Decision Procedures in First Order Logic

Decision Procedures for Equality Logic
Part III – Decision Procedures for Equality Logic and Uninterpreted Functions

- Algorithm I – From Equality to Propositional Logic
  - Adding transitivity constraints
  - Making the graph chordal
  - An improved procedure: consider polarity

- Algorithm II – Range-Allocation
  - What is the small-model property?
  - Finding a small adequate range (domain) to each variable
  - Reducing to Propositional Logic
We will first investigate methods that solve Equality Logic. Uninterpreted functions are eliminated with one of the reduction schemes.

Our starting point: the E-Graph $G^E(\phi^E)$

Recall: $G^E(\phi^E)$ represents an abstraction of $\phi^E$:
It represents ALL equality formulas with the same set of equality predicates as $\phi^E$
From Equality to Propositional Logic
Bryant & Velev 2000: the *Sparse* method

\[ \phi^E = x_1 = x_2 \land x_2 = x_3 \land x_1 \neq x_3 \]
\[ \phi_{\text{enc}} = e_1 \land e_2 \land \neg e_3 \]

- Encode all edges with Boolean variables
  - (note: for now, ignore polarity)
  - This is an abstraction
  - Transitivity of equality is lost!
  - Must add transitivity constraints!
From Equality to Propositional Logic

\[ \phi^E = x_1 = x_2 \land x_2 = x_3 \land x_1 \neq x_3 \]
\[ \phi_{\text{enc}} = e_1 \land e_2 \land \neg e_3 \]

- For each cycle add a transitivity constraint

\[ \phi_{\text{trans}} = (e_1 \land e_2 \rightarrow e_3) \land (e_1 \land e_3 \rightarrow e_2) \land (e_3 \land e_2 \rightarrow e_1) \]

Check: \[ \phi_{\text{enc}} \land \phi_{\text{trans}} \]
There can be an exponential number of cycles, so let’s try to make it better.

**Thm:** it is sufficient to constrain simple cycles only
Still, there is an exponential number of simple cycles.

Thm [Bryant & Velev]: *It is sufficient to constrain chord-free simple cycles*
Still, there can be an exponential number of chord-free simple cycles…

Solution: make the graph ‘chordal’ by adding edges.
From Equality to Propositional Logic

- Dfn: A graph is chordal iff every cycle of size 4 or more has a chord.

- How to make a graph chordal? Eliminate vertices one at a time, and connect their neighbors.
From Equality to Propositional Logic

- Once the graph is chordal, we can constrain only the triangles.

- Note that this procedure adds not more than a polynomial # of edges, and results in a polynomial no. of constraints.
Part III – Decision Procedures for Equality Logic and Uninterpreted Functions

- Algorithm I – From Equality to Propositional Logic
  - Adding transitivity constraints
  - Making the graph chordal
  - An improved procedure: consider polarity

- Algorithm II – Range-Allocation
  - What is the small-model property?
  - Finding a small adequate range (domain) to each variable
  - Reducing to Propositional Logic
Range allocation

- The small model property
- Range Allocation
Uninterpreted functions

From a general formula:

\[ u_1 = x_1 + y_1 \land u_2 = x_2 + y_2 \land z = u_1 \times u_2 \rightarrow \]

\[ z = (x_1 + y_1) \times (x_2 + y_2) \]

To a formula with uninterpreted functions

\[ u_1 = F(x_1, y_1) \land u_2 = F(x_2, y_2) \land z = G(u_1, u_2) \rightarrow \]

\[ z = G(F(x_1, y_1), F(x_2, y_2)) \]
Ackerman’s reduction

*From a formula with uninterpreted functions:*

\[ u_1 = F(x_1, y_1) \land u_2 = F(x_2, y_2) \land z = G(u_1, u_2) \rightarrow \]
\[ z = G(F(x_1, y_1), F(x_2, y_2)) \]

*To a formula in the theory of equality*

\[
\begin{pmatrix}
(x_1 = x_2 \land y_1 = y_2 \rightarrow f_1 = f_2) \land \\
(u_1 = f_1 \land u_2 = f_2 \rightarrow g_1 = g_2) \land \\
(u_1 = f_1 \land u_2 = f_2 \land z = g_1)
\end{pmatrix}
\rightarrow z = g_2
\]
The Small Model Property

- Equality Logic enjoys the *Small Model Property*
- This means that if a formula in this logic is satisfiable, then there is a *finite, bounded in size, model* that satisfies it.
- It gets better: in Equality Logic we can compute this *bound*, which suggests a decision procedure.
- What is this bound?
The Small Model Property

- **Claim**: the range 1..n is adequate, where n is the number of variables in \( \phi \)

- **Proof**:
  - Every satisfying assignment defines a partition of the variables
  - Every assignment that results in the same partitioning also satisfies the formula
  - The range 1..n allows all partitionings
Complexity

- We need $\log n$ variables to encode the range $1 \ldots n$
- For $n$ variables we need $n \log n$ bits.
- This is already better than the worst-case $O(n^2)$ bits required by the Boolean encoding method …
Finite Instantiations revisited

\[
\begin{align*}
\left( x_1 = x_2 \land y_1 = y_2 \rightarrow f_1 = f_2 \right) \land \\
\left( u_1 = f_1 \land u_2 = f_2 \rightarrow g_1 = g_2 \right) \land \\
\left( u_1 = f_1 \land u_2 = f_2 \land z = g_1 \right)
\end{align*}
\]

Instead of giving the range \([1..11]\), analyze connectivity:

\[
\begin{array}{c}
x_1 & \rightarrow & x_2 \\
y_1 & \rightarrow & y_2 \\
u_1 & \rightarrow & f_1 & \rightarrow & f_2 & \rightarrow & u_2 \\
g_1 & \rightarrow & g_2 \\
z
\end{array}
\]

\[
\begin{align*}
x_1, y_1, x_2, y_2 & : \{0-1\} & u_1, f_1, f_2, u_2 & : \{0-3\} & g_1, g_2, z & : \{0-2\}
\end{align*}
\]

The state-space: from \(11^{11}\) to \(\sim 10^5\)
Or even better:

\[ x_1, y_1, g_1, u_1 : \{0\} \quad x_2, y_2, g_2, f_1 : \{0-1\} \]
\[ f_2, z : \{0-2\} \quad u_2 : \{0-3\} \]

The state-space: from \(\sim 10^5\) to 576

An Upper-bound: State-space \(\leq n!\)
Choosing a minimal range for the integer variables

\[ \phi^E: \neg((a = b) \lor \neg(b = c)) \rightarrow ((d \neq e)) \]

0. \(a, b, c, d, e: \{\infty\}\) \hspace{2cm} (\infty) \hspace{2cm} (normal)

1. \(a, b, c, d, e: \{1..5\}\) \hspace{2cm} (3125) \hspace{2cm} (1..n)

2. \(a, b, c: \{1..3\}\) \hspace{2cm} (connectivity analysis)
   \(d, e: \{1..2\}\) \hspace{2cm} (108)

3. \(a: \{1\}, b: \{1-2\}, c: \{1-3\}\) \hspace{2cm} (factorial reduction)
   \(d: \{1\}, e: \{1-2\}\) \hspace{2cm} (12)

4. \(\ldots\) \hspace{2cm} \(\ldots\) \hspace{2cm} \(\ldots\)
Before and after, in SMV

MODULE main

VAR

H_zN1_693_c : 0..31;
zN1_693_c : 0..31;
N1_643_c : 0..31;
T1_c : 0..31;
T1_644_c : 0..31;
N1_c : 0..31;
f_plus1 : 0..31;
f_plus2 : 0..31;
f_minus1 : 0..31;
f_minus2 : 0..31;
f_minus3 : 0..31;
f_minus4 : 0..31;
f_mul1 : 0..31;
f_mul2 : 0..31;
f_div1 : 0..31;
f_div2 : 0..31;
f_div3 : 0..31;
f_div4 : 0..31;
sqrt_1 : 0..31;
sqrt_2 : 0..31;
POSM_c : boolean;
POSM_33_c : boolean;
H0_99_c : boolean;

MODULE main

VAR

H_zN1_693_c : {33};
zN1_693_c : {33};
N1_643_c : {19};
T1_c : {27};
T1_644_c : {27, 28};
N1_c : {19};
f_plus1 : {0, 21, 22};
f_plus2 : {21, 0};
f_minus1 : {8, 9, 10, 11};
f_minus2 : {8, 9, 10, 11};
f_minus3 : {8, 9, 10, 11};
f_minus4 : {8, 9, 10, 11};
f_mul1 : {16};
f_mul2 : {16};
f_div1 : {23, 24, 25};
f_div2 : {23, 24, 25};
f_div3 : {24, 23};
f_div4 : {23};
sqrt_1 : {29};
sqrt_2 : {29, 30};
POSM_c : boolean;
POSM_33_c : boolean;
H0_99_c : boolean;
The Range-Minimization Problem

Given an Equality formula $\phi^E$, find in polynomial time a small domain $D^*$ sufficient to preserve its satisfiability.

In other words: find $D^*$ such that

$\phi^E$ is satisfiable under an infinite domain $D \rightarrow \phi^E$ is satisfiable under the finite domain $D^*$
The strategy

1. Determine a range allocation $R$, mapping each variable $x_i \in \phi^E$ into a small set of integers, s.t. $\phi^E$ is satisfiable iff it is satisfiable over some $R$-interpretation.

2. Encode each variable $x_i$ as an enumerated type over $R(x_i)$, and use a standard satisfiability checker as a decision procedure.
What range is adequate?

- Recall that a subgraph of an E-Graph $G^E(\phi^E)$ is unsatisfiable iff it contains a contradictory cycle.

- Dfn: A Range Allocation $R$ is adequate for $G^E(\phi^E)$ if every satisfiable subgraph can be satisfied under $R$.

- Now we need an algorithm to find adequate ranges.
Examples:

<table>
<thead>
<tr>
<th>$\phi^E$</th>
<th>Predicates in $\phi^E$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x_1=x_2) \land (x_2=x_3)$</td>
<td>${(x_1=x_2),(x_2=x_3)}$</td>
<td>$x_1, x_2, x_3 \rightarrow {0}$</td>
</tr>
<tr>
<td>$(x_1\neq x_2) \land (x_2\neq x_3)$</td>
<td>${(x_1\neq x_2),(x_2\neq x_3)}$</td>
<td>$x_1 \rightarrow {0}$, $x_2 \rightarrow {1}$, $x_3 \rightarrow {2}$</td>
</tr>
<tr>
<td>$(x_1\neq x_2) \lor (\text{False} \land (x_1=x_2))$</td>
<td>${(x_1\neq x_2),(x_1=x_2)}$</td>
<td>$x_1 \rightarrow {0}$, $x_2 \rightarrow {0,1}$</td>
</tr>
<tr>
<td>$(x_1=x_2) \lor (\text{False} \land (x_1\neq x_2))$</td>
<td>${(x_1\neq x_2),(x_1=x_2)}$</td>
<td>$x_1 \rightarrow {0}$, $x_2 \rightarrow {0,1}$</td>
</tr>
</tbody>
</table>

The price of a polynomial procedure:
The predicates of $\phi^E$ hold less information than $\phi^E$. 
Building the E-Graph

\[ E_+: \{ (x_1 \neq x_2), (y_1 \neq y_2), (u_1 \neq f_1), (u_2 \neq f_2), (g_2 \neq z) \} \]

\[ E_- : \{ (f_1 = f_2), (g_1 = g_2), (u_1 = f_1), (u_2 = f_2), (g_1 = z) \} \]

Note: 1. Inconsistent subsets appear as contradictory cycles
2. Some of the vertices are mixed
The Range-Allocation Algorithm

Step I - pre-processing:

A. Remove all solid edges not belonging to contradictory cycles.
B. Add a single unique value to singleton vertices, and remove them from the graph.
Step II - Set construction:

A. For each mixed vertex $x_i$:
   1. Add a unique value $u_i$ to $R(x_i)$
   2. Broadcast $u_i$ on $G_=$
   3. Remove $x_i$ from the graph

B. Add a unique value to each remaining $G_=$ component
1. 

\[ \begin{array}{c}
\{6\} \\
\text{\(u_1\)} \quad \text{\(f_1\)} \quad \text{\(f_2\)} \quad \text{\(u_2\)} \\
\end{array} \]

2. 

\[ \begin{array}{c}
\{6,7\} \\
\text{\(f_1\)} \quad \text{\(f_2\)} \quad \text{\(u_2\)} \\
\end{array} \]

3. 

\[ \begin{array}{c}
\text{\(f_1\)} \\
\{6,7,\bar{8}\} \quad \{6,7,\bar{9}\} \\
\text{\(u_2\)} \\
\end{array} \]

Decision Procedures
An algorithmic point of view

29
Is the allocated range adequate?

■ We have to satisfy every consistent subset $B$:
  □ For all $x \in B$, assign the smallest value allocated in step $A$ to a mixed vertex which is $G_\geq(B)$ connected to $x$.
  □ If there isn’t any, choose the value given in step $B$.
Further optimizations

- The order in which mixed vertices are eliminated has a strong effect.
- Not all mixed vertices need to start from a unique value. An analysis that involves solving a coloring problem can help here…
- … (see lecture notes)
A state-space story

$$1..n \quad 1..i \quad \text{connectivity} \quad \text{basic} \quad \text{order} \quad \text{color}$$

$$11^{11} \quad 11! \quad 576 \quad 72 \quad 48 \quad 16 \quad ?$$

Decision Procedures
An algorithmic point of view 32