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

From Equality to Propositional Logic
Bryant & Velev 2000: the Sparse method

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

- 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

\[ \phi^E = (x_1 = x_2 \land x_2 = x_3 \land x_1 \neq x_3) \]

\[ \phi_{enc} = e_1 \land e_2 \land \neg e_3 \]

- For each cycle add a transitivity constraint

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

Check: \( \phi_{enc} \land \phi_{trans} \)

Decision Procedures
An algorithmic point of view

From Equality to Propositional Logic

- There can be an exponential number of cycles, so let’s try to make it better.
- Thm: it suffices to constrain simple cycles only.

Thm [Bryant & Velev]: It suffices to constrain chord-free simple cycles.

Still, there can be an exponential number of 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.

**Example**

- 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 number of constraints.

Improvement

- So far we did not consider the polarity of the edges.
- **Claim:** in the following graph \( \varphi_{rbc} = e_3 \land e_2 \rightarrow e_1 \) is sufficient
- **This is only true because of monotonicity of NNF**

Skipping rest of this set of slides. Left here for completeness (but they need work).
Definitions

- Dfn: A contradictory cycle \( C \) is constrained under formula \( T \) if \( T \) does not allow this assignment

\[
C = \begin{array}{ccc}
T & T & T \\
T & F & T \\
\end{array}
\]

Main theorem

- If \( T^R \) constrains all simple contradictory cycles, and

\[
\text{For every assignment } \alpha^S, \alpha^S \vdash T^S \rightarrow \alpha^S \vdash T^R
\]

- then

\( \phi^E \) is satisfiable iff \( B \wedge T^R \) is satisfiable

Proof of the main theorem

- \( \rightarrow \) \( \phi^E \) is satisfiable \( \Rightarrow B \wedge T^S \) is satisfiable \( \Rightarrow B \wedge T^R \) is satisfiable

- \( \leftarrow \) Proof strategy:
  - Let \( \alpha^R \) be a satisfying assignment to \( B \wedge T^R \)
  - We will construct \( \alpha^S \) that satisfies \( B \wedge T^S \)
  - From this we will conclude that \( \phi^E \) is satisfiable
Definitions for the proof…

- A **Violating cycle** under an assignment $\alpha^R$:

- This assignment violates $T^S$ but not necessarily $T^R$.

More definitions for the proof…

- An edge $e = (v_i, v_j)$ is **equal under an assignment** $\alpha$ iff there is an equality path between $v_i$ and $v_j$ all assigned $T$ under $\alpha$. Denote: $v_i \equiv^*_\alpha v_j$.

- An edge $e = (v_i, v_j)$ is **disequal under an assignment** $\alpha$ iff there is a disequality path between $v_i$ and $v_j$ in which the solid edge is the only one assigned false by $\alpha$. Denote: $v_i \not\equiv^*_\alpha v_j$.

Proof…

- Observation 1: The combination is impossible if $\alpha = \alpha^R \not\equiv v_2 \implies v_1 \not\equiv v_3$ (recall: $\alpha^R \not\subseteq T^R$).

- Observation 2: if $(v_1, v_3)$ is solid, then $v_1 \not\equiv v_3$. 
ReConstructing $\alpha^S$

**Type 1:**
It is *not* the case that $v_2 \models^\alpha v_3$

- Assign $\alpha^S(e_{23}) = F$
- Assign $\alpha(e_{13}) = T$

In all other cases $\alpha^S = \alpha^R$

**Type 2:**
Otherwise it is *not* the case that $v_1 \not\models^\alpha v_2$

- Assign $\alpha^S(e_{13}) = T$
- Assign $\alpha(e_{13}) = T$

Proof...

- Invariant: contradictory cycles are not violating throughout the reconstruction.

- $v_2 \models^\alpha v_2$ contradicts the precondition to make this assignment...

Proof...

- Invariant: contradictory cycles are not violating throughout the reconstruction.

- $v_1 \not\models^\alpha v_3$ contradicts the precondition to make this assignment...
Applying RTC

- How can we use the theorem without enumerating contradictory cycles?
- Answer:
  - Consider the chordal graph.
  - Constrain triangles if they are part of a (simple) contradictory cycle
  - How?

The RTC algorithm

- Bi-Connected Component (BCC) – maximal set of edges s.t. any two edges in the set lie on a common simple cycle.
- Find all BCC-s with one solid edge.
- Add solid chords from the graph, and auxiliary dashed chords to make the BCC chordal.

Decomposing the graph

- Focus on Bi-connected dashed components built on top of a solid edge
  - Includes all contradictory cycles involving this edge
Make the component chordal
- Chordal-ity guarantees: every cycle contains a simplicial vertex, i.e. a vertex that its neighbors are connected.

The RTC algorithm
- Constraints cache:
  - $e_2 \land e_3 \rightarrow e_1$
  - $e_4 \land e_5 \rightarrow e_2$
  - $e_6 \land e_5 \rightarrow e_4$

Constrains all contradictory cycles
- Constraints cache:
  - $e_2 \land e_3 \rightarrow e_1$
  - $e_4 \land e_5 \rightarrow e_2$
  - $e_6 \land e_3 \rightarrow e_4$

Results – random graphs
V=200, E=800, 16 random topologies
### Random graphs (Satisfiable)

<table>
<thead>
<tr>
<th>ratio</th>
<th>constraints</th>
<th>s.d.</th>
<th>Sparse</th>
<th>RTC</th>
<th>zChaff</th>
<th>Sparse</th>
<th>RTC</th>
<th>HalfSat</th>
<th>Sparse</th>
<th>RTC</th>
<th>siege_v4</th>
<th>Sparse</th>
<th>RTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:10</td>
<td></td>
<td>373068.8</td>
<td>161707.8</td>
<td>59.1</td>
<td>265.6</td>
<td>549.2</td>
<td>257.4</td>
<td>1521.6</td>
<td>556.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:1</td>
<td></td>
<td>373068.8</td>
<td>267826.6</td>
<td>5.2</td>
<td>0.4</td>
<td>5.9</td>
<td>3.0</td>
<td>1.2</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:1</td>
<td></td>
<td>373068.8</td>
<td>123623.4</td>
<td>0.1</td>
<td>0.01</td>
<td>0.6</td>
<td>0.22</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:1</td>
<td></td>
<td>373068.8</td>
<td>493.9</td>
<td>0.1</td>
<td>0.01</td>
<td>0.6</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:1</td>
<td></td>
<td>373068.8</td>
<td>10.3</td>
<td>0.1</td>
<td>0.01</td>
<td>0.6</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td></td>
<td>373,068.8</td>
<td>161,057.3</td>
<td>255.2</td>
<td>212.3</td>
<td>251.0</td>
<td>208.7</td>
<td>360.45</td>
<td>243.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>