Decision Procedures in First Order Logic

Decision Procedures for Equality Logic

Outline

- Introduction
  - Definition, complexity
  - Reducing Uninterpreted Functions to Equality Logic
  - Using Uninterpreted Functions in proofs
  - Simplifications
- Introduction to the decision procedures
  - The framework: assumptions and Normal Forms
  - General terms and notions
  - Solving a conjunction of equalities
  - Simplifications

Basic assumptions and notations

- Input formulas are in NNF
- Input formulas are checked for satisfiability
- Formula with Uninterpreted Functions: $\phi^{UF}$
- Equality formula: $\phi^E$

First: conjunction of equalities

- **Input:** A conjunction of equalities and disequalities
  1. Define an equivalence class for each variable. For each equality $x = y$ unite the equivalence classes of $x$ and $y$. Repeat until reach a fixed point.
  2. For each disequality $u \neq v$ if $u$ is in the same equivalence class as $v$ return 'UNSAT'.
  3. Return 'SAT'.
Example

- $x_1 = x_2 \land x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1$

Is there a disequality between members of the same class?

Next: add Uninterpreted Functions

- $x_1 = x_2 \land x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1 \land F(x_1) \neq F(x_2)$

Next: Compute the Congruence Closure

- $x_1 = x_2 \land x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1 \land F(x_1) \neq F(x_2)$

Now - is there a disequality between members of the same class?

This is called the Congruence Closure

And now: consider a Boolean structure

- $x_1 = x_2 \lor (x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1 \land F(x_1) \neq F(x_2))$

Syntactic case splitting: this is what we want to avoid!
Deciding Equality Logic with UFs

- Input: Equality Logic formula $\phi^{\text{UF}}$
- Convert $\phi^{\text{UF}}$ to DNF
- For each clause:
  - Define an equivalence class for each variable and each function instance.
  - For each equality $x = y$ unite the equivalence classes of $x$ and $y$. For each function symbol $F$, unite the classes of $F(x)$ and $F(y)$. Repeat until reach a fixed point.
  - If all disequalities are between terms from different equivalence classes, return 'SAT'.
- Return 'UNSAT'.

Basic notions

$\phi^E$: $x = y \land y = z \land z \neq x$

- The Equality predicates: \{$x = y, y = z, z \neq x$\} which we can break to two sets: $E_\times =$\{$x = y, y = z$\}, $E_\neq =$\{$z \neq x$\}
- The Equality Graph $G^E(\phi^E) = (V, E_\times, E_\neq)$ (a.k.a “E-graph”)

Basic notions

$\phi_1^E$: $x = y \land y = z \land z \neq x$ unsatisfiable
$\phi_2^E$: $x = y \land y = z \lor z \neq x$ satisfiable

The graph $G^E(\phi^E)$ represents an abstraction of $\phi^E$. It ignores the Boolean structure of $\phi^E$. 

Basic notions

- Dfn: a path made of $E_\times$ edges is an Equality Path. We write $x =^* z$ if there is an equality path from $x$ to $z$
- Dfn: a path made of $E_\times$ edges and exactly one edge from $E_\neq$ is a Disequality Path. We write $x \neq^* y$ if there is a disequality path from $x$ to $y$. 

Basic notions

- **Dfn.** A cycle with exactly one disequality edge is a **Contradictory Cycle.**
- In a Contradictory Cycle, for every two nodes $x, y$ it holds that $x \neq^* y$ and $x \neq^* y$.

Basic notions

- **Thm:** Every Contradictory Cycle is either simple or contains a simple contradictory cycle

Simplifications, again

- Let $S$ be the set of edges that are not part of any Contradictory Cycle
- **Thm:** replacing all solid edges in $S$ with False, and all dashed edges in $S$ with True, preserves satisfiability
Simplification: example

\[(x_1 = x_2 \lor x_1 = x_3) \land (x_1 \neq x_3 \lor x_2 = x_3)\]
\[(x_1 = x_2 \lor \text{True}) \land (x_1 \neq x_3 \lor x_2 = x_3)\]
\[\neg \text{False} \lor \text{True} = \text{True}\]

Satisfiable!

Syntactic vs. Semantic splits

- So far we saw how to handle disjunctions through syntactic case-splitting.
- There are much better ways to do it than simply transforming it to DNF:
  - Semantic Tableaux,
  - SAT-based splitting,
  - others…
- We will investigate some of these methods later in the course.

Now we start looking at methods that split the search space instead. This is called semantic splitting.

SAT is a very good engine for performing semantic splitting, due to its ability to guide the search, prune the search-space etc.