SAT and SMT Solving

Sarah Winkler

KRDB
Department of Computer Science
Free University of Bozen-Bolzano

lecture 5
WS 2022

Outline

- Summary of Last Week
- Satisfiability Modulo Theories
- DPLL(T)
- Using SMT Solvers with Theories

Definitions

for unsatisfiable CNF formula φ given as set of clauses

- $\psi \subseteq \varphi$ such that $\bigwedge_{C \in \psi} C$ is unsatisfiable is unsatisfiable core (UC) of φ
- minimal unsatisfiable core ψ is UC such that every subset of ψ is satisfiable
- SUC (minimum unsatisfiable core) is UC such that $|\psi|$ is minimal

Remark

SUC is always minimal unsatisfiable core

Definition (Resolution Graph)

directed acyclic graph $G = (V, E)$ is resolution graph for set of clauses φ if

1. $V = V_i \uplus V_c$ is set of clauses and $V_i = \varphi$,
2. V_i nodes have no incoming edges,
3. there is exactly one node $\square$ without outgoing edges,
4. $\forall C \in V_c \; \exists$ edges $D \to C$, $D' \to C$ such that C is resolvent of D and D', and
5. there are no other edges.

Algorithm minUnsatCore(φ)

Input: unsatisfiable formula φ
Output: minimal unsatisfiable core of φ

build resolution graph $G = (V_i \uplus V_c, E)$ for φ
while $\exists$ unmarked clause in V_i **do**
    $C \leftarrow$ unmarked clause in V_i
    if SAT($\overline{Reach_G(C)}$) **then** ▷ subgraph without C satisfiable?
        mark C ▷ C is UC member
    else
        build resolution graph $G' = (V_i' \uplus V_c', E')$ for $\overline{Reach_G(C)}$
        $V_i \leftarrow V_i \setminus \{C\}$ and $V_c \leftarrow V_c' \cup (V_c \setminus Reach_G(C))$
        $E \leftarrow E' \cup (E \setminus Reach_G^E(C))$
        $G \leftarrow (V_i \cup V_c, E)$
        $G \leftarrow G|_{BReach_G(\Box)}$ ▷ restrict to nodes with path to $\Box$
return V_i

Theorem

if φ unsatisfiable then minUnsatCore(φ) is minimal unsatisfiable core of φ

Definition (Partial minUNSAT)

pminUNSAT(χ, φ) is minimal $|\psi|$ such that $\psi \subseteq \varphi$ and $\chi \wedge \bigwedge_{C \in \psi} \neg C$ satisfiable

---

Algorithm FuMalik(χ, φ)

Input: clause set φ and satisfiable clause set χ

$cost \leftarrow 0$
while $\neg$SAT$(\chi \cup \varphi)$ **do**
  $UC \leftarrow$ unsatCore$(\chi \cup \varphi)$ ▷ must be minimal
  $B \leftarrow \varnothing$
  for $C \in UC \cap \varphi$ **do** ▷ loop over soft clauses in core
    $b \leftarrow$ new blocking variable
    $\varphi \leftarrow \varphi \setminus \{C\} \cup \{C \vee b\}$
    $B \leftarrow B \cup \{b\}$
  $\chi \leftarrow \chi \cup \mathtt{CNF}(\sum_{b \in B} b = 1)$ ▷ cardinality constraint is hard
  $cost \leftarrow cost + 1$
return $cost$

Theorem

FuMalik$(\chi, \varphi) =$ pminUNSAT(χ, φ)

$|\varphi| =$ pminUNSAT$(\chi, \varphi) +$ pmaxSAT(χ, φ)

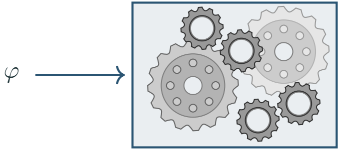
Outline

- Summary of Last Week
- Satisfiability Modulo Theories
  - Recap: First-Order Logic
  - Eager and Lazy Paradigms
- DPLL(T)
- Using SMT Solvers with Theories

SMT Solving

input: formula φ involving theory T

output:

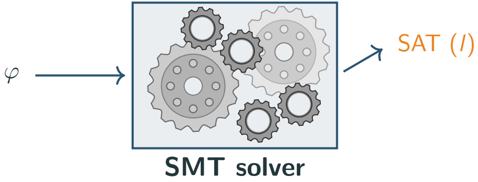


SMT solver

SMT Solving

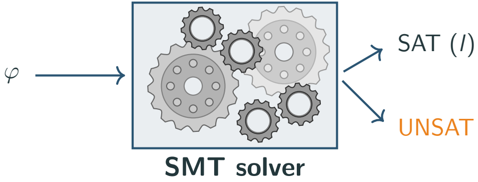
input: formula φ involving theory T

output: SAT + valuation v such that $v(\varphi) = T$ if φ is T-satisfiable



SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

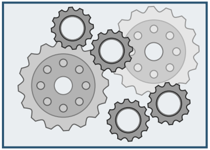


SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

φ $\longrightarrow$

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (I) $\quad v(a) = 3 \quad v(b) = 0 \quad v(c) = 0 \quad v(p) = \mathrm{T}$

UNSAT

Example (Common theories)

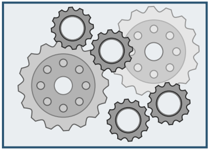
- arithmetic $\qquad 2a + b \geqslant c \vee (a - b = c + 3 \wedge p)$

SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

φ $\longrightarrow$

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

SAT (I) $v(a) = 3 \quad v(b) = 0$
$v(c) = 0 \quad v(p) = \mathrm{T}$

UNSAT

Example (Common theories)

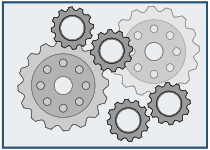
- arithmetic $2a + b \geqslant c \vee (a - b = c + 3 \wedge p)$
- uninterpreted functions $\mathrm{f}(x, y) \neq \mathrm{f}(y, x) \wedge \mathrm{g}(\mathrm{a}) = \mathrm{a} \to \mathrm{g}(\mathrm{f}(x, x)) = \mathrm{g}(y)$

SMT Solving

input:	formula φ involving theory T	
output:	SAT + valuation v such that $v(\varphi) = T$	if φ is T-satisfiable
	UNSAT	otherwise

φ →

$a + b \geqslant c \vee (a = 0 \wedge p)$



SMT solver

→ SAT (I) $\quad v(a) = 3 \quad v(b) = 0$

$\qquad\qquad v(c) = 0 \quad v(p) = \mathrm{T}$

→ UNSAT

Example (Common theories)

- arithmetic $\qquad 2a + b \geqslant c \vee (a - b = c + 3 \wedge p)$
- uninterpreted functions $\qquad \mathrm{f}(x, y) \neq \mathrm{f}(y, x) \wedge \mathrm{g}(\mathrm{a}) = \mathrm{a} \rightarrow \mathrm{g}(\mathrm{f}(x, x)) = \mathrm{g}(y)$
- bit vectors $\qquad ((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of

First-Order Logic: Syntax

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of
  - set of function symbols $\mathcal{F}$
  - set of predicate symbols $\mathcal{P}$

  where each symbol is associated with fixed arity (i.e., number of arguments)

First-Order Logic: Syntax

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of
  - set of function symbols $\mathcal{F}$
  - set of predicate symbols $\mathcal{P}$

  where each symbol is associated with fixed arity (i.e., number of arguments)
- function/predicate symbols with arity
  - 1 are called unary
  - 2 are called binary
  - 0 are called constants

First-Order Logic: Syntax

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of
  - set of function symbols $\mathcal{F}$
  - set of predicate symbols $\mathcal{P}$

  where each symbol is associated with fixed arity (i.e., number of arguments)
- function/predicate symbols with arity
  - 1 are called unary
  - 2 are called binary
  - 0 are called constants

Definitions (Formulas)

- Σ-terms t are built according to grammar

  $$t \quad ::= \quad x \mid c \mid f(\underbrace{t, \dots, t}_{n})$$

  for constant $c \in \mathcal{F}$, function symbol $f \in \mathcal{F}$ of arity $n > 0$, and variable $x \in X$

First-Order Logic: Syntax

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of
  - set of function symbols $\mathcal{F}$
  - set of predicate symbols $\mathcal{P}$

  where each symbol is associated with fixed arity (i.e., number of arguments)
- function/predicate symbols with arity
  - 1 are called unary
  - 2 are called binary
  - 0 are called constants

Definitions (Formulas)

- Σ-terms t are built according to grammar

$$t \quad ::= \quad x \mid c \mid f(\underbrace{t, \dots, t}_{n})$$

infinite set of variables X

for constant $c \in \mathcal{F}$, function symbol $f \in \mathcal{F}$ of arity $n > 0$, and variable $x \in X$

First-Order Logic: Syntax

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of
  - set of function symbols $\mathcal{F}$
  - set of predicate symbols $\mathcal{P}$

  where each symbol is associated with fixed arity (i.e., number of arguments)
- function/predicate symbols with arity
  - 1 are called unary
  - 2 are called binary
  - 0 are called constants

Definitions (Formulas)

- Σ-terms t are built according to grammar

  $$t \quad ::= \quad x \mid c \mid f(\underbrace{t, \dots, t}_{n})$$

  for constant $c \in \mathcal{F}$, function symbol $f \in \mathcal{F}$ of arity $n > 0$, and variable $x \in X$
- Σ-formulas are built according to grammar

  $$\varphi \quad ::= \quad Q \mid P(\underbrace{t, \dots, t}_{n}) \mid \bot \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall x.\, \varphi \mid \exists x.\, \varphi$$

  for constant $Q \in \mathcal{P}$, predicate symbol $P \in \mathcal{P}$ of arity $n > 0$, and Σ-terms t

First-Order Logic: Syntax

Definitions (Signature)

- signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of
  - set of function symbols $\mathcal{F}$
  - set of predicate symbols $\mathcal{P}$

  where each symbol is associated with fixed arity (i.e., number of arguments)
- function/predicate symbols with arity
  - 1 are called unary
  - 2 are called binary
  - 0 are called constants

Definitions (Formulas)

- Σ-terms t are built according to grammar

$$t \quad ::= \quad x \mid c \mid f(\underbrace{t, \dots, t}_{n})$$

  for constant $c \in \mathcal{F}$, function symbol $f \in \mathcal{F}$ of arity $n > 0$, and variable $x \in X$
- Σ-formulas are built according to grammar

$$\varphi \quad ::= \quad Q \mid P(\underbrace{t, \dots, t}_{n}) \mid \bot \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \forall x.\, \varphi \mid \exists x.\, \varphi$$

  for constant $Q \in \mathcal{P}$, predicate symbol $P \in \mathcal{P}$ of arity $n > 0$, and Σ-terms t
- variable x is free in φ if it is not bound by quantifier above

Notation
write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$

Notation
write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$

Notation

write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ {=}/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$

Notation
write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \wedge \mathsf{Q}$

Notation
write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \wedge \mathsf{Q}$
  - $\neg(\mathsf{P}(\mathsf{a}) \wedge \mathsf{P}(x) \wedge \mathsf{P}(y))$

Notation
write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \wedge \mathsf{Q}$
  - $\neg(\mathsf{P}(\mathsf{a}) \wedge \mathsf{P}(x) \wedge \mathsf{P}(y))$
  - $\exists x.\, \mathsf{P}(x)$

Notation

write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \wedge \mathsf{Q}$
  - $\neg(\mathsf{P}(\mathsf{a}) \wedge \mathsf{P}(x) \wedge \mathsf{P}(y))$
  - $\exists x.\, \mathsf{P}(x)$
  - $\mathsf{P}(x) \vee (\exists x.\, \mathsf{P}(x) \wedge \mathsf{f}(y) = x)$

Notation
write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \wedge \mathsf{Q}$
  - $\neg(\mathsf{P}(\mathsf{a}) \wedge \mathsf{P}(x) \wedge \mathsf{P}(y))$
  - $\exists x.\, \mathsf{P}(x)$
  - $\mathsf{P}(x) \vee (\exists x.\, \mathsf{P}(x) \wedge \mathsf{f}(y) = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$

Notation

write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas:
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \wedge \mathsf{Q}$
  - $\neg(\mathsf{P}(\mathsf{a}) \wedge \mathsf{P}(x) \wedge \mathsf{P}(y))$
  - $\exists x.\, \mathsf{P}(x)$
  - $\mathsf{P}(x) \vee (\exists x.\, \mathsf{P}(x) \wedge \mathsf{f}(y) = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$

write $\varphi \rightarrow \psi$ for $\neg\varphi \vee \psi$

Notation

write f/n or P/n to express that f or P have arity n

Example

- let $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ with $\mathcal{F} := \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1,\ \mathsf{g}/2\}$ and $\mathcal{P} := \{\mathsf{Q}/0,\ \mathsf{P}/1,\ =/2\}$
- the following are Σ-terms:
  - a, b, and $\mathsf{f}(\mathsf{a})$
  - x, y, and z — x, y, z are variables
  - $\mathsf{g}(\mathsf{a}, \mathsf{f}(x))$ and $\mathsf{g}(\mathsf{g}(\mathsf{a}, y), \mathsf{f}(\mathsf{b}))$
- the following are Σ-formulas (free variables highlighted):
  - $\mathsf{a} = \mathsf{f}(\mathsf{b})$
  - $\mathsf{P}(\mathsf{a}) \land \mathsf{Q}$
  - $\neg(\mathsf{P}(\mathsf{a}) \land \mathsf{P}(x) \land \mathsf{P}(y))$
  - $\exists x.\, \mathsf{P}(x)$
  - $\mathsf{P}(x) \lor (\exists x.\, \mathsf{P}(x) \land \mathsf{f}(y) = x)$
  - $\forall x\, y\, z.\, (x = y \land y = z \ \to\ x = z)$
  - $\forall x\, y.\, (x = y \ \to\ y = x)$

write $\varphi \to \psi$ for $\neg\varphi \lor \psi$

First-Order Logic: Semantics

Definition (Model)

model $\mathcal{M}$ for signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of

1. non-empty set A (universe of concrete values)
2. function $f^{\mathcal{M}} \colon A^n \to A$ for every n-ary $f \in \mathcal{F}$
3. set of n-tuples $P^{\mathcal{M}} \subseteq A^n$ for every n-ary $P \in \mathcal{P}$

Definition (Model)

model $\mathcal{M}$ for signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of

1. non-empty set A (universe of concrete values)
2. function $f^{\mathcal{M}} \colon A^n \to A$ for every n-ary $f \in \mathcal{F}$
3. set of n-tuples $P^{\mathcal{M}} \subseteq A^n$ for every n-ary $P \in \mathcal{P}$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$

First-Order Logic: Semantics

Definition (Model)

model $\mathcal{M}$ for signature $\Sigma = \langle \mathcal{F}, \mathcal{P} \rangle$ consists of

1. non-empty set A (universe of concrete values)
2. function $f^{\mathcal{M}}\colon A^n \to A$ for every n-ary $f \in \mathcal{F}$
3. set of n-tuples $P^{\mathcal{M}} \subseteq A^n$ for every n-ary $P \in \mathcal{P}$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$

Definitions

- environment for model $\mathcal{M} = \langle A, \{f^{\mathcal{M}}\}_{f \in \mathcal{F}}, \{P^{\mathcal{M}}\}_{P \in \mathcal{P}} \rangle$ is mapping $I \colon X \to A$

Definitions

- environment for model $\mathcal{M} = \langle A, \{f^{\mathcal{M}}\}_{f \in \mathcal{F}}, \{P^{\mathcal{M}}\}_{P \in \mathcal{P}} \rangle$ is mapping $I \colon X \to A$
- value $t^{\mathcal{M},I}$ of term t in model $\mathcal{M}$ wrt environment I is defined inductively:
$$t^{\mathcal{M},I} = \begin{cases} I(t) & \text{if } t \text{ is a variable} \\ f^{\mathcal{M}}(t_n^{\mathcal{M},I}, \dots, t_n^{\mathcal{M},I}) & \text{if } t = f(t_1, \dots, t_n) \end{cases}$$

Definitions

- environment for model $\mathcal{M} = \langle A, \{f^{\mathcal{M}}\}_{f \in \mathcal{F}}, \{P^{\mathcal{M}}\}_{P \in \mathcal{P}} \rangle$ is mapping $I \colon X \to A$
- value $t^{\mathcal{M},I}$ of term t in model $\mathcal{M}$ wrt environment I is defined inductively:

$$t^{\mathcal{M},I} = \begin{cases} I(t) & \text{if } t \text{ is a variable} \\ f^{\mathcal{M}}(t_n^{\mathcal{M},I}, \ldots, t_n^{\mathcal{M},I}) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

- for environment I, variable x and $a \in A$, extended environment $I[x \mapsto a]$ is

$$(I[x \mapsto a])(y) = \begin{cases} a & \text{if } x = y \\ I(y) & \text{otherwise} \end{cases}$$

Definitions

- environment for model $\mathcal{M} = \langle A, \{f^{\mathcal{M}}\}_{f \in \mathcal{F}}, \{P^{\mathcal{M}}\}_{P \in \mathcal{P}} \rangle$ is mapping $I\colon X \to A$
- value $t^{\mathcal{M},I}$ of term t in model $\mathcal{M}$ wrt environment I is defined inductively:

$$t^{\mathcal{M},I} = \begin{cases} I(t) & \text{if } t \text{ is a variable} \\ f^{\mathcal{M}}(t_n^{\mathcal{M},I}, \dots, t_n^{\mathcal{M},I}) & \text{if } t = f(t_1, \dots, t_n) \end{cases}$$

- for environment I, variable x and $a \in A$, extended environment $I[x \mapsto a]$ is

$$(I[x \mapsto a])(y) = \begin{cases} a & \text{if } x = y \\ I(y) & \text{otherwise} \end{cases}$$

- satisfaction relation $\mathcal{M} \models_I \varphi$ is defined inductively:

$$\mathcal{M} \models_I \varphi \quad \iff \quad \begin{cases} (t_n^{\mathcal{M},I}, \dots, t_n^{\mathcal{M},I}) \in P^{\mathcal{M}} & \text{if } \varphi = P(t_1, \dots, t_n) \\ \mathcal{M} \not\models_I \psi & \text{if } \varphi = \neg\psi \\ \mathcal{M} \models_I \varphi_1 \text{ and } \mathcal{M} \models_I \varphi_2 & \text{if } \varphi = \varphi_1 \wedge \varphi_2 \\ \mathcal{M} \models_I \varphi_1 \text{ or } \mathcal{M} \models_I \varphi_2 & \text{if } \varphi = \varphi_1 \vee \varphi_2 \\ \mathcal{M} \models_{I[x \mapsto a]} \psi \text{ for all } a \in A & \text{if } \varphi = \forall x.\, \psi \\ \mathcal{M} \models_{I[x \mapsto a]} \psi \text{ for some } a \in A & \text{if } \varphi = \exists x.\, \psi \end{cases}$$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$

$$\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$$
$$\mathsf{a}^{\mathcal{M}_1} = 0$$
$$\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1,\ \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1,\ \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} = \texttt{@elonmusk}$
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad \varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad \varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \rightarrow \mathsf{R}(x, z)$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} = \texttt{@elonmusk}$
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x.\mathsf{R}(x, a)$ $\varphi_2 = \forall x.\mathsf{R}(x, \mathsf{f}(x))$ $\varphi_3 = \forall x\, y\, z.\, \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1,\ \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \rightarrow \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$ $\mathcal{M}_1 \models_I \varphi_2$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} = \texttt{@elonmusk}$
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad\varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad\varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \rightarrow \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$ $\quad\mathcal{M}_1 \models_I \varphi_2$ $\quad\mathcal{M}_1 \models_I \varphi_3$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1,\ \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad \varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad \varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$ $\quad \mathcal{M}_1 \models_I \varphi_2$ $\quad \mathcal{M}_1 \models_I \varphi_3$

$\mathcal{M}_2 \models_I \varphi_1$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad\varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad\varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$ $\quad\mathcal{M}_1 \models_I \varphi_2$ $\quad\mathcal{M}_1 \models_I \varphi_3$

$\mathcal{M}_2 \models_I \varphi_1$ $\quad\mathcal{M}_2 \not\models_I \varphi_2$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad$ $\varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad$ $\varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$	$\mathcal{M}_1 \models_I \varphi_2$	$\mathcal{M}_1 \models_I \varphi_3$
$\mathcal{M}_2 \models_I \varphi_1$	$\mathcal{M}_2 \not\models_I \varphi_2$	$\mathcal{M}_2 \not\models_I \varphi_3$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} = \texttt{@elonmusk}$
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$ $\mathcal{M}_1 \models_I \varphi_2$ $\mathcal{M}_1 \models_I \varphi_3$

$\mathcal{M}_2 \models_I \varphi_1$ $\mathcal{M}_2 \not\models_I \varphi_2$ $\mathcal{M}_2 \not\models_I \varphi_3$

$\mathcal{M}_3 \models_I \varphi_1$

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad \varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad \varphi_3 = \forall x\, y\, z.\, \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$	$\mathcal{M}_1 \models_I \varphi_2$	$\mathcal{M}_1 \models_I \varphi_3$
$\mathcal{M}_2 \models_I \varphi_1$	$\mathcal{M}_2 \not\models_I \varphi_2$	$\mathcal{M}_2 \not\models_I \varphi_3$
$\mathcal{M}_3 \models_I \varphi_1$	$\mathcal{M}_3 \models_I \varphi_2$	

Example

function and predicate symbols $\mathcal{F} = \{\mathsf{f}/1, \mathsf{a}/0\}$ and $\mathcal{P} = \{\mathsf{R}/2\}$

1. model $\mathcal{M}_1$: universe $A_1 = \mathbb{N}$
   $\mathsf{f}^{\mathcal{M}_1}(x) = 2x + 1$
   $\mathsf{a}^{\mathcal{M}_1} = 0$
   $\mathsf{R}^{\mathcal{M}_1} = \{(x, y) \mid x < y\}$
2. model $\mathcal{M}_2$: universe A_2 is set of all Twitter users
   $\mathsf{f}^{\mathcal{M}_2}(x) =$ last person who started following x (or x if no follower)
   $\mathsf{a}^{\mathcal{M}_2} =$ `@elonmusk`
   $\mathsf{R}^{\mathcal{M}_2} = \{(x, y) \mid x \text{ follows } y\}$
3. model $\mathcal{M}_3$: universe A_3 is set of all days since year 2000
   $\mathsf{f}^{\mathcal{M}_3}(x)$ is day after x
   $\mathsf{a}^{\mathcal{M}_3} =$ "11.09.2001"
   $\mathsf{R}^{\mathcal{M}_3} = \{(x, y) \mid y \text{ is after } x\}$

$\varphi_1 = \exists x. \mathsf{R}(x, a)$ $\quad \varphi_2 = \forall x. \mathsf{R}(x, \mathsf{f}(x))$ $\quad \varphi_3 = \forall x\, y\, z.\ \mathsf{R}(x, y) \wedge \mathsf{R}(y, z) \to \mathsf{R}(x, z)$

$\mathcal{M}_1 \not\models_I \varphi_1$	$\mathcal{M}_1 \models_I \varphi_2$	$\mathcal{M}_1 \models_I \varphi_3$
$\mathcal{M}_2 \models_I \varphi_1$	$\mathcal{M}_2 \not\models_I \varphi_2$	$\mathcal{M}_2 \not\models_I \varphi_3$
$\mathcal{M}_3 \models_I \varphi_1$	$\mathcal{M}_3 \models_I \varphi_2$	$\mathcal{M}_3 \models_I \varphi_3$

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Definitions

for Σ-theory T, Σ-formulas φ and ψ and list of literals M:

- φ is T-satisfiable (or T-consistent) if $\varphi \cup \{T\}$ is satisfiable

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Definitions

for Σ-theory T, Σ-formulas φ and ψ and list of literals M:

- φ is T-satisfiable (or T-consistent) if $\varphi \cup \{T\}$ is satisfiable
- φ is T-unsatisfiable if not T-satisfiable

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Definitions

for Σ-theory T, Σ-formulas φ and ψ and list of literals M:

- φ is T-satisfiable (or T-consistent) if $\varphi \cup \{T\}$ is satisfiable
- φ is T-unsatisfiable if not T-satisfiable
- $M = l_1, \dots, l_k$ is T-satisfiable if $l_1 \wedge \cdots \wedge l_k$ is

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Definitions

for Σ-theory T, Σ-formulas φ and ψ and list of literals M:

- φ is T-satisfiable (or T-consistent) if $\varphi \cup \{T\}$ is satisfiable
- φ is T-unsatisfiable if not T-satisfiable
- $M = l_1, \dots, l_k$ is T-satisfiable if $l_1 \wedge \dots \wedge l_k$ is
- M is T-model of φ if $M \models \varphi$ and M is T-satisfiable

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Definitions

for Σ-theory T, Σ-formulas φ and ψ and list of literals M:

- φ is T-satisfiable (or T-consistent) if $\varphi \cup \{T\}$ is satisfiable
- φ is T-unsatisfiable if not T-satisfiable
- $M = l_1, \dots, l_k$ is T-satisfiable if $l_1 \wedge \cdots \wedge l_k$ is
- M is T-model of φ if $M \models \varphi$ and M is T-satisfiable
- φ entails ψ in T (denoted $\varphi \models_T \psi$) if $\varphi \wedge \neg\psi$ is T-unsatisfiable

Remark

- formula φ without free variables is called sentence
- if φ is sentence, $\mathcal{M} \models_I \varphi$ is independent of I, so simply write $\mathcal{M} \models \varphi$

Definition

- formula φ is satisfiable if $\mathcal{M} \models_I \varphi$ for some $\mathcal{M}$ and I
- set of formulas T is satisfiable if $\mathcal{M} \models_I \bigwedge_{\varphi \in T} \varphi$ for some $\mathcal{M}$ and I

Definition (Theory)

Σ-theory T is set of Σ-sentences that is satisfiable

Definitions

for Σ-theory T, Σ-formulas φ and ψ and list of literals M:

- φ is T-satisfiable (or T-consistent) if $\varphi \cup \{T\}$ is satisfiable
- φ is T-unsatisfiable if not T-satisfiable
- $M = l_1, \dots, l_k$ is T-satisfiable if $l_1 \wedge \cdots \wedge l_k$ is
- M is T-model of φ if $M \models \varphi$ and M is T-satisfiable
- φ entails ψ in T (denoted $\varphi \models_T \psi$) if $\varphi \wedge \neg\psi$ is T-unsatisfiable
- φ and ψ are T-equivalent (denoted $\varphi \equiv_T \psi$) if $\varphi \models_T \psi$ and $\psi \models_T \varphi$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ EQ-unsatisfiable

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ EQ-unsatisfiable
- $x = y \wedge y \neq z \models_{\mathsf{EQ}} z \neq x$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\ y.\ (x = y \rightarrow y = x)$
  - $\forall x\ y\ z.\ (x = y \wedge y = z \rightarrow x = z)$

Example

- $x = y \wedge y \neq z$ — EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ — EQ-unsatisfiable
- $x = y \wedge y \neq z \models_{\mathsf{EQ}} z \neq x$ — ✓

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ EQ-unsatisfiable
- $x = y \wedge y \neq z \models_{\mathsf{EQ}} z \neq x$ ✓
- $x = y \equiv_{\mathsf{EQ}} y = x$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ EQ-unsatisfiable
- $x = y \wedge y \neq z \models_{\mathsf{EQ}} z \neq x$ ✓
- $x = y \equiv_{\mathsf{EQ}} y = x$ ✓

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y \ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ EQ-unsatisfiable
- $x = y \wedge y \neq z \models_{\mathsf{EQ}} z \neq x$ ✓
- $x = y \equiv_{\mathsf{EQ}} y = x$ ✓
- $x = y \wedge y \neq z \equiv_{\mathsf{EQ}} z \neq x$

Definition (Theory of Equality EQ)

- signature: no function symbols, binary predicate $=$
- axioms:
  - $\forall x.\ (x = x)$
  - $\forall x\, y.\ (x = y\ \rightarrow\ y = x)$
  - $\forall x\, y\, z.\ (x = y \wedge y = z\ \rightarrow\ x = z)$

Example

- $x = y \wedge y \neq z$ EQ-satisfiable
- $x = y \wedge y \neq z \wedge (z = x \vee x = z)$ EQ-unsatisfiable
- $x = y \wedge y \neq z \models_{\mathsf{EQ}} z \neq x$ ✓
- $x = y \equiv_{\mathsf{EQ}} y = x$ ✓
- $x = y \wedge y \neq z \equiv_{\mathsf{EQ}} z \neq x$ ✗

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\,(x = x) \quad \forall x\,y.\,(x = y \;\rightarrow\; y = x) \quad \forall x\,y\,z.\,(x = y \wedge y = z \;\rightarrow\; x = z)$$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\, y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\, y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\, y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\ y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\ y\ z.\ (x = y \land y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \land \cdots \land x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \land \cdots \land x_n = y_n \ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) = \mathsf{a} \land \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\ y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\ y\ z.\ (x = y \land y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\ y_1\ \dots\ x_n\ y_n.\ (x_1 = y_1 \land \cdots \land x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\ y_1\ \dots\ x_n\ y_n.\ (x_1 = y_1 \land \cdots \land x_n = y_n \ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) = \mathsf{a} \land \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \land \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\, y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\, y\, z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

  plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \ldots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n))$$

  plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\, y_1\ \ldots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ (P(x_1, \ldots, x_n) \rightarrow P(y_1, \ldots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \wedge \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\, y.\ (x = y \ \to\ y = x) \quad \forall x\, y\, z.\ (x = y \land y = z \ \to\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \ldots\ x_n\, y_n.\ (x_1 = y_1 \land \cdots \land x_n = y_n \ \to\ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\, y_1\ \ldots\ x_n\, y_n.\ (x_1 = y_1 \land \cdots \land x_n = y_n \ \to\ (P(x_1, \ldots, x_n) \to P(y_1, \ldots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) = \mathsf{a} \land \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \land \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable
- $\mathsf{a} = \mathsf{b} \land \mathsf{f}(\mathsf{a}) \neq \mathsf{a} \models_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) \neq \mathsf{b}$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\ y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\ y\ z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

  plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\ y_1\ \dots\ x_n\ y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

  plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\ y_1\ \dots\ x_n\ y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \wedge \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{a} \models_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) \neq \mathsf{b}$ ✓

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\, y.\ (x = y \ \to\ y = x) \quad \forall x\, y\, z.\ (x = y \wedge y = z \ \to\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \to\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \to\ (P(x_1, \dots, x_n) \to P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \wedge \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{a} \models_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) \neq \mathsf{b}$ ✓
- $\mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{a}) \equiv_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{f}(\mathsf{a}))$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\ y.\ (x = y\ \rightarrow\ y = x) \quad \forall x\ y\ z.\ (x = y \wedge y = z\ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\ y_1\ \dots\ x_n\ y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n\ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\ y_1\ \dots\ x_n\ y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n\ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \wedge \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{a} \models_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) \neq \mathsf{b}$ ✓
- $\mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{a}) \equiv_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{f}(\mathsf{a}))$ ✓

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\ y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\ y\ z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\ y_1\ \ldots\ x_n\ y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

$$\forall x_1\ y_1\ \ldots\ x_n\ y_n.\ (x_1 = y_1 \wedge \cdots \wedge x_n = y_n \ \rightarrow\ (P(x_1, \ldots, x_n) \rightarrow P(y_1, \ldots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \wedge \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{a} \models_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) \neq \mathsf{b}$ ✓
- $\mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{a}) \equiv_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{f}(\mathsf{a}))$ ✓
- $\mathsf{P}(\mathsf{a}) \wedge \mathsf{a} \neq \mathsf{b} \models_{\mathsf{EQ}} \neg\mathsf{P}(\mathsf{b})$

Definition (Theory of Equality With Uninterpreted Functions EUF)

- signature: function symbols $\mathcal{F}$, predicate symbols $\mathcal{P}$ including binary $=$
- axioms:

$$\forall x.\ (x = x) \quad \forall x\ y.\ (x = y \ \rightarrow\ y = x) \quad \forall x\ y\ z.\ (x = y \wedge y = z \ \rightarrow\ x = z)$$

plus for all n-ary $f \in \mathcal{F}$:

$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \dots \wedge x_n = y_n \ \rightarrow\ f(x_1, \dots, x_n) = f(y_1, \dots, y_n))$$

plus for all n-ary $P \in \mathcal{P} \setminus \{=\}$:

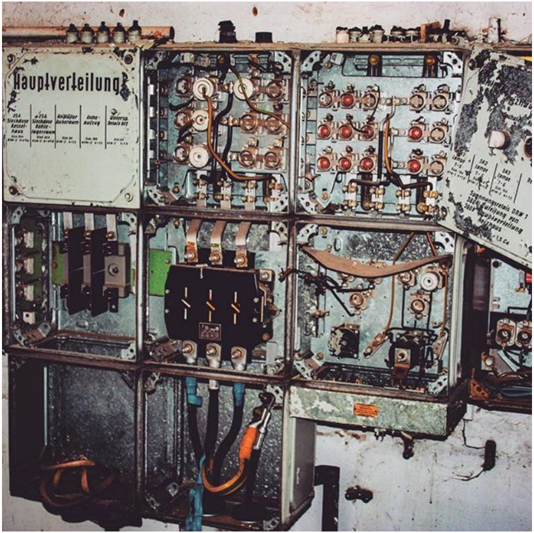
$$\forall x_1\, y_1\ \dots\ x_n\, y_n.\ (x_1 = y_1 \wedge \dots \wedge x_n = y_n \ \rightarrow\ (P(x_1, \dots, x_n) \rightarrow P(y_1, \dots, y_n)))$$

Example

for $\mathcal{F} = \{\mathsf{a}/0,\ \mathsf{b}/0,\ \mathsf{f}/1, \mathsf{g}/2\}$ and $\mathcal{P} = \{=/2, Q/1\}$

- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{g}(\mathsf{f}(\mathsf{a}), \mathsf{b}) \neq \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-unsatisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{b} \wedge \mathsf{g}(\mathsf{g}(\mathsf{b}, \mathsf{b}), \mathsf{b}) = \mathsf{g}(\mathsf{b}, \mathsf{b})$ UEQ-satisfiable
- $\mathsf{a} = \mathsf{b} \wedge \mathsf{f}(\mathsf{a}) \neq \mathsf{a} \models_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) \neq \mathsf{b}$ ✓
- $\mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{a}) \equiv_{\mathsf{UEQ}} \mathsf{f}(\mathsf{a}) = \mathsf{a} \wedge \mathsf{P}(\mathsf{f}(\mathsf{a}))$ ✓
- $\mathsf{P}(\mathsf{a}) \wedge \mathsf{a} \neq \mathsf{b} \models_{\mathsf{EQ}} \neg\mathsf{P}(\mathsf{b})$ ✗

Uninterpreted Functions in Real Life



Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathrm{read}(\mathrm{write}(A, i, v), j)$

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathrm{read}(\mathrm{write}(A, i, v), j)$
- bitvectors (BV) $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathrm{read}(\mathrm{write}(A, i, v), j)$
- bitvectors (BV) $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$
- strings $x \,@\, y = z \,@\, \mathrm{replace}(y, \mathtt{a}, \mathtt{b})$

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathsf{read}(\mathsf{write}(A, i, v), j)$
- bitvectors (BV) $((\mathsf{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$
- strings $x \mathbin{@} y = z \mathbin{@} \mathsf{replace}(y, \mathtt{a}, \mathtt{b})$
- ...
- their combinations

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x,a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathrm{read}(\mathrm{write}(A, i, v), j)$
- bitvectors (BV) $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$
- strings $x \mathbin{@} y = z \mathbin{@} \mathrm{replace}(y, \mathtt{a}, \mathtt{b})$
- ...
- their combinations

SMT-LIB

- language standard and benchmarks: `http://www.smt-lib.org`

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathrm{read}(\mathrm{write}(A, i, v), j)$
- bitvectors (BV) $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$
- strings $x \mathbin{@} y = z \mathbin{@} \mathrm{replace}(y, \mathtt{a}, \mathtt{b})$
- ...
- their combinations

SMT-LIB

- language standard and benchmarks: `http://www.smt-lib.org`
- annual solver competition: `http://www.smt-comp.org`

Theories of Interest in SMT Solvers

- equality + uninterpreted functions (EUF) $f(x, a) = g(y)$
- difference logic (DL) $x - y \leqslant 1$
- linear arithmetic $3x - 5y + 7z \leqslant 1$
  - over integers $\mathbb{Z}$ (LIA)
  - over reals $\mathbb{R}$ (LRA)
- arrays (A) $\mathrm{read}(\mathrm{write}(A, i, v), j)$
- bitvectors (BV) $((\mathrm{zext}_{32}\ a_8) + b_{32}) \times c_{32} >_u 0_{32}$
- strings $x \mathbin{@} y = z \mathbin{@} \mathrm{replace}(y, \mathtt{a}, \mathtt{b})$
- ...
- their combinations

SMT-LIB

- language standard and benchmarks: `http://www.smt-lib.org`
- annual solver competition: `http://www.smt-comp.org`
- solvers: Yices, OpenSMT, MathSAT, Z3, CVC4, Barcelogic, ...

The Eager Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

The Eager Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Approach 1: Eager SMT Solving

- use satisfiability-preserving transformation from T literals to SAT formula, ship one big formula to SAT solver

The Eager Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Approach 1: Eager SMT Solving

- use satisfiability-preserving transformation from T literals to SAT formula, ship one big formula to SAT solver
- requires sophisticated translation for each theory:
  done for EUF, difference logic, linear integer arithmetic, arrays

The Eager Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Approach 1: Eager SMT Solving

- use satisfiability-preserving transformation from T literals to SAT formula, ship one big formula to SAT solver
- requires sophisticated translation for each theory:
  done for EUF, difference logic, linear integer arithmetic, arrays
- still dominant approach for bit-vector arithmetic (known as "bit blasting")

The Eager Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Approach 1: Eager SMT Solving

- use satisfiability-preserving transformation from T literals to SAT formula, ship one big formula to SAT solver
- requires sophisticated translation for each theory:
  done for EUF, difference logic, linear integer arithmetic, arrays
- still dominant approach for bit-vector arithmetic (known as "bit blasting")
- advantage: use SAT solver off the shelf

The Eager Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Approach 1: Eager SMT Solving

- use satisfiability-preserving transformation from T literals to SAT formula, ship one big formula to SAT solver
- requires sophisticated translation for each theory:
  done for EUF, difference logic, linear integer arithmetic, arrays
- still dominant approach for bit-vector arithmetic (known as “bit blasting”)
- advantage: use SAT solver off the shelf
- drawbacks:
  - expensive translations: infeasible for large formulas
  - even more complicated with multiple theories

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables
   - obtain pure SAT formula, transform to CNF formula ψ

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables
   - obtain pure SAT formula, transform to CNF formula ψ
2. ship ψ to SAT solver

The Lazy Paradigm

Aim
given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea
use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables
   - obtain pure SAT formula, transform to CNF formula ψ
2. ship ψ to SAT solver
   - if ψ unsatisfiable, so is φ

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables
   - obtain pure SAT formula, transform to CNF formula ψ
2. ship ψ to SAT solver
   - if ψ unsatisfiable, so is φ
   - if ψ satisfiable by v, check v with T-solver:

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables
   - obtain pure SAT formula, transform to CNF formula ψ
2. ship ψ to SAT solver
   - if ψ unsatisfiable, so is φ
   - if ψ satisfiable by v, check v with T-solver:
     - if v is T-consistent then also φ is satisfiable

The Lazy Paradigm

Aim

given Σ-theory T and Σ-formula φ mixing propositional logic with symbols from Σ, determine T-satisfiability

Idea

use specialized T-solver that can deal with conjunction of theory literals

Approach 2: Lazy SMT Solving

1. abstract φ to propositional CNF:
   - "forget theory" by replacing T-literals with fresh propositional variables
   - obtain pure SAT formula, transform to CNF formula ψ
2. ship ψ to SAT solver
   - if ψ unsatisfiable, so is φ
   - if ψ satisfiable by v, check v with T-solver:
     - if v is T-consistent then also φ is satisfiable
     - otherwise T-solver generates T-consequence C of φ excluding v, repeat from 1 with $\varphi \wedge C$

Example

$$g(a) = c \land (\neg(f(g(a)) = f(c)) \lor g(a) = d) \land \neg(c = d\ )$$

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$

Example

$$\underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{c}}_{x_1} \wedge (\neg(\underbrace{\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c})}_{x_2}) \vee \underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{d}}_{x_3}) \wedge \neg(\underbrace{\mathsf{c} = \mathsf{d}}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$

Example

$$\underbrace{\mathsf{g(a) = c}}_{x_1} \land (\neg(\underbrace{\mathsf{f(g(a)) = f(c)}}_{x_2}) \lor \underbrace{\mathsf{g(a) = d}}_{x_3}) \land \neg(\underbrace{\mathsf{c = d}}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \land (\neg x_2 \lor x_3) \land \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - *T*-solver gets $\mathsf{g(a) = c} \land \mathsf{f(g(a))} \neq \mathsf{f(c)} \land \mathsf{c} \neq \mathsf{d}$

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$
   - T-unsatisfiable: $g(a) = c$ implies $f(g(a)) = f(c)$

Example

$$\underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{c}}_{x_1} \wedge (\neg(\underbrace{\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c})}_{x_2}) \vee \underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{d}}_{x_3}) \wedge \neg(\underbrace{\mathsf{c} = \mathsf{d}}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - T-solver gets $\mathsf{g}(\mathsf{a}) = \mathsf{c} \wedge \mathsf{f}(\mathsf{g}(\mathsf{a})) \neq \mathsf{f}(\mathsf{c}) \wedge \mathsf{c} \neq \mathsf{d}$
   - T-unsatisfiable: $\mathsf{g}(\mathsf{a}) = \mathsf{c}$ implies $\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c})$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathrm{T}$ and $v_1(x_2) = v_1(x_4) = \mathrm{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$
   - T-unsatisfiable: $g(a) = c$ implies $f(g(a)) = f(c)$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$

Example

$$\underbrace{\mathsf{g(a) = c}}_{x_1} \wedge (\neg\underbrace{(\mathsf{f(g(a)) = f(c)})}_{x_2} \vee \underbrace{\mathsf{g(a) = d}}_{x_3}) \wedge \neg\underbrace{(\mathsf{c = d}\,)}_{x_4}$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - T-solver gets $\mathsf{g(a) = c} \wedge \mathsf{f(g(a))} \neq \mathsf{f(c)} \wedge \mathsf{c} \neq \mathsf{d}$
   - T-unsatisfiable: $\mathsf{g(a) = c}$ implies $\mathsf{f(g(a)) = f(c)}$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$
2. satisfiable: $v_2(x_1) = v_2(x_2) = v_2(x_3) = \mathsf{T}$ and $v_2(x_4) = \mathsf{F}$

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$
   - T-unsatisfiable: $g(a) = c$ implies $f(g(a)) = f(c)$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$
2. satisfiable: $v_2(x_1) = v_2(x_2) = v_2(x_3) = \mathsf{T}$ and $v_2(x_4) = \mathsf{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) = f(c) \wedge g(a) = d \wedge c \neq d$

Example

$$\underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{c}}_{x_1} \wedge (\neg(\underbrace{\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c})}_{x_2}) \vee \underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{d}}_{x_3}) \wedge \neg(\underbrace{\mathsf{c} = \mathsf{d}}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - T-solver gets $\mathsf{g}(\mathsf{a}) = \mathsf{c} \wedge \mathsf{f}(\mathsf{g}(\mathsf{a})) \neq \mathsf{f}(\mathsf{c}) \wedge \mathsf{c} \neq \mathsf{d}$
   - T-unsatisfiable: $\mathsf{g}(\mathsf{a}) = \mathsf{c}$ implies $\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c})$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$
2. satisfiable: $v_2(x_1) = v_2(x_2) = v_2(x_3) = \mathsf{T}$ and $v_2(x_4) = \mathsf{F}$
   - T-solver gets $\mathsf{g}(\mathsf{a}) = \mathsf{c} \wedge \mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c}) \wedge \mathsf{g}(\mathsf{a}) = \mathsf{d} \wedge \mathsf{c} \neq \mathsf{d}$
   - T-unsatisfiable

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathrm{T}$ and $v_1(x_2) = v_1(x_4) = \mathrm{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$
   - T-unsatisfiable: $g(a) = c$ implies $f(g(a)) = f(c)$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$
2. satisfiable: $v_2(x_1) = v_2(x_2) = v_2(x_3) = \mathrm{T}$ and $v_2(x_4) = \mathrm{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) = f(c) \wedge g(a) = d \wedge c \neq d$
   - T-unsatisfiable
   - block valuation v_2 in future: add $\neg x_1 \vee \neg x_3 \vee x_4$

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathrm{T}$ and $v_1(x_2) = v_1(x_4) = \mathrm{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$
   - T-unsatisfiable: $g(a) = c$ implies $f(g(a)) = f(c)$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$
2. satisfiable: $v_2(x_1) = v_2(x_2) = v_2(x_3) = \mathrm{T}$ and $v_2(x_4) = \mathrm{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) = f(c) \wedge g(a) = d \wedge c \neq d$
   - T-unsatisfiable
   - block valuation v_2 in future: add $\neg x_1 \vee \neg x_3 \vee x_4$

1. $\psi_3 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$

Example

$$\underbrace{g(a) = c}_{x_1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{x_2}) \vee \underbrace{g(a) = d}_{x_3}) \wedge \neg(\underbrace{c = d}_{x_4})$$

1. abstract to propositional skeleton $\psi_1 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4$
2. satisfiable: $v_1(x_1) = \mathsf{T}$ and $v_1(x_2) = v_1(x_4) = \mathsf{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$
   - T-unsatisfiable: $g(a) = c$ implies $f(g(a)) = f(c)$
   - block valuation v_1 in future: add $\neg x_1 \vee x_2$

1. $\psi_2 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2)$
2. satisfiable: $v_2(x_1) = v_2(x_2) = v_2(x_3) = \mathsf{T}$ and $v_2(x_4) = \mathsf{F}$
   - T-solver gets $g(a) = c \wedge f(g(a)) = f(c) \wedge g(a) = d \wedge c \neq d$
   - T-unsatisfiable
   - block valuation v_2 in future: add $\neg x_1 \vee \neg x_3 \vee x_4$

1. $\psi_3 = x_1 \wedge (\neg x_2 \vee x_3) \wedge \neg x_4 \wedge (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4)$
2. unsatisfiable

Outline

- Summary of Last Week
- Satisfiability Modulo Theories
- DPLL(T)
- Using SMT Solvers with Theories

Approach

- most state-of-the-art SMT solvers use DPLL(T):
  lazy approach combining DPLL with theory propagation

Approach

- most state-of-the-art SMT solvers use DPLL(T):
  lazy approach combining DPLL with theory propagation
- advantages: not specific to theory, also extends to theory combinations

Approach

- most state-of-the-art SMT solvers use DPLL(T):
  lazy approach combining DPLL with theory propagation
- advantages: not specific to theory, also extends to theory combinations

Definition (DPLL(T) Transition Rules)

DPLL(T) consists of DPLL rules unit propagate, decide, fail, and restart plus

Approach

- most state-of-the-art SMT solvers use DPLL(T):
  lazy approach combining DPLL with theory propagation
- advantages: not specific to theory, also extends to theory combinations

Definition (DPLL(T) Transition Rules)

DPLL(T) consists of DPLL rules unit propagate, decide, fail, and restart plus

- T-backjump $\quad M\, l^d\, N \parallel F, C \quad \Longrightarrow \quad M\, l' \parallel F, C$
  if $M\, l^d\, N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models_T C' \vee l'$
  - $M \models \neg C'$ and l' is undefined in M, and l' or l'^c occurs in F or in $M\, l^d\, N$

Approach

- most state-of-the-art SMT solvers use DPLL(T):
  lazy approach combining DPLL with theory propagation
- advantages: not specific to theory, also extends to theory combinations

Definition (DPLL(T) Transition Rules)

DPLL(T) consists of DPLL rules unit propagate, decide, fail, and restart plus

- T-backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \models \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \models_T C' \vee l'$
  - $M \models \neg C'$ and l' is undefined in M, and l' or l'^c occurs in F or in $M\ l^d\ N$
- T-learn $\quad M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C$
  if $F \models_T C$ and all atoms of C occur in M or F

Approach

- most state-of-the-art SMT solvers use DPLL(T): lazy approach combining DPLL with theory propagation
- advantages: not specific to theory, also extends to theory combinations

Definition (DPLL(T) Transition Rules)

DPLL(T) consists of DPLL rules unit propagate, decide, fail, and restart plus

- T-backjump $\qquad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$

  if $M\ l^d\ N \vDash \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \vDash_T C' \vee l'$
  - $M \vDash \neg C'$ and l' is undefined in M, and l' or l'^c occurs in F or in $M\ l^d\ N$

- T-learn $\qquad M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C$

  if $F \vDash_T C$ and all atoms of C occur in M or F

- T-forget $\qquad M \parallel F,\ C \quad \Longrightarrow \quad M \parallel F$

  if $F \vDash_T C$

Approach

- most state-of-the-art SMT solvers use DPLL(T):
  lazy approach combining DPLL with theory propagation
- advantages: not specific to theory, also extends to theory combinations

Definition (DPLL(T) Transition Rules)

DPLL(T) consists of DPLL rules unit propagate, decide, fail, and restart plus

- T-backjump $\quad M\ l^d\ N \parallel F, C \quad \Longrightarrow \quad M\ l' \parallel F, C$
  if $M\ l^d\ N \vDash \neg C$ and $\exists$ clause $C' \vee l'$ such that
  - $F, C \vDash_T C' \vee l'$
  - $M \vDash \neg C'$ and l' is undefined in M, and l' or l'^c occurs in F or in $M\ l^d\ N$
- T-learn $\quad M \parallel F \quad \Longrightarrow \quad M \parallel F,\ C$
  if $F \vDash_T C$ and all atoms of C occur in M or F
- T-forget $\quad M \parallel F,\ C \quad \Longrightarrow \quad M \parallel F$
  if $F \vDash_T C$
- T-propagate $\quad M \parallel F \quad \Longrightarrow \quad M\ l \parallel F$
  if $M \vDash_T l$, literal l or l^c occurs in F, and l is undefined in M

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \dots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \ldots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use T-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \ldots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use T-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$
- apply restart

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \ldots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use T-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$
- apply restart

Improvement 1: Incremental T-Solver

- T-solver checks T-satisfiability of model M whenever literal is added to M

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \ldots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use T-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$
- apply restart

Improvement 1: Incremental T-Solver

- T-solver checks T-satisfiability of model M whenever literal is added to M

Improvement 2: On-Line SAT solver

- after T-learn added clause, apply fail or T-backjump instead of restart

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \ldots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use T-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$
- apply restart

Improvement 1: Incremental T-Solver

- T-solver checks T-satisfiability of model M whenever literal is added to M

Improvement 2: On-Line SAT solver

- after T-learn added clause, apply fail or T-backjump instead of restart

Improvement 3: Eager Theory Propagation

- apply T-propagate before decide

Simple Strategy using DPLL(T)

- whenever state $M \parallel F$ is final wrt unit propagate, decide, fail, T-backjump: check T-satisfiability of M with T-solver
- if M is T-consistent then T-satisfiability is proven
- otherwise $\exists l_1, \ldots, l_k$ subset of M such that $F \vDash_T \neg(l_1 \wedge \cdots \wedge l_k)$
- use T-learn to add $\neg l_1 \vee \cdots \vee \neg l_k$
- apply restart

Improvement 1: Incremental T-Solver

- T-solver checks T-satisfiability of model M whenever literal is added to M

Improvement 2: On-Line SAT solver

- after T-learn added clause, apply fail or T-backjump instead of restart

Improvement 3: Eager Theory Propagation

- apply T-propagate before decide

Remark

all three improvements can be combined

Example (Revisited with DPLL(T))

$$\underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{c}}_{1} \wedge (\neg(\underbrace{\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c})}_{2}) \vee \underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{d}}_{3}) \wedge \neg(\underbrace{\mathsf{c} = \mathsf{d}}_{4})$$

$$\| \; 1, \; (\overline{2} \vee 3), \; \overline{4}$$

Example (Revisited with DPLL(T))

$$\underbrace{g(a) = c}_{1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{2}) \vee \underbrace{g(a) = d}_{3}) \wedge \neg(\underbrace{c = d}_{4})$$

$$\| \; 1, \; (\overline{2} \vee 3), \; \overline{4}$$

$$\Longrightarrow \quad 1 \, \| \; 1, \; (\overline{2} \vee 3), \; \overline{4} \qquad \text{unit propagate}$$

Example (Revisited with DPLL(T))

$$\underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{c}}_{1} \wedge (\neg\underbrace{(\mathsf{f}(\mathsf{g}(\mathsf{a})) = \mathsf{f}(\mathsf{c}))}_{2} \vee \underbrace{\mathsf{g}(\mathsf{a}) = \mathsf{d}}_{3}) \wedge \neg\underbrace{(\mathsf{c} = \mathsf{d}\,)}_{4}$$

	$\parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	
$\Longrightarrow$	$1 \parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	unit propagate
$\Longrightarrow$	$1\ \overline{4} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	unit propagate

Example (Revisited with DPLL(T))

$$\underbrace{g(a) = c}_{1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{2}) \vee \underbrace{g(a) = d}_{3}) \wedge \neg(\underbrace{c = d}_{4})$$

$$
\begin{array}{lrl}
 & \| \; 1,\ (\overline{2} \vee 3),\ \overline{4} & \\
\Longrightarrow & 1 \; \| \; 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\,\overline{4} \; \| \; 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\,\overline{4}\,\overline{2}^{d} \; \| \; 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{decide}
\end{array}
$$

Example (Revisited with DPLL(T))

$$\underbrace{g(a) = c}_{1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{2}) \vee \underbrace{g(a) = d}_{3}) \wedge \neg(\underbrace{c = d}_{4})$$

$$
\begin{array}{lrl}
 & \parallel 1,\ (\overline{2} \vee 3),\ \overline{4} & \\
\Longrightarrow & 1 \parallel 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\,\overline{4} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\,\overline{4}\,\overline{2}^{d} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{decide} \\
\Longrightarrow & 1\,\overline{4}\,\overline{2}^{d} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2) & T\text{-learn}
\end{array}
$$

Example (Revisited with DPLL(T))

$$\underbrace{\mathsf{g(a) = c}}_{1} \wedge (\neg(\underbrace{\mathsf{f(g(a)) = f(c)}}_{2}) \vee \underbrace{\mathsf{g(a) = d}}_{3}) \wedge \neg(\underbrace{\mathsf{c = d}}_{4})$$

$$
\begin{array}{lrll}
 & \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \\
\Longrightarrow & 1 \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\ \overline{4} \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\ \overline{4}\ \overline{2}^{d} \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{decide} \\
\Longrightarrow & 1\ \overline{4}\ \overline{2}^{d} \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2) & T\text{-learn} \\
\Longrightarrow & 1\ \overline{4}\ 2 \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2) & T\text{-backjump}
\end{array}
$$

Example (Revisited with DPLL(T))

$$\underbrace{g(a) = c}_{1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{2}) \vee \underbrace{g(a) = d}_{3}) \wedge \neg(\underbrace{c = d}_{4})$$

	$\parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	
$\Longrightarrow$	$1 \parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	unit propagate
$\Longrightarrow$	$1\,\overline{4} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	unit propagate
$\Longrightarrow$	$1\,\overline{4}\,\overline{2}^{d} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4}$	decide
$\Longrightarrow$	$1\,\overline{4}\,\overline{2}^{d} \parallel 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2)$	T-learn
$\Longrightarrow$	$1\,\overline{4}\,2 \parallel 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2)$	T-backjump
$\Longrightarrow$	$1\,\overline{4}\,2\,3 \parallel 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2)$	unit propagate

Example (Revisited with DPLL(T))

$$\underbrace{g(a) = c}_{1} \wedge (\neg(\underbrace{f(g(a)) = f(c)}_{2}) \vee \underbrace{g(a) = d}_{3}) \wedge \neg(\underbrace{c = d}_{4})$$

$$
\begin{array}{lrl r}
 & \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \\
\Longrightarrow & 1 \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\ \overline{4} \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{unit propagate} \\
\Longrightarrow & 1\ \overline{4}\ \overline{2}^{d} \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4} & \text{decide} \\
\Longrightarrow & 1\ \overline{4}\ \overline{2}^{d} \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2) & T\text{-learn} \\
\Longrightarrow & 1\ \overline{4}\ 2 \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2) & T\text{-backjump} \\
\Longrightarrow & 1\ \overline{4}\ 2\ 3 \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2) & \text{unit propagate} \\
\Longrightarrow & 1\ \overline{4}\ 2\ 3 \parallel & 1,\ (\overline{2} \vee 3),\ \overline{4},\ (\overline{1} \vee 2),\ (\overline{1} \vee \overline{3} \vee 4) & T\text{-learn} \\
\Longrightarrow & \text{FailState} & & \text{fail}
\end{array}
$$

Lazyness in DPLL(T)


I'D LIKE TO HIRE SOMEONE TO DO ALL OF THE UNPLEASANT PARTS OF MY JOB.
THAT WAY I'LL BE FREE TO CONCENTRATE ON STRATEGY.
FAIR ENOUGH.
SECONDLY, I'D LIKE TO HIRE SOMEONE TO DO STRATEGY.

T-solver

Lazyness in DPLL(T)

T-solver SAT solver

Outline

- Summary of Last Week
- Satisfiability Modulo Theories
- DPLL(T)
- Using SMT Solvers with Theories

Example (SMT-LIB 2 for Propositional Logic)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_3)$ can be expressed by

```
(declare-const x1 Bool)
(declare-const x2 Bool)
(declare-const x3 Bool)
(assert (or x1 (not x3)))
(assert (or x2 x3 (not x1)))
(assert (or (not x1) x2 x3))
(check-sat)
(get-model)
```

Example (SMT-LIB 2 for Propositional Logic)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_3)$ can be expressed by

```
(declare-const x1 Bool)
(declare-const x2 Bool)
(declare-const x3 Bool)
(assert (or x1 (not x3)))
(assert (or x2 x3 (not x1)))
(assert (or (not x1) x2 x3))
(check-sat)
(get-model)
```

Propositional Logic in SMT-LIB 2

- `declare-const` x `Bool` creates propositional variable named x

Example (SMT-LIB 2 for Propositional Logic)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_3)$ can be expressed by

```
(declare-const x1 Bool)
(declare-const x2 Bool)
(declare-const x3 Bool)
(assert (or x1 (not x3)))
(assert (or x2 x3 (not x1)))
(assert (or (not x1) x2 x3))
(check-sat)
(get-model)
```

Propositional Logic in SMT-LIB 2

- `declare-const` x `Bool` creates propositional variable named x
- prefix notation for `and`, `or`, `not`, `implies`

Example (SMT-LIB 2 for Propositional Logic)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_3)$ can be expressed by

```
(declare-const x1 Bool)
(declare-const x2 Bool)
(declare-const x3 Bool)
(assert (or x1 (not x3)))
(assert (or x2 x3 (not x1)))
(assert (or (not x1) x2 x3))
(check-sat)
(get-model)
```

Propositional Logic in SMT-LIB 2

- `declare-const` x `Bool` creates propositional variable named x
- prefix notation for `and`, `or`, `not`, `implies`
- `assert` demands given formula to be satisfied

Example (SMT-LIB 2 for Propositional Logic)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_3)$ can be expressed by

```
(declare-const x1 Bool)
(declare-const x2 Bool)
(declare-const x3 Bool)
(assert (or x1 (not x3)))
(assert (or x2 x3 (not x1)))
(assert (or (not x1) x2 x3))
(check-sat)
(get-model)
```

Propositional Logic in SMT-LIB 2

- `declare-const` x `Bool` creates propositional variable named x
- prefix notation for `and`, `or`, `not`, `implies`
- `assert` demands given formula to be satisfied
- `check-sat` issues satisfiability check of conjunction of assertions

Example (SMT-LIB 2 for Propositional Logic)

formula $(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee x_2 \vee x_3)$ can be expressed by

```
(declare-const x1 Bool)
(declare-const x2 Bool)
(declare-const x3 Bool)
(assert (or x1 (not x3)))
(assert (or x2 x3 (not x1)))
(assert (or (not x1) x2 x3))
(check-sat)
(get-model)
```

Propositional Logic in SMT-LIB 2

- `declare-const` x `Bool` creates propositional variable named x
- prefix notation for `and`, `or`, `not`, `implies`
- `assert` demands given formula to be satisfied
- `check-sat` issues satisfiability check of conjunction of assertions
- `get-model` prints model (after satisfiability check)

Example (SMT-LIB 2 for EUF)

$f(f(a)) = a \land f(a) = b \land \neg(a = b)$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

Example (SMT-LIB 2 for EUF)

$\mathsf{f}(\mathsf{f}(\mathsf{a})) = \mathsf{a} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{b} \wedge \neg(\mathsf{a} = \mathsf{b})$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

EUF in SMT-LIB 2

- terms must have sort, so declare fresh sort and use for all symbols: `declare-sort` S creates sort named S

Example (SMT-LIB 2 for EUF)

$\mathsf{f}(\mathsf{f}(\mathsf{a})) = \mathsf{a} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{b} \wedge \neg(\mathsf{a} = \mathsf{b})$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

EUF in SMT-LIB 2

- terms must have sort, so declare fresh sort and use for all symbols: `declare-sort` S creates sort named S
- `declare-const` x s creates variable named x of sort S

Example (SMT-LIB 2 for EUF)

$f(f(a)) = a \wedge f(a) = b \wedge \neg(a = b)$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

EUF in SMT-LIB 2

- terms must have sort, so declare fresh sort and use for all symbols: `declare-sort` S creates sort named S
- `declare-const` x s creates variable named x of sort S
- `declare-fun` F $(S_1 \dots S_n)$ T creates uninterpreted $F\colon S_1 \times \cdots \times S_n \to T$

Example (SMT-LIB 2 for EUF)

$f(f(a)) = a \wedge f(a) = b \wedge \neg(a = b)$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

EUF in SMT-LIB 2

- terms must have sort, so declare fresh sort and use for all symbols:
  `declare-sort` S creates sort named S
- `declare-const` x s creates variable named x of sort S
- `declare-fun` F $(S_1 \dots S_n)$ T creates uninterpreted $F\colon S_1 \times \cdots \times S_n \to T$
- prefix notation as in `(f (f a))` to denote $f(f(a))$

Example (SMT-LIB 2 for EUF)

$\mathsf{f}(\mathsf{f}(\mathsf{a})) = \mathsf{a} \wedge \mathsf{f}(\mathsf{a}) = \mathsf{b} \wedge \neg(\mathsf{a} = \mathsf{b})$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

EUF in SMT-LIB 2

- terms must have sort, so declare fresh sort and use for all symbols:
  `declare-sort` S creates sort named S
- `declare-const` x s creates variable named x of sort S
- `declare-fun` F $(S_1 \dots S_n)$ T creates uninterpreted $F\colon S_1 \times \cdots \times S_n \to T$
- prefix notation as in `(f (f a))` to denote $\mathsf{f}(\mathsf{f}(\mathsf{a}))$ and `(=` x y`)` for equality

Example (SMT-LIB 2 for EUF)

$f(f(a)) = a \land f(a) = b \land \neg(a = b)$ is expressed as

```
(declare-sort A)
(declare-const a A)
(declare-const b A)
(declare-fun f (A) A)
(assert (= (f (f a)) a))
(assert (= (f a) b))
(assert (distinct a b))
(check-sat)
(get-model)
```

EUF in SMT-LIB 2

- terms must have sort, so declare fresh sort and use for all symbols: `declare-sort` S creates sort named S
- `declare-const` x s creates variable named x of sort S
- `declare-fun` F $(S_1 \dots S_n)$ T creates uninterpreted $F\colon S_1 \times \cdots \times S_n \to T$
- prefix notation as in `(f (f a))` to denote $f(f(a))$ and `(=` x y`)` for equality
- `(distinct` x y`)` is equivalent to `not(=` x y`)`

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Integer Arithmetic in SMT-LIB 2

- `declare-const` x `Int` creates integer variable named x

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Integer Arithmetic in SMT-LIB 2

- `declare-const` x `Int` creates integer variable named x
- numbers `0`, `1`, `-1`, `42`,... are built-in

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Integer Arithmetic in SMT-LIB 2

- `declare-const` x `Int` creates integer variable named x
- numbers `0`, `1`, `-1`, `42`,... are built-in
- `+`, `*`, `-` are $+_\mathbb{Z}$, $\cdot_\mathbb{Z}$, $-_\mathbb{Z}$

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Integer Arithmetic in SMT-LIB 2

- `declare-const` x `Int` creates integer variable named x
- numbers `0`, `1`, `-1`, `42`,… are built-in
- `+`, `*`, `-` are $+_{\mathbb{Z}}$, $\cdot_{\mathbb{Z}}$, $-_{\mathbb{Z}}$, used in prefix notation: `(+ 2 3)`

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Integer Arithmetic in SMT-LIB 2

- `declare-const` x `Int` creates integer variable named x
- numbers `0`, `1`, `-1`, `42`,... are built-in
- `+`, `*`, `-` are $+_{\mathbb{Z}}$, $\cdot_{\mathbb{Z}}$, $-_{\mathbb{Z}}$, used in prefix notation: `(+ 2 3)`
- `=` also covers equality on $\mathbb{Z}$

Example (SMT-LIB 2 for LIA)

$2x \geqslant y + z \wedge \neg(x = y)$ is expressed as

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(assert (>= (* 2 x) (+ y z)))
(assert (not (= x y)))
(check-sat)
(get-model)
```

Integer Arithmetic in SMT-LIB 2

- `declare-const` x `Int` creates integer variable named x
- numbers `0`, `1`, `-1`, `42`,... are built-in
- `+`, `*`, `-` are $+_{\mathbb{Z}}$, $\cdot_{\mathbb{Z}}$, $-_{\mathbb{Z}}$, used in prefix notation: `(+ 2 3)`
- `=` also covers equality on $\mathbb{Z}$
- `<`, `<=`, `>`, `>=` are $<_{\mathbb{Z}}$, $\leqslant_{\mathbb{Z}}$, $>_{\mathbb{Z}}$, $\geqslant_{\mathbb{Z}}$

EUF in python/z3

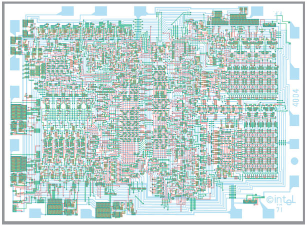
```
A = DeclareSort('A') # new uninterpreted sort named 'A'
a = Const('a', A) # create constant of sort A
b = Const('b', A) # create another constant of sort A
f = Function('f', A, A) # create function of sort A -> A

s = Solver()
s.add(f(f(a)) == a, f(a) == b, a != b)

print(s.check()) # sat
m = s.model()
print("interpretation assigned to A:")
print(m[A]) # [A!val!0, A!val!1]
print("interpretations:")
print(m[f]) # [A!val!0 -> A!val!1, A!val!1 -> A!val!0, ...]
print(m[a]) # A!val!0
print(m[b]) # A!val!1
```

EUF Application: Verification of Microprocessors

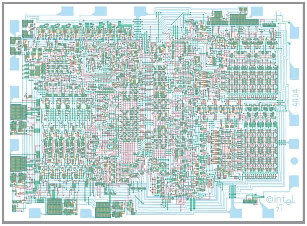
- verify that 3-stage pipelined MIPS processor satisfies intended instruction set architecture



Miroslav N. Velev and Randal E. Bryant.
Bit-level abstraction in the verification of pipelined microprocessors by correspondence checking.
In Proc. of Formal Methods in Computer-Aided Design, pp. 18–35, 1998.

EUF Application: Verification of Microprocessors

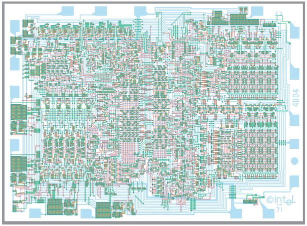
- verify that 3-stage pipelined MIPS processor satisfies intended instruction set architecture
- encoding
  - data as bit sequence
  - memory as uninterpreted function (UF)
  - computation logic as UF
  - control logic as uninterpreted predicate



Miroslav N. Velev and Randal E. Bryant.
Bit-level abstraction in the verification of pipelined microprocessors by correspondence checking.
In Proc. of Formal Methods in Computer-Aided Design, pp. 18–35, 1998.

EUF Application: Verification of Microprocessors

- verify that 3-stage pipelined MIPS processor satisfies intended instruction set architecture
- encoding
  - data as bit sequence
  - memory as uninterpreted function (UF)
  - computation logic as UF
  - control logic as uninterpreted predicate
- EUF ensures functional consistency: same data results in same computation



Miroslav N. Velev and Randal E. Bryant.
Bit-level abstraction in the verification of pipelined microprocessors by correspondence checking.
In Proc. of Formal Methods in Computer-Aided Design, pp. 18–35, 1998.

DPLL(T)

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.

Application

Miroslav N. Velev and Randal E. Bryant.
Bit-level abstraction in the verification of pipelined microprocessors by correspondence checking.
In Proc. of Formal Methods in Computer-Aided Design, pp. 18–35, 1998.