1, \ldots, t_n)^{\mathcal{M}}[e] = i(f)(t_1^{\mathcal{M}}[e], \ldots, t_n^{\mathcal{M}}[e])$, if f is n-ary function symbol.

Tarski's definition of truth

Let $\mathcal{M} = (D, i)$ be a model for L, e be an evaluation in $\mathcal{M}$, then we say that a formula φ is satisfied in $\mathcal{M}$ by e, denoted $\mathcal{M} \models \varphi[e]$, or e satisfies φ in $\mathcal{M}$, if

- $\mathcal{M} \models p(t_1, \ldots, t_n)[e]$ iff $(t_1^{\mathcal{M}}[e], \ldots, t_n^{\mathcal{M}}[e]) \in i(p)$, where p is n-ary predicate symbol in L,
- $\mathcal{M} \models (t_1 = t_2)[e]$ iff $(t_1^{\mathcal{M}}[e], t_2^{\mathcal{M}}[e]) \in \mathrm{id}_D$, (in FOL with eq.)
- $\mathcal{M} \models (\neg\psi)[e]$ iff $\mathcal{M} \not\models \psi[e]$,
- $\mathcal{M} \models (\psi \to \chi)[e]$ iff $\mathcal{M} \not\models \psi[e]$ or $\mathcal{M} \models \chi[e]$,
- $\mathcal{M} \models (\psi \wedge \chi)[e]$ iff $\mathcal{M} \models \psi[e]$ and $\mathcal{M} \models \chi[e]$,
- $\mathcal{M} \models (\psi \vee \chi)[e]$ iff $\mathcal{M} \models \psi[e]$ or $\mathcal{M} \models \chi[e]$,
- $\mathcal{M} \models (\forall X \psi)[e]$ iff for every $a \in D$ holds $\mathcal{M} \models \psi[e(X \mapsto a)]$,
- $\mathcal{M} \models (\exists X \psi)[e]$ iff exists $a \in D$ s.t. $\mathcal{M} \models \psi[e(X \mapsto a)]$.

A formula φ is satisfiable, if there is $\mathcal{M}$ and e s.t. $\mathcal{M} \models \varphi[e]$. A set of formulae Γ is satisfiable, if there is $\mathcal{M}$ and e s.t. $\mathcal{M} \models \varphi[e]$, for every $\varphi \in \Gamma$.

Semantic consequence relation

A formula φ is valid (or holds) in $\mathcal{M}$, denoted $\mathcal{M} \models \varphi$, if φ is satisfied in $\mathcal{M}$ by any evaluation e.

A formula φ follows from (or is a consequence of) a set of formula Γ, denoted $\Gamma \models \varphi$, if and only if for any model $\mathcal{M}$ and evaluation e, if for every $\psi \in \Gamma$ holds $\mathcal{M} \models \psi[e]$, then $\mathcal{M} \models \varphi[e]$. We write $\models \varphi$, if $\Gamma = \emptyset$ and say that φ is valid (or holds).

$$\Gamma \models \varphi \quad \text{iff} \quad \forall \mathcal{M} \forall e (\forall \psi \in \Gamma (\mathcal{M} \models \psi[e]) \Rightarrow \mathcal{M} \models \varphi[e])$$

Note that

$$\Gamma \models \varphi \quad \text{iff} \quad \Gamma \cup \{\neg\varphi\} \text{ is unsatisfiable.}$$

We say that two formulae φ and ψ are (semantically) equivalent, denoted $\varphi \equiv \psi$, if $\{\varphi\} \models \psi$ and $\{\psi\} \models \varphi$.

SMT and FOL language

It is common in SMT that we are mainly interested in formulae containing no variables (and hence no quatifiers), they are called ground formulae.

Note that when it comes to satisfiability, uninterpreted constant symbols behave like free variables; they are not bounded by quantification (nor implicitly). However, this can be slightly misleading, because later on all variables will be implicitly universally quantified.

Strictly speaking, we should talk about expansions of a theory, because we add new constants into our language, however, we will happily ignore this formal problem (or we can treat them as free variables).

Example

$\varphi = p(X, f(a, Y)) \wedge q(X, Z, c)$ is not ground, but $\varphi' = p(x, f(a, y)) \wedge q(x, z, c)$ is ground. Moreover, φ and φ' are equisatisfiable.

Why are we interested in this fragment of FOL?

For example, say that a compiler produced from

$$z = (x_1 + y_1) \cdot (x_2 + y_2);$$

the following code

$$\begin{aligned} u_1 &= x_1 + y_1; \\ u_2 &= x_2 + y_2; \\ z &= u_1 \cdot u_2; \end{aligned}$$

So we want to show that this translation is correct by proving

$$\begin{aligned} &(u_1 = x_1 + y_1) \land (u_2 = x_2 + y_2) \land (z = u_1 \cdot u_2) \\ &\qquad\qquad\qquad \to (z = (x_1 + y_1) \cdot (x_2 + y_2)). \end{aligned}$$

Clearly, our fragment is sufficient for that.

Interpretations and theories

When we speak about theories, it means that we want to restrict the interpretation (meaning) of some symbols in the language.

There are two main approaches how to do that

- axiomatic — we restrict the interpretations indirectly by providing axioms that have to be satisfied,
  - e.g., the equality axioms,
  - axioms usually require quantifiers,
- restricting interpretations — we allow only such classes of interpretations that correspond to our intended meaning,
  - e.g., we say that all the variables range over integers.

Note that some theories do not have appropriate axiomatic systems.

Theory $\mathcal{T}$

A theory $\mathcal{T}$ is given by a first-order language L.

We say that an interpretation $\mathcal{M} = (D, i)$ for L is a $\mathcal{T}$-interpretation if

- $\mathcal{M}$ satisfies all axioms of $\mathcal{T}$, or
- i admits only intended interpretations of $\mathcal{T}$.

We say that a formula φ is

- $\mathcal{T}$-satisfiable, if $\mathcal{M} \models \varphi$ for a $\mathcal{T}$-interpretation $\mathcal{M}$;
- $\mathcal{T}$-valid, if $\mathcal{M} \models \varphi$ for every $\mathcal{T}$-interpretation $\mathcal{M}$.

A set of formulae Γ $\mathcal{T}$-entails a formula φ, denoted $\Gamma \models_{\mathcal{T}} \varphi$, if every $\mathcal{T}$-interpretation satisfying all formulae in Γ satisfies also φ.

Example

If $\mathcal{T}$ is real arithmetic, then D are real numbers and $i(\leq)$, $i(+)$, ... have their standard meanings.

Satisfiability Modulo Theories (SMT)

We have a formula that has a propositional structure, but propositional variables are expressions in a theory $\mathcal{T}$.

Example

From

$$(x = 0 \vee x = 1) \wedge (x + y + z \neq 0) \wedge (f(y) > f(z)),$$

we obtain

$$(p \vee q) \wedge \neg r \wedge s,$$

by so-called propositional abstraction, where p is $x = 0$, q is $x = 1$, r is $x + y + z = 0$, and s is $f(y) > f(z)$.

Bibliography I

Biere, Armin, Marijn Heule, et al., eds. (2021). *Handbook of Satisfiability*. 2nd. Vol. 336. Frontiers in Artificial Intelligence and Applications. Washington: IOS Press. ISBN: 978-1-64368-161-0.

Biere, Armin, Marijn J. H. Heule, et al., eds. (Feb. 2009). *Handbook of Satisfiability*. Vol. 185. Frontiers in Artificial Intelligence and Applications. IOS Press, p. 980. ISBN: 978-1-58603-929-5.