Theories in Predicate Logic

and Satisfiability Modulo Theories (SMT)

Georg Hofferek
IAIK – Graz University of Technology
georg.hofferek@iaik.tugraz.at
Motivation

- Common “Use Cases”
  - Number Theory
  - Memory Models
  - …

- Common Frameworks
  - Fixed Premises (Axioms)
    - Specifying “Meaning” of Symbols
  - Specialized Tools
Outline

- What are Theories
  - Definition
  - Examples
- Model View
- Satisfiability for (Quantifier-free) Theory Formulas
  - Eager Encoding
    - Ackermann’s Reduction
    - Graph-based Reduction
  - Lazy Encoding
    - Congruence Closure Algorithm
  - DPLL(T)
Learning Targets

- Explain what a “Theory in Predicate Logic” is
  - Based on examples

- Explain the meaning of “Satisfiability Modulo Theories”
  - Based on examples

- Explain the Concept of Eager Encoding
  - Apply it to Formulas in $\mathcal{T}_{UE}$ Using Ackermann’s Reduction and the Graph-based Reduction

- Explain the Concept of Lazy Encoding
  - Apply it to Formulas in $\mathcal{T}_{UE}$ Using Congruence Closure

- Explain DPLL(T) and its advantages over Eager/Lazy Encoding
## Notion of “Theory”

<table>
<thead>
<tr>
<th>Application Domain</th>
<th>Structures &amp; Objects</th>
<th>Predicates &amp; Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>Numbers (Integers, Rationals, Reals)</td>
<td>= &lt; &gt; ≤ ≥ + ·</td>
</tr>
<tr>
<td>Computer Programs</td>
<td>Arrays, Bitvectors</td>
<td>Array-Read, Array-Write, …</td>
</tr>
</tbody>
</table>
Definition of a Theory

First-Order Theory $T$:

1. Signature $\Sigma$
   - Constants
   - Predicates
   - Functions

2. Set of Axioms $\mathcal{A}$
   - Sentences (=Formulas without free variables) with symbols from $\Sigma$ only

$\Sigma$-formula: (non-logic) symbols from $\Sigma$ only

$\Sigma, \mathcal{A}$: possibly infinite
Example: Theory of Equality $\mathcal{I}_E$

- **Signature** $\Sigma_E = \{=, a_0, b_0, c_0, d_0, ... \}$
  - Binary equality predicate $=$
  - Arbitrary constant symbols

- **Axioms** $\mathcal{A}_E$:
  1. $\forall x. x = x$ \hspace{1cm} (reflexivity)
  2. $\forall x. \forall y. (x = y \rightarrow y = x)$ \hspace{1cm} (symmetry)
  3. $\forall x. \forall y. \forall z. (x = y \land y = z \rightarrow x = z)$ \hspace{1cm} (transitivity)
Only models satisfying axioms are relevant for

- Satisfiability, Validity, Equivalence, Entailment
- \(\Rightarrow\) “Satisfiability modulo (=‘with respect to’) theories”
$\mathcal{T}$-Satisfiability

- **Green:** Models Satisfying all Axioms
- **Violet:** Models Satisfying Formula in Question
$\mathcal{T}$-Validity

- **Green**: Models Satisfying all Axioms
- **Violet**: Models Satisfying Formula in Question

$\mathcal{T}$-Valid

$\mathcal{T}$-Valid

Not $\mathcal{T}$-Valid
\( \mathcal{T} \)-Entailment and \( \mathcal{T} \)-Equivalence

- Similar to Satisfiability & Validity

- Only consider Models satisfying all axioms
  - Models not satisfying (at least) one axiom: Irrelevant Model!
Theory Formulas vs. Predicate Logic

Theory

Formula

$\phi^T$

$\mathcal{A} \rightarrow \phi$

equivalid

equisatisfiable

$\mathcal{A} \land \phi$
Fragment of a Theory

- Syntactically restricted subset
  - Quantifier-free fragment
  - Conjunctive fragment
  - Array Property fragment
    - Special grammar for index use
  - ...

Deciding Satisfiability of (quantifier-free) Theory Formulas

- Satisfiability *modulo* Theories (SMT)
  - Theory Atoms
    - \( a = b, b = c, ... \)
  - Propositional structure
    - \( (a = b) \lor (b = c) \land \neg (c = a) \)
    - \( p \lor q \land \neg r \)
Deciding Satisfiability of (quantifier-free) Theory Formulas

- **Eager Encoding**
  - Equisatisfiable propositional formula
    - Constraint clauses
  - SAT solver

- **Lazy Encoding**
  - SAT Solver
  - Theory Solver
    - Conjunctive Fragment
    - Blocking Clauses
Example: Theory of Uninterpreted Functions and Equality $\mathcal{T}_{UE}$

- **Signature** $\Sigma_{UE} = \{=, a, b, c, d, ... \}$
  - Binary equality predicate $=$
  - Arbitrary constant- and function-symbols

- **Axioms** $\mathcal{A}_{UE}$:
  1.-3. same as in $\mathcal{A}_E$ (reflexivity), (symmetry), (transitivity)
  4. $\forall \bar{x}. \forall \bar{y}. ((\wedge_i x_i = y_i) \rightarrow f(\bar{x}) = f(\bar{y}))$ (function congruence)

**Axiom Schema:** Template for (infinite number of) axioms
Two-Stage Eager Encoding

quant.-free $\mathcal{T}_{UE}$-formula

Ackermann’s Reduction

equisatisfiable quant.-free $\mathcal{T}_{E}$-formula

Graph-based Reduction

equisatisfiable propositional formula
Ackermann’s Reduction

- Fresh Variables
  - \( f(x) \mapsto f_x \)

- Functional Constraints
  - \( (x = y) \rightarrow (f_x = f_y) \)

- \( \phi_E = \phi_{FC} \land \hat{\phi}_{UE} \)
Graph-Based Reduction

- Non-Polar Equality Graph
  - Node per variable
  - Edge per (dis)equality

- Make it **chordal**
  - No chord-free cycles (size > 3)
Graph-Based Reduction

- **Fresh Propositional Variables**
  - \(a = b \iff e_{a=b}\)
  - **Order!**
    - \(b = a \iff e_{a=b}\)

- **Triangle \((i, j, k)\):**
  - **Transitivity Constraints**
    - \((e_{i=j} \land e_{j=k} \rightarrow e_{i=k}) \land (e_{i=j} \land e_{i=k} \rightarrow e_{j=k}) \land (e_{i=k} \land e_{j=k} \rightarrow e_{i=j})\)

- \(\phi_{prop} = \phi_{TC} \land \hat{\phi}_E\)
Lazy Encoding

φ

SAT Solver

Assignment of Literals

Theory Solver

Blocking Clause

UNSAT

SAT
Conjunctive Fragment of $\mathcal{T}_{UE}$

- Conjunction of theory literals
  - Equalities ($t_1 = t_2$)
  - Disequalities $\neg(t_1 = t_2)$
    - alternatively: ($t_1 \neq t_2$)
- Terms $t_i$
  - Constants
    - $a, b, c, d, ...$
  - Uninterpreted Function instances
    - $f(a), g(b), h(c, d), ...$
Congruence-Closure Algorithm

- **Equivalence Classes**
  - $t_1 = t_2$: same class for $t_1, t_2$
  - Other terms: singleton class
  - Shared Term between classes:
    - Merge classes! (repeat)
  - $t_i, t_j$ from same class:
    - Merge classes of $f(t_i), f(t_j)$ (repeat)

- **Check Disequalities** $t_k \neq t_l$
  - $t_k, t_l$ in same class: **UNSAT**!
  - Otherwise: **SAT**!
Disadvantages of Lazy Encoding

- Many (similar) SAT calls
- Full assignment before call to Theory solver
- Very specific blocking clauses

Solution:
Integration into DPLL
DPLL(T)

Start

Decide

full assignment

partial assignment

Learn & Backtrack

conflict

partial assignment

BCP/PL

Analyze Conflict

UNSAT

partial assignment

Add Clauses

theory propagation / conflict

Theory Solver

partial assignment

SAT

partial assignment
Summary

- Notion of “Theory”
  - Satisfiability Modulo Theories

- Eager Encoding
  - Example: Ackermann’s Reduction & Graph-based Reduction

- Lazy Encoding
  - Example: Congruence Closure

- DPLL(T)
  - Advantages over Eager/Lazy Encoding