

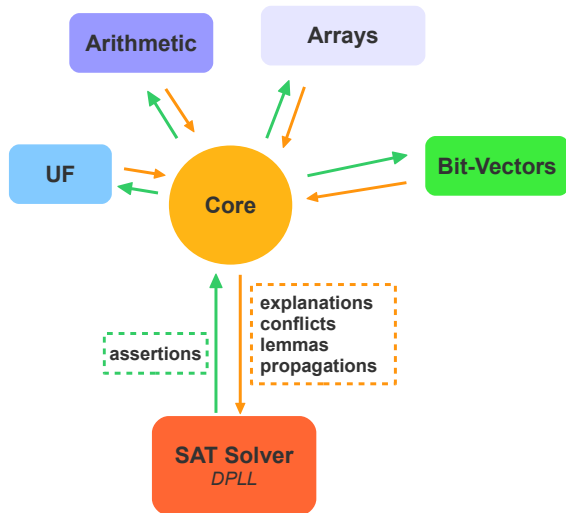
Logical reasoning and programming

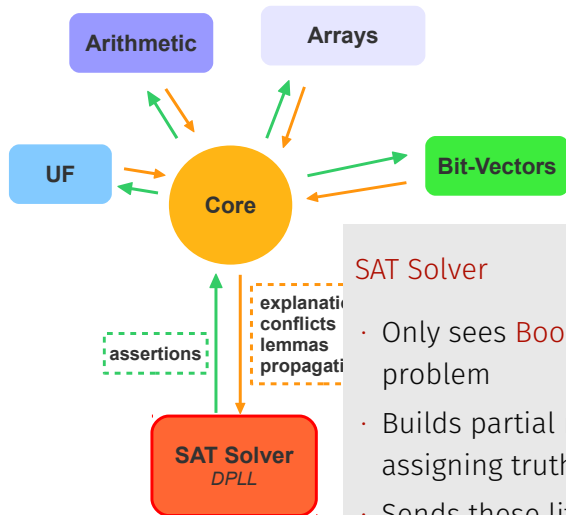
SMT (cont'd) and quantifiers in FOL

Karel Chvalovský

CIIRC CTU

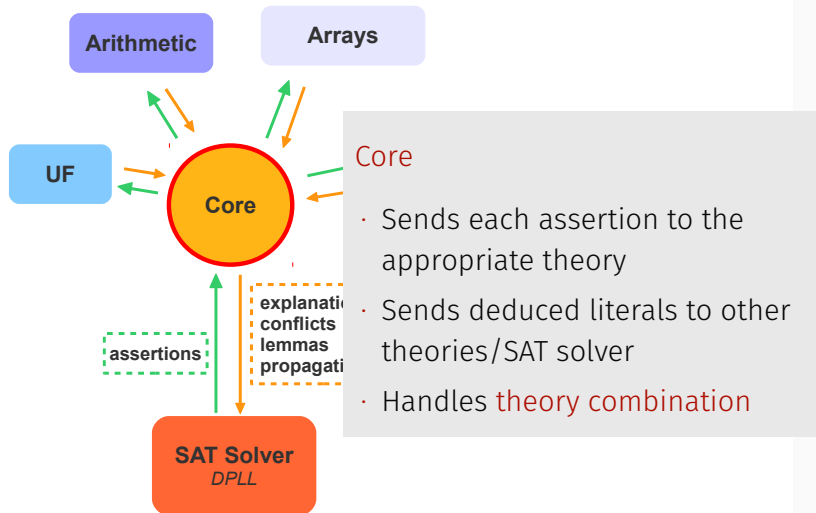
Parts of this presentation are significantly based on materials from recent SAT/SMT Summer Schools and SC² Summer School.

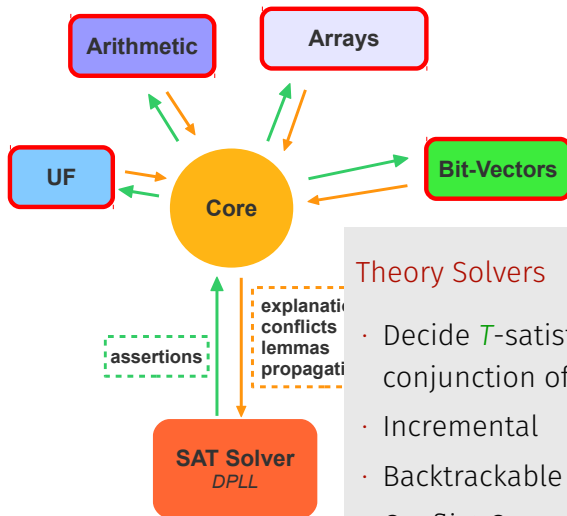




SAT Solver

- Only sees **Boolean skeleton** of problem
- Builds partial model by assigning truth values to literals
- Sends these literals to the core as **assertions**





Theory Solvers

- Decide T -satisfiability of a conjunction of theory literals
- Incremental
- Backtrackable
- Conflict Generation
- Theory Propagation

Combining theories

Instead of solving satisfiability of a formula φ in a theory \mathcal{T} , we can also try to solve the same problem for a theory that is created as a union of more theories $\mathcal{T}_1 \cup \dots \cup \mathcal{T}_n$, where we already have solvers for all individual \mathcal{T}_i , for $1 \leq i \leq n$. For example, we can combine \mathcal{T}_{LRA} and \mathcal{T}_{UF} .

We could develop a special solver for the new theory $\mathcal{T}_1 \cup \dots \cup \mathcal{T}_n$, or attempt to combine individual solvers together in a uniform way by

- ▶ separating reasoning for individual theories (purification),
- ▶ exchanging entailed equalities between solvers (equality propagation).

If individual theories remain satisfiable after exchanging all entailed equalities between solvers (propagation), then the combination is also satisfiable, otherwise it is unsatisfiable.

MOTIVATING EXAMPLE (CONVEX CASE)

Consider the following set of literals over $T_{\text{LRA}} \cup T_{\text{UF}}$
(T_{LRA} , linear real arithmetic):

$$\begin{aligned}f(f(x) - f(y)) &= a \\f(0) &> a + 2 \\x &= y\end{aligned}$$

MOTIVATING EXAMPLE (CONVEX CASE)

Consider the following set of literals over $T_{\text{LRA}} \cup T_{\text{UF}}$
(T_{LRA} , linear real arithmetic):

$$\begin{aligned}f(f(x) - f(y)) &= a \\f(0) &> a + 2 \\x &= y\end{aligned}$$

First step: *purify* literals so that each belongs to a single theory

MOTIVATING EXAMPLE (CONVEX CASE)

Consider the following set of literals over $T_{\text{LRA}} \cup T_{\text{UF}}$
(T_{LRA} , linear real arithmetic):

$$\begin{aligned}f(f(x) - f(y)) &= a \\f(0) &> a + 2 \\x &= y\end{aligned}$$

First step: *purify* literals so that each belongs to a single theory

$$\begin{aligned}f(f(x) - f(y)) = a &\implies f(e_1) = a &&\implies f(e_1) = a \\e_1 = f(x) - f(y) &&&e_1 = e_2 - e_3 \\&&&e_2 = f(x) \\&&&e_3 = f(y)\end{aligned}$$

MOTIVATING EXAMPLE (CONVEX CASE)

Consider the following set of literals over $T_{\text{LRA}} \cup T_{\text{UF}}$
(T_{LRA} , linear real arithmetic):

$$\begin{aligned}f(f(x) - f(y)) &= a \\f(0) &> a + 2 \\x &= y\end{aligned}$$

First step: *purify* literals so that each belongs to a single theory

$$\begin{aligned}f(0) > a + 2 &\implies f(e_4) > a + 2 &\implies f(e_4) = e_5 \\e_4 = 0 &&e_4 = 0 \\&&e_5 > a + 2\end{aligned}$$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	
$x = y$	

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	
$x = y$	

$$L_1 \models_{\text{UF}} e_2 = e_3$$

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	

$$L_2 \models_{\text{LRA}} e_1 = e_4$$

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	
$e_1 = e_4$	

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	
$e_1 = e_4$	

$$L_1 \models_{\text{UF}} a = e_5$$

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	$a = e_5$
$e_1 = e_4$	

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	$a = e_5$
$e_1 = e_4$	

Third step: check for

satisfiability locally

$$L_1 \not\models_{\text{UF}} \perp$$

$$L_2 \models_{\text{LRA}} \perp$$

MOTIVATING EXAMPLE (CONVEX CASE)

Second step: exchange entailed *interface equalities*, equalities over shared constants $e_1, e_2, e_3, e_4, e_5, a$

L_1	L_2
$f(e_1) = a$	$e_2 - e_3 = e_1$
$f(x) = e_2$	$e_4 = 0$
$f(y) = e_3$	$e_5 > a + 2$
$f(e_4) = e_5$	$e_2 = e_3$
$x = y$	$a = e_5$
$e_1 = e_4$	

Third step: check for

satisfiability locally

$$L_1 \not\models_{\text{UF}} \perp$$

$$L_2 \models_{\text{LRA}} \perp$$

Report **unsatisfiable**

Nelson–Oppen approach

A theory \mathcal{T} is stably infinite, if every \mathcal{T} -satisfiable ground formula is \mathcal{T} -satisfied by an infinite \mathcal{T} -interpretation. For example, finite structures like (QF_BV) are not stably infinite.

A theory \mathcal{T} is convex, if \mathcal{T} is a finite set of literals and if $\Gamma \models_{\mathcal{T}} \varphi_1 \vee \dots \vee \varphi_n$, then $\Gamma \models_{\mathcal{T}} \varphi_i$ for some $i \in \{1, \dots, n\}$. For example, (QF_UF) and (QF_LRA) are convex, but (QF_LIA), (QF_AX), and (QF_BV) are not convex.

It is possible to combine theories (Nelson–Oppen method) that are

- ▶ signature-disjoint (equalities are shared),
- ▶ stably infinite, and
- ▶ convex.

It is even possible to combine non-convex theories by propagating disjunctions of equalities (splitting). From practical point of view, many optimizations are required. There are also other methods, e.g., model-based theory combinations.

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Consider the following **unsatisfiable** set of literals over $T_{LIA} \cup T_{UF}$ (T_{LIA} , linear **integer** arithmetic):

$$1 \leq x \leq 2$$

$$f(1) = a$$

$$f(2) = f(1) + 3$$

$$a = b + 2$$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Consider the following **unsatisfiable** set of literals over $T_{LIA} \cup T_{UF}$ (T_{LIA} , linear **integer** arithmetic):

$$1 \leq x \leq 2$$

$$f(1) = a$$

$$f(2) = f(1) + 3$$

$$a = b + 2$$

First step: *purify* literals so that each belongs to a single theory

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Consider the following **unsatisfiable** set of literals over $T_{LIA} \cup T_{UF}$ (T_{LIA} , linear **integer** arithmetic):

$$1 \leq x \leq 2$$

$$f(1) = a$$

$$f(2) = f(1) + 3$$

$$a = b + 2$$

First step: *purify* literals so that each belongs to a single theory

$$f(1) = a \implies \begin{aligned} f(e_1) &= a \\ e_1 &= 1 \end{aligned}$$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Consider the following **unsatisfiable** set of literals over $T_{LIA} \cup T_{UF}$ (T_{LIA} , linear **integer** arithmetic):

$$1 \leq x \leq 2$$

$$f(1) = a$$

$$f(2) = f(1) + 3$$

$$a = b + 2$$

First step: *purify* literals so that each belongs to a single theory

$$f(2) = f(1) + 3 \implies e_2 = 2$$

$$f(e_2) = e_3$$

$$f(e_1) = e_4$$

$$e_3 = e_4 + 3$$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

No more entailed equalities, but $L_1 \models_{LIA} x = e_1 \vee x = e_2$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Consider each case of $x = e_1 \vee x = e_2$ separately

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Case 1) $x = e_1$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_1$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_1$	

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_1$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_1$	

$L_2 \models_{\text{UF}} a = b$, which entails \perp when sent to L_1

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	
$e_3 = e_4 + 3$	
$a = e_4$	

Case 2) $x = e_2$

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_2$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_2$	

MOTIVATING EXAMPLE (NON-CONVEX CASE)

Second step: exchange entailed *interface equalities* over shared constants $x, e_1, a, b, e_2, e_3, e_4$

L_1	L_2
$1 \leq x$	$f(e_1) = a$
$x \leq 2$	$f(x) = b$
$e_1 = 1$	$f(e_2) = e_3$
$a = b + 2$	$f(e_1) = e_4$
$e_2 = 2$	$x = e_2$
$e_3 = e_4 + 3$	
$a = e_4$	
$x = e_2$	

$L_2 \models_{\text{UF}} e_3 = b$, which entails \perp when sent to L_1

Why is SMT useful?

- ▶ it combines (not only) propositional logic reasoning with a domain-specific reasoning in a modular way,
- ▶ it covers commonly used theories and their combinations, for example, in software verification, we reason about
 - ▶ equalities,
 - ▶ arithmetic,
 - ▶ data structures,
- ▶ many applications in
 - ▶ scheduling,
 - ▶ test generation,
 - ▶ symbolic software verification,
 - ▶ static analysis,
 - ▶ program verification,
 - ▶ hardware verification,
- ▶ used by major companies like Microsoft (develops Z3), Amazon, ...

Quantifiers?

It is convenient to use \forall and \exists and modern SMT solvers can deal with quantifiers.

For some theories, it is even possible to eliminate quantifiers.

However, in SMT, it usually means that instances are produced in an ad hoc way. . .

Here, we continue with a systematic approach how to treat quantifiers. Hence we want to discuss provability in the full FOL.

Example (group theory)

Assume we have axioms

$$\forall X(1 \cdot X = X)$$

$$\forall X(X^{-1} \cdot X = 1)$$

$$\forall X \forall Y \forall Z((X \cdot Y) \cdot Z = X \cdot (Y \cdot Z))$$

and we want to know whether

1. $\forall X(X \cdot 1 = X)$,
2. $\forall X(X \cdot X^{-1} = 1)$, and
3. $\forall X \forall Y(X \cdot Y = Y \cdot X)$

follow from them. These are tasks in which automated theorem provers usually outperform people.

Two types of occurrences of variables

There is no difference between

$$\forall X p(X) \quad \text{and} \quad \forall Y p(Y)$$

when it comes to their meaning. Hence they can be freely replaced in formulae.

However, there is clearly a difference between

$$p(X) \quad \text{and} \quad p(Y),$$

because if we replace $p(X)$ by $p(Y)$ in

$$p(X) \vee \neg p(Y),$$

then we get

$$p(Y) \vee \neg p(Y).$$

Free and bounded occurrences of variables

We distinguish two types of occurrences of variables in a formula φ

- ▶ free — not under a scope of a quantifier, denoted $FV(\varphi)$,
- ▶ bounded — under a scope of a quantifier, denoted $BV(\varphi)$.

$$FV(\varphi) = \begin{cases} \{X \mid X \text{ occurs in } \varphi\}, & \text{if } \varphi \text{ is atomic,} \\ FV(\psi), & \text{if } \varphi = \neg\psi, \\ FV(\psi) \cup FV(\chi), & \text{if } \varphi = \psi \circ \chi \text{ for } \circ \in \{\wedge, \vee, \rightarrow\}, \\ FV(\psi) \setminus \{X\}, & \text{if } \varphi = QX\psi \text{ for } Q \in \{\forall, \exists\}. \end{cases}$$

$$BV(\varphi) = \begin{cases} \emptyset, & \text{if } \varphi \text{ is atomic,} \\ BV(\psi), & \text{if } \varphi = \neg\psi, \\ BV(\psi) \cup BV(\chi), & \text{if } \varphi = \psi \circ \chi \text{ for } \circ \in \{\wedge, \vee, \rightarrow\}, \\ BV(\psi) \cup \{X\}, & \text{if } \varphi = QX\psi \text{ for } Q \in \{\forall, \exists\}. \end{cases}$$

It is possible that $FV(\varphi) \cap BV(\varphi) \neq \emptyset$.

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

- ▶ $\mathcal{M} \models p(t_1, \dots, t_n)[e]$ iff $(t_1^{\mathcal{M}}[e], \dots, t_n^{\mathcal{M}}[e]) \in i(p)$, where p is n -ary predicate symbol in L ,
- ▶ $\mathcal{M} \models (t_1 = t_2)[e]$ iff $(t_1^{\mathcal{M}}[e], t_2^{\mathcal{M}}[e]) \in \text{id}_D$, (in FOL with eq.)
- ▶ $\mathcal{M} \models (\neg\psi)[e]$ iff $\mathcal{M} \not\models \psi[e]$,
- ▶ $\mathcal{M} \models (\psi \rightarrow \chi)[e]$ 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 $\{\psi\} \models \varphi$.

Basic properties

Let φ , ψ , and χ be formulae such that $X \notin FV(\psi)$, then

- ▶ $\neg \forall X \varphi \equiv \exists X \neg \varphi$,
- ▶ $\neg \exists X \varphi \equiv \forall X \neg \varphi$,
- ▶ $\forall X \forall Y \varphi \equiv \forall Y \forall X \varphi$,
- ▶ $\exists X \exists Y \varphi \equiv \exists Y \exists X \varphi$,
- ▶ $\forall X (\varphi \wedge \chi) \equiv \forall X \varphi \wedge \forall X \chi$,
- ▶ $\exists X (\varphi \vee \chi) \equiv \exists X \varphi \vee \exists X \chi$,
- ▶ $\exists X (\varphi \rightarrow \chi) \equiv \forall X \varphi \rightarrow \exists X \chi$,
- ▶ $(\psi \wedge \forall X \varphi) \equiv \forall X (\psi \wedge \varphi)$,
- ▶ $(\psi \wedge \exists X \varphi) \equiv \exists X (\psi \wedge \varphi)$,
- ▶ $(\psi \vee \forall X \varphi) \equiv \forall X (\psi \vee \varphi)$,
- ▶ $(\psi \vee \exists X \varphi) \equiv \exists X (\psi \vee \varphi)$,
- ▶ $(\psi \rightarrow \forall X \varphi) \equiv \forall X (\psi \rightarrow \varphi)$,
- ▶ $(\psi \rightarrow \exists X \varphi) \equiv \exists X (\psi \rightarrow \varphi)$,
- ▶ $(\forall X \varphi \rightarrow \psi) \equiv \exists X (\varphi \rightarrow \psi)$,
- ▶ $(\exists X \varphi \rightarrow \psi) \equiv \forall X (\varphi \rightarrow \psi)$.

Equivalent formulae

We can freely replace (sub)formulae by equivalent formulae. More formally

Lemma

Let ψ be a subformula of a formula φ , and χ be a formula such that $\psi \equiv \chi$. A formula φ' is obtained by replacing ψ in φ by χ . It holds that $\varphi \equiv \varphi'$.

Example

For example, this is useful for renaming bounded variables. Clearly $\forall X r(X, Y) \equiv \forall Z r(Z, Y)$ and hence $\forall X (p(X) \wedge \forall X r(X, Y))$ is equivalent to $\forall X (p(X) \wedge \forall Z r(Z, Y))$.

Note that $\forall X r(X, Y)$ is not equivalent to $\forall Y r(Y, Y)$!

How to decide whether $\Gamma \models \varphi$ for FOL using computers?

First, we know that this problem is undecidable. . .

However, we can still use our favorite recipe

1. show that it is sufficient to deal only with a restricted class of formulae by presenting various transformations and
(=clauses)
2. use techniques developed for less expressive systems
(=resolution)

to create a procedure that is quite useful.

Note that this is not the only possible approach! Moreover, the other approaches may have various advantages. Similarly, using only CNFs in propositional logic may lead to various problems.

Prenexing

We say that a formula φ is in prenex (normal) form, if

$$\varphi = Q_1 X_1, \dots, Q_n X_n \psi,$$

where Q_1, \dots, Q_n are quantifiers and ψ is an open (quantifier-free) formula.

Lemma

For every formula φ , there exists a formula ψ in prenex (normal) form such that $\varphi \equiv \psi$.

Proof.

By induction on the structure of the formula φ using previous equivalences and renaming of bounded variables. □

Substitutions

A substitution, $\sigma: Var \rightarrow Term$, is a function that assigns terms to variables. An application of a substitution σ on a formula φ , denoted $\varphi\sigma$, is a formula φ with all free occurrences of variables replaced simultaneously by their σ images. We usually denote substitutions σ , θ , and η .

Note that we usually provide only the non-identity part of a substitution. A substitution $\sigma: Var \rightarrow Var$ is called a *renaming*.

Example

Let $\sigma = \{X \mapsto f(X, Z), Y \mapsto a\}$, then

$$((p(X, Y) \rightarrow q(Y)) \vee (\forall V r(V, X)))\sigma$$

is

$$((p(f(X, Z), a) \rightarrow q(a)) \vee (\forall V r(V, f(X, Z))))).$$

Substitutability

A term t is *substitutable* into a formula φ for a variable X , if no occurrence of a variable in t becomes bounded in φ when all free occurrences of X in φ are replaced by t .

This directly extends to substitutions. From now on, we assume that if we apply a substitution, it is substitutable. However, we can always avoid all these potential problems by renaming bounded variables appropriately.

Example

Let $\sigma = \{X \mapsto f(X, Y), Y \mapsto g(X), Z \mapsto g(X)\}$, then

- ▶ $(\forall Z p(X, Y, Z))\sigma = \forall Z p(f(X, Y), g(X), Z)$, and
- ▶ $(\forall Y p(X, Y, Z))\sigma$ is not substitutable, but
- ▶ $(\forall U p(X, U, Z))\sigma = \forall U (f(X, Y), U, g(X))$.

Sentences

A term is ground (or closed), if it contains no variables. A formula φ is a sentence (or closed), if it contains no free occurrences of variables. A formula φ is open, if it contains no quantifiers.

Lemma

Let φ be a sentence, σ be a substitution, \mathcal{M} be an interpretation, and e be an evaluation, then

1. $\varphi\sigma = \varphi$,
2. $\mathcal{M} \models \varphi[e]$ iff $\mathcal{M} \models \varphi[e']$ for every evaluation e' ,
3. $\mathcal{M} \models \varphi$ or $\mathcal{M} \models \neg\varphi$.

Example

$p(a)$ and $\forall X\forall Y(p(X, b, g(Y, X)) \rightarrow q(f(f(b)), X))$ are sentences.
 $\forall Y(p(X, b, g(Y, X)) \rightarrow q(f(f(b)), X))$ is not a sentence.

Skolem functions

It is possible to get rid of existential quantifiers by introducing Skolem functions (or Skolem constants) that behave as witnesses (or choice functions).

We know that

$$\exists X \forall Y \exists Z p(X, Y, Z) \quad (1)$$

follows from

$$\forall Y p(c, Y, f(Y)) \quad (2)$$

where c and $f/1$ are fresh. Although (2) does not follow from (1), they are equisatisfiable.

Skolemization

We say that a formula is in Skolem normal form if it is in prenex normal form and it contains no existential quantifiers.

We can obtain a formula in Skolem normal form from a formula φ in prenex normal form by eliminating the first existential quantifier in

$$\varphi = \forall X_1 \dots \forall X_n \exists Y \psi.$$

We obtain

$$\varphi' = \forall X_1 \dots \forall X_n \psi\{Y \mapsto f(X_1, \dots, X_n)\}$$

where f is a fresh n -ary function symbol. Then we repeat the whole process with φ' until there is no existential quantifier in the formula. The resulting formula is equisatisfiable with φ .

We prefer Skolem functions with smaller arities.

Usual transformations

NNF (negation normal form)

Apply the following rewriting steps as long as possible:

$$\begin{array}{lcl} \neg\neg\varphi & \rightsquigarrow & \varphi \\ \varphi \rightarrow \psi & \rightsquigarrow & \neg\varphi \vee \psi \\ \neg(\varphi \wedge \psi) & \rightsquigarrow & \neg\varphi \vee \neg\psi \\ \neg(\varphi \vee \psi) & \rightsquigarrow & \neg\varphi \wedge \neg\psi \\ \neg(\forall X\varphi) & \rightsquigarrow & \exists X\neg\varphi \\ \neg(\exists X\varphi) & \rightsquigarrow & \forall X\neg\varphi \end{array}$$

Rectified formulae

A formula φ is rectified if

- ▶ no variable occurs both free and bounded in φ ,
- ▶ no two quantifiers in φ quantify over the same variable.

We obtain a rectified formula by renaming bounded variables.

Clauses in FOL

We adapt our terminology from propositional logic.

A *literal* is an atomic formula (positive), or a negation of an atomic formula (negative).

A *clause* is a disjunction of finitely many literals. An important special case is the empty clause, denoted \square .

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

Recall two special cases:

- ▶ the empty clause \square (empty disjunction) is unsatisfiable,
- ▶ the empty conjunction is satisfiable.

CNF

The universal closure of a formula φ , denoted $\forall\varphi$, is a formula $\forall X_1 \dots \forall X_n \varphi$, where $\{X_1, \dots, X_n\} = FV(\varphi)$.

We produce a CNF $\text{cnf}(\varphi)$ (implicitly universally quantified) from a sentence φ by performing the following steps

1. produce a NNF,
2. rectify,
3. skolemize (an obvious generalization for sentences not in prenex normal form),
4. remove all universal quantifiers,
5. produce a CNF as in propositional logic.

Let $\text{cnf}(\varphi) = \chi_1 \wedge \dots \wedge \chi_m$, where χ_i are clauses. It holds that

φ is satisf. iff $\forall \text{cnf}(\varphi)$ is satisf. iff $\forall \chi_1 \wedge \dots \wedge \forall \chi_m$ is satisf.

Our problem

Let $\Gamma = \{\psi_1, \dots, \psi_n\}$ be a set of sentences and φ be a sentence.
We know that

$$\Gamma \models \varphi$$

iff

$$\{\psi_1, \dots, \psi_n\} \cup \{\neg\varphi\} \text{ is unsatisfiable}$$

iff

$$\forall \bigwedge \text{cnf}(\psi_1 \wedge \dots \wedge \psi_n \wedge \neg\varphi) \text{ is unsatisfiable,}$$

iff

$$\forall \chi_1 \wedge \dots \wedge \forall \chi_m \text{ is unsatisfiable,}$$

where $\text{cnf}(\psi_1 \wedge \dots \wedge \psi_n \wedge \neg\varphi) = \{\chi_1, \dots, \chi_m\}$. (=a set of clauses).

From now on, we always assume that **a set of clauses is implicitly universally quantified.**

Instances

Lemma

Let χ be a clause and σ be a substitution, then $\forall\chi \models \chi\sigma$.

We say that $\chi\sigma$ is an *instance* of χ . If an instance contains no variable, then we call it a *ground instance*.

Example

From $\forall X\forall Y(p(X) \vee \neg q(X, Y))$, for example, follows $p(a) \vee \neg q(a, f(Z))$ and $p(b) \vee \neg q(b, f(a))$ (a ground instance).

Herbrand models

We can restrict the types of interpretations (and ground instances) that have to be considered. Let Γ be a set of clauses.

Herbrand universe

The Herbrand universe of Γ , denoted $HU(\Gamma)$, is the set of all ground terms in the language of Γ . If Γ contains no constants, we add a fresh constant c to the language.

Herbrand base

The Herbrand base of Γ , denoted $HB(\Gamma)$, is the set of all ground atomic formulae in the language of Γ , where only terms from $HU(\Gamma)$ are allowed.

Herbrand interpretation

A Herbrand interpretation of Γ is a subset of $HB(\Gamma)$.

Herbrand model

A Herbrand model \mathcal{M} of Γ is a Herbrand interpretation of Γ s.t. $\mathcal{M} \models \Gamma$.

Example

$\Gamma = \{\neg p(X, Y), q(f(Y), X)\}$. $HU(\Gamma) = \{c, f(c), f(f(c)), \dots\}$ and $HB(\Gamma) = \{p(c, c), p(c, f(c)), \dots, q(f(c), c), \dots\}$.

Herbrand's theorem

Theorem

Let Γ be a set of clauses. The following conditions are equivalent:

- 1. Γ is unsatisfiable,*
- 2. the set of all ground instances of Γ is unsatisfiable,*
- 3. a finite subset of the set of all ground instances of Γ is unsatisfiable.*

Note that Γ has a model iff it has a Herbrand model. However, we should note that clauses are quantifier-free. For example, a formula $p(c) \wedge \exists X \neg p(X)$ is clearly satisfiable, but has no Herbrand model; the Herbrand universe contains only c .

It is even possible to use so called Herbrand semantics, which is common in logic programming, instead of Tarskian semantics, check, for example, The Herbrand Manifesto.

Naïve approach

Herbrand's theorem provides a propositional criterion for unsatisfiability of a set of clauses Γ , because a ground atomic formula can be seen as a propositional atom (like in SMT).

Several early approaches (Gilmore; David and Putnam in 1960) work as follows

- ▶ generate ground instances and use propositional resolution,
- ▶ if it is propositionally satisfiable, then produce more instances (there is usually infinitely many of them) and repeat.

However, such an approach is generally very inefficient.

But variants of it are widely used, for example, in

- ▶ iProver,
 - ▶ EPR (effectively propositional, or Bernays–Schönfinkel–Ramsey class)—no function symbols and a quantifier prefix $\exists^* \forall^*$ hence $|D|$ is bounded by the number of constants occurring in the problem; decidable (NEXPTIME-complete).
- ▶ SMT,
- ▶ Answer Set Programming.

Lifting lemma

A technique to prove completeness theorems for the non-ground case using completeness for a ground instance.

For example, we want to satisfy two clauses

$$\{q(Y, f(X)), p(X, g(a, Y))\} \text{ and } \{\neg p(U, V), r(U, V)\}.$$

We want to represent infinitely many ground instances and possible resolution steps by a single non-ground instance.

$$\frac{\{q(Y, f(X)), p(X, g(a, Y))\} \quad \{\neg p(U, V), r(U, V)\}}{\{q(Y, f(X)), r(X, g(a, Y))\}}$$

Use unification!



Remark

$\{q(Y, f(X)), p(X, g(a, Y))\}$ means $\forall X \forall Y (q(Y, f(X)) \vee p(X, g(a, Y)))$
and $\{\neg p(U, V), r(U, V)\}$ means $\forall U \forall V (\neg p(U, V) \vee r(U, V))$.

Used presentation

The slides 1, 3, and 5 are taken from Tinelli 2017.

Bibliography I

-  Robinson, John Alan and Andrei Voronkov, eds. (2001). *Handbook of Automated Reasoning*. Vol. 1. Elsevier Science.
-  Tinelli, Cesare (2017). “Foundations of Satisfiability Modulo Theories”. SC² Summer School 2017. URL: <http://www.sc-square.org/CSA/school/lectures/SCSC-Tinelli.pdf>.