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 convergence.

2. For each disequality \( u \neq v \) if \( u \) is in the same equivalence class as \( v \) return 'UNSAT'.

3. Return 'SAT'.

Example

- \( x_1 = x_2 \not\in \equiv x_2 = x_3 \not\in \equiv x_4 = x_5 \not\in \equiv x_5 \neq x_1 \)

Is there a disequality between members of the same class?
Next: add Uninterpreted Functions

\[ x_1 = x_2 \land x_3 = x_4 = x_5 \neq x_1 \land \neg F(x_1) \neq F(x_2) \]
Next: Compute the *Congruence Closure*

\[ x_1 = x_2 \land x_3 \land x_4 = x_5 \land x_1 \neq x_1 \land \neg F(x_1) \neq F(x_2) \]

**Equivalence class**

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

This is called the *Congruence Closure*
And now: consider a Boolean structure

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

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

- Input: Equality Logic formula $\phi^{UF}$
- Convert $\phi^{UF}$ 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 convergence.
  - If all disequalities are between terms from different equivalence classes, return 'SAT'.
- Return 'UNSAT'.

Decision Procedures
An algorithmic point of view
Algorithm 3.3.1: ACKERMANN’S-REDUCTION

**Input:** An EUF formula $\varphi^\text{UF}$ with $m$ instances of an uninterpreted function $F$.

**Output:** An equality logic formula $\varphi^E$ such that $\varphi^E$ is valid if and only if $\varphi^\text{UF}$ is valid.

1. Assign indices to the uninterpreted-function instances from subexpressions outwards. Denote by $F_i$ the instance of $F$ that is given the index $i$, and by $\text{arg}(F_i)$ its single argument.
2. Let $\text{flat}^E \doteq T(\varphi^\text{UF})$, where $T$ is a function that takes an EUF formula (or term) as input and transforms it to an equality formula (or term, respectively) by replacing each uninterpreted-function instance $F_i$ with a new term-variable $f_i$ (in the case of nested functions, only the variable corresponding to the most external instance remains).
3. Let $FC^E$ denote the following conjunction of functional consistency constraints:

$$FC^E := \bigwedge_{i=1}^{m-1} \bigwedge_{j=i+1}^m (T(\text{arg}(F_i)) = T(\text{arg}(F_j))) \implies f_i = f_j.$$

4. Let

$$\varphi^E := FC^E \implies \text{flat}^E.$$

Return $\varphi^E$. 

Decision Procedures
An algorithmic point of view
10
**Algorithm 3.3.2: BRYANT’S-REDUCTION**

**Input:** An EUF formula \( \varphi_{\text{UF}} \) with \( m \) instances of an uninterpreted function \( F \)

**Output:** An equality logic formula \( \varphi^E \) such that \( \varphi^E \) is valid if and only if \( \varphi_{\text{UF}} \) is valid

1. Assign indices to the uninterpreted-function instances from subexpressions outwards. Denote by \( F_i \) the instance of \( F \) that is given the index \( i \), and by \( \text{arg}(F_i) \) its single argument.

2. Let \( \text{flat}^E = T^*(\varphi_{\text{UF}}) \), where \( T^* \) is a function that takes an EUF formula (or term) as input and transforms it to an equality formula (or term, respectively) by replacing each uninterpreted-function instance \( F_i \) with a new term-variable \( F_i^* \) (in the case of nested functions, only the variable corresponding to the most external instance remains).

3. For \( i \in \{1, \ldots, m\} \), let \( f_i \) be a new variable, and let \( F_i^* \) be defined as follows:

   \[
   F_i^* := \begin{cases} 
   \text{case } T^*(\text{arg}(F_i^*)) = T^*(\text{arg}(F_i^*)) : f_1 \\
   \vdots \\
   \vdots \\
   \text{TRUE} : f_i \\
   \end{cases} \quad (3.19)
   
   \]

   Finally, let

   \[
   FC^E := \bigwedge_{i=1}^{m} F_i^* \quad (3.20)
   
   \]

4. Let

   \[
   \varphi^E := FC^E \implies \text{flat}^E \quad (3.21)
   
   \]

Return \( \varphi^E \).
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:
  $E = \{x = y, y = z\}, \quad E \neq = \{z \neq x\}$

- The **Equality Graph** $G^E(\phi^E) = \langle V, E =, E \neq \rangle$
  (a.k.a “E-graph”)

\[ \begin{array}{c}
\text{y} \\
\text{x} \\
\text{z} \\
\end{array} \]
Basic notions

\[ \phi_1^E: \quad x = y \land \forall y = z \land \forall z \neq x \quad \text{unsatisfiable} \]
\[ \phi_2^E: \quad x = y \land \forall y = z \land z \neq x \quad \text{satisfiable} \]

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 $E_\equiv$ edges is an *Equality Path.* We write $x =^* z$.

- **Dfn:** a path made of $E_\equiv$ edges + exactly one edge from $E_\not\equiv$ is a *Disequality Path.* We write $x \not=^* y$. 

Decision Procedures
An algorithmic point of view
14
Basic notions

- **Dfn.** A cycle with one disequality edge is a Contradictory Cycle.

- In a Contradictory Cycle, for every two nodes $x, y$ it holds that $x =^* y$ and $x \neq^* y$. 

Basic notions

- **Dfn:** A subgraph is called *satisfiable* iff the conjunction of the predicates represented by its edges is satisfiable.

- **Thm:** A subgraph is unsatisfiable iff it contains a **Contradictory cycle**
Basic notions

Thm: Every Contradictory Cycle is either simple or contains a simple contradictory cycle
Algorithm 4.3.1: SIMPLIFY-EQUALITY-FORMULA

**Input:** An equality formula $\varphi^E$

**Output:** An equality formula $\varphi_E'$ equisatisfiable with $\varphi^E$, with

length less than or equal to the length of $\varphi^E$

1. Let $\varphi_E' := \varphi^E$.
2. Construct the equality graph $G^E(\varphi_E')$.
3. Replace each pure literal in $\varphi_E'$ whose corresponding edge is not part of a simple contradictory cycle with TRUE.
4. Simplify $\varphi_E'$ with respect to the Boolean constants TRUE and FALSE (e.g., replace TRUE $\lor \phi$ with TRUE, and FALSE $\land \phi$ with FALSE).
5. If any rewriting has occurred in the previous two steps, go to step 2.
6. Return $\varphi_E'$.
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

- $(x_1 = x_2 \land x_1 = x_4) \not\equiv (x_1 \neq x_3 \land x_2 = x_3)$
- $(x_1 = x_2 \land \text{True}) \not\equiv (x_1 \neq x_3 \land x_2 = x_3)$
- $(\text{False} \land \text{True}) = \text{True}$

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.
Syntactic vs. Semantic splits

- 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.