SMT (Satisfiability Modulo Theory)

Reasoning

Synthesis

Programming by Example

Type Directed Synthesis

Syntax

Guided Synthesis

Guided Inductive Synthesis

Refinement Types

Type Theory

Axiomatic Semantics

Satisfiability Modulo Theory

Boolean SAT

- 60-second crash course
  \[ A \land \neg (B \lor C) \land \neg D \]
  \[ \neg (B \lor C) \land (B \lor \neg D) \land (D \lor \neg A) \land (A \lor B \lor D) \]

Theories

- Is the following formula satisfiable?
  \[ x \geq 0 \land y \geq 0 \land 2 \cdot x \leq 10 \land |x - y| > 20 \land 4 \cdot x + 2 \cdot y = 47 \]

  \[ A \text{ theory is:} \]
  - A vocabulary of (first-order) symbols
  - Designated interpretations for some or all of the symbols in the vocabulary

Theory of QF_UF

- Quantifier-Free Uninterpreted Functions
  - Consists of: arbitrary function symbols, equality relation “=”
  - “=” must be interpreted as... equality.
    - Satisfies equality axioms:
      \[ \forall x, y. x = y \rightarrow f(x) = f(y) \] (for all functions, of any arity)

* QF_UF is also known as UF, EUF, and T[0]
QF_UF Solver

- **Input:** a set \( \varphi \) of equalities and disequalities in QF_UF.
- **Output:** A model \( M \models \varphi \), or UNSAT if none exists.

\[
\begin{align*}
g(a) &= c \\
g(a) &= f(b) \\
h(g(a)) &= h(c) \\
a \neq b \\
a \neq c \\
b \neq c
\end{align*}
\]

\[
\begin{align*}
g(a) &= c \\
g(a) &= f(b) \\
h(g(a)) &= h(c) \\
f(c) \neq f(f(b))
\end{align*}
\]

**Algorithm.**
- Collect all the terms occurring in \( \varphi \).
- Construct the congruence relation \( t_1 \sim t_2 \):
  - start with the input equalities and compute a closure w.r.t. equality axioms.
- Create a domain with one element per congruence class.
  - Set the interpretations of terms according to which class contains them.
- Check the obtained model against the disequalities.
  - All hold: return model. Otherwise: UNSAT.

DPLL(\( T \)) = SAT + Theory

- **Input:** a Boolean combination of propositional variables and literals from the theory.
- **Output:** (same as before) model or UNSAT.

**General idea.**
- Treat literals as opaque propositions and try to find a SAT assignment.
- Query theory solver to check if this assignment is consistent with the theory.
  - Theory solver should return a new (“learned”) literal in case of UNSAT.

Example: DPLL(QF_UF)

\[
\begin{align*}
g(a) &= c \\
&\land (f(g(a)) \neq f(c) \lor g(a) = d) \land c \neq d
\end{align*}
\]

Theory of QF_LIA

- Quantifier-Free Linear Integer Arithmetic
  - Consists of: +, -, multiplication by constant, equality (=), inequalities (\( \leq, \geq \)).
  - Designated interpretation: standard integer domain. Standard arithmetic and comparison operators.
- Turns out — QF_LIA is decidable.
  - (we will skip the algorithm;☺︎)
Quantifiers

- Our two main weapons:
  - Skolemization: basically gets rid of existentials.
    \[ \forall x. \exists y. Q(x, y) \land P(y) \]  
    \[ \equiv \forall x. Q(x, f(x)) \land P(f(x)) \]  
  - Herbrand’s Theorem: instructs us to instantiate universals.
    \[ \forall x. \phi \]  
    \[ \equiv \phi(x_0, \phi(x_1), \cdots) \]  
    (+ compactness)

Why should we even CARE?

Proof by Refutation

- Task: Prove that a formula \( \varphi \) is valid.
  \[ \models \varphi \]
- Alternatively:
  Prove that \( \neg \varphi \) is unsatisfiable.
  \[ \neg \exists M. M \models \neg \varphi \]

Proving Things About Programs

- ...but in first-order logic.

Proving Things About Programs

- ...but in first-order logic.
A general algorithm that will do this systematically for any program written in WHILE-language

- Known more succinctly as “weakest precondition”
- Define a new function this time:
  - $\text{wp}(c; Q)$
    Maps a command (c) and a postcondition (Q, an assertion) to the weakest precondition (P) that will make \{P\} c \{Q\} hold.

### Weakest Precondition

- $\text{wp}[	ext{skip}]; Q = Q$
- $\text{wp}[x := e]; Q = Q[e/x]$
- $\text{wp}[c_1; c_2]; Q = \text{wp}[c_1]; \text{wp}[c_2]; Q$
- $\text{wp}[\text{if } b \text{ then } c_1 \text{ else } c_2]; Q = (b \land \text{wp}[c_1]; Q) \lor (\neg b \land \text{wp}[c_2]; Q)$

### Verification Condition

- Now it’s almost too easy.
  - $\{P\} c \{Q\} \quad \phi \quad P \rightarrow \text{wp}[c]; Q$
  - (up to “good” invariant annotations)
  - $\text{wp}[c]; Q \in \{Q\}$
  - (warn)

### Next week

- We will do some synthesis.
• Lower bounds
  - All positive integers (except heads, which may be 0)

• Upper bounds
  - Sum over row ≤ width
  - Sum over column ≤ height

• Consistency
  - Row-wise color = column-wise color