Decision Procedures in First Order Logic

Decision Procedures for Equality Logic

Range Allocation

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:

$$u_1 = x_1 + y_1 \wedge u_2 = x_2 + y_2 \wedge z = u_1 \times u_2 \rightarrow$$
$$z = (x_1 + y_1) \times (x_2 + y_2)$$

To a formula with uninterpreted functions

$$u_1 = F(x_1, y_1) \wedge u_2 = F(x_2, y_2) \wedge z = G(u_1, u_2) \rightarrow$$
$$z = G(F(x_1, y_1), F(x_2, y_2))$$

Ackerman's reduction

From a formula with uninterpreted functions:

$$u_1 = F(x_1, y_1) \wedge u_2 = F(x_2, y_2) \wedge z = G(u_1, u_2) \rightarrow$$
$$z = G(F(x_1, y_1), F(x_2, y_2))$$

To a formula in the theory of equality

$$\begin{bmatrix} (x_1 = x_2 \wedge y_1 = y_2 \rightarrow f_1 = f_2) \wedge \\ (u_1 = f_1 \wedge u_2 = f_2 \rightarrow g_1 = g_2) \wedge \\ (u_1 = f_1 \wedge u_2 = f_2 \wedge z = g_1) \end{bmatrix} \rightarrow z = g_2$$

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 ϕ
- 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

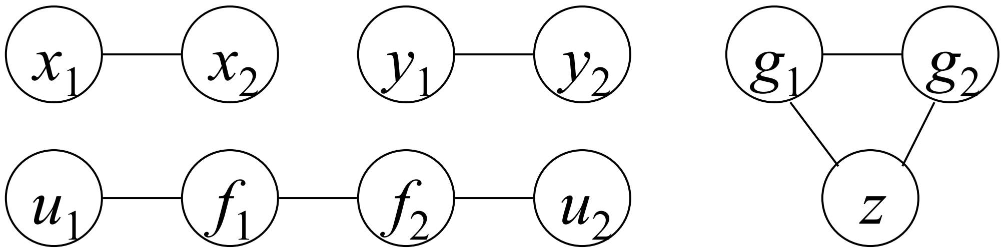
Complexity

- 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

$$\left[\begin{array}{l}(x_1 = x_2 \wedge y_1 = y_2 \rightarrow f_1 = f_2)\wedge \\ (u_1 = f_1 \wedge u_2 = f_2 \rightarrow g_1 = g_2)\wedge \\ (u_1 = f_1 \wedge u_2 = f_2 \wedge z = g_1)\end{array}\right] \rightarrow z = g_2$$

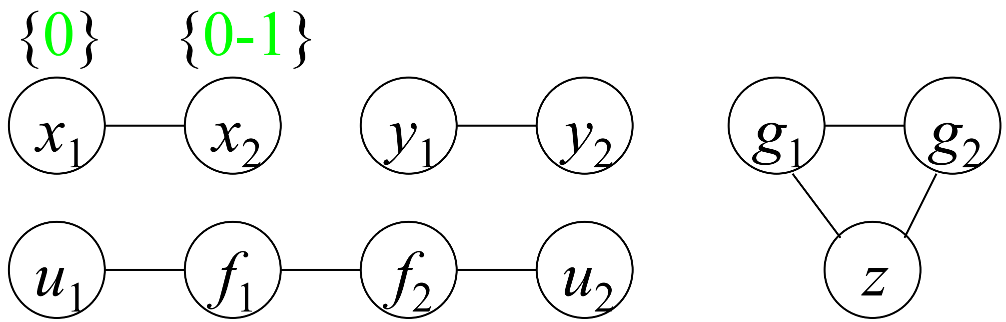
Instead of giving the range [1..11], analyze connectivity:



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

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 $\leq n!$