Transfer Function Synthesis
without Quantifier Elimination

Jörg Brauer and Andy King

RWTH Aachen University
Portcullis Computer Security
One instruction at a time abstraction by transfer function lookup
Transfer function synthesis with $\exists_x$ and $\forall_x$ [SAS’10]
Transfer function synthesis without $\exists_x$ and $\forall_x$
Stage 1

Feasible mode combinations
Consider the following:

1: ADD $R_0, R_1$
2: MOV $R_2, R_0$
3: EOR $R_2, R_1$
4: LSL $R_2$
5: SBC $R_2, R_2$
6: ADD $R_0, R_2$
7: EOR $R_0, R_2$

Implements $R_0' := \text{isign}(R_0+R_1,R_1)$ where $\text{isign}$ assigns $\text{abs}(R_0+R_1)$ to $R_0$ if $R_1 \geq 0$ and $-\text{abs}(R_0+R_1)$ otherwise.

Need to extract cases:

- Cases which are there by design: $R_1 \geq 0$
- Cases which are implementation artefacts: when $\text{abs}$ is applied to $-2^{31}$ then the result is $2^{31}$ subject to overflow which is $-2^{31}$
Let $\mu$ (mu) denote a Boolean encoding of ADD R0, R1 over bit-vectors $\{\vec{r}0, \vec{r}1, \ldots\}$ obtained through SSA and

$$
\begin{align*}
\mu_O &= \mu \land \neg \vec{r}0[31] \land \neg \vec{r}1[31] \land \vec{r}0'[31] \\
\mu_U &= \mu \land \vec{r}0[31] \land \vec{r}1[31] \land \neg \vec{r}0'[31] \\
\mu_E &= \mu \land (\vec{r}0[31] \lor \vec{r}1[31] \lor \neg \vec{r}0'[31]) \land (\neg \vec{r}0[31] \lor \neg \vec{r}1[31] \lor \vec{r}0'[31])
\end{align*}
$$

Let $\nu_O$ and $\nu_E$ (nu) express the overflow and exact modes of LSL R2.

In an analogous way to the first ADD, let $\eta_O$, $\eta_U$ and $\eta_E$ express the semantics of ADD R0, R2.
Using these encodings that satisfy a single mode, we can compose a formula for a fixed mode-combination.

The combination of $\mu_U$, $\nu_E$ and $\eta_E$ is infeasible.

The above block constitutes $3 \cdot 2 \cdot 3 = 18$ combinations of modes, but only five of which are satisfiable.

We derive a guard and update only for the feasible mode-combinations.
Synthesising guards
Consider the case where (1) underflows, (4) overflows and (6) is exact, with the corresponding formula denoted $\pi$.

To derive an octagonal guard for $\pi$, consider the problem of computing least $d$ such that $-\langle r\vec{0} \rangle - \langle r\vec{1} \rangle \leq d$.

Let $\kappa$ be a formula encodes $\langle \vec{d} \rangle = -\langle r\vec{0} \rangle - \langle r\vec{1} \rangle$ where $\vec{d}$ is signed and $\kappa$ is extended to 34 bits to prevent wraps.
Maximising $-2^{33} \leq d < 2^{33}$ bit-by-bit

- Then check:
  \[
  \psi^1 = \pi \land \kappa \land \neg \overrightarrow{d}[33]
  \]
- Satisfiability of $\psi^1$ shows $0 \leq d < 2^{33}$
- Then check:
  \[
  \psi^2 = \pi \land \kappa \land \neg \overrightarrow{d}[33] \land \overrightarrow{d}[32]
  \]
- Satisfiability of $\psi^2$ shows $2^{32} \leq d < 2^{33}$
- Then check:
  \[
  \psi^3 = \pi \land \kappa \land \neg \overrightarrow{d}[33] \land \overrightarrow{d}[32] \land \overrightarrow{d}[31]
  \]
- Unsatisfiability of $\psi^3$ shows $2^{32} \leq d < 2^{32} + 2^{31}$
- Continuing in this way we infer $2^{32} \leq d < 2^{32} + 1$. 
Repeating this tactic for all five feasible mode-combinations:

\[ g_{O(1), O(4), U(6)} = 2^{31} \leq \langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle \leq 2^{31} \quad \land \quad 0 \leq \langle \vec{r}_1 \rangle \leq 2^{31} - 1 \]

\[ g_{E(1), E(4), E(6)} = -2^{31} \leq \langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle \leq 2^{31} - 1 \]

\[ g_{U(1), O(4), E(6)} = -2^{32} \leq \langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle \leq -2^{31} - 1 \]

\[ g_{E(1), O(4), E(6)} = 0 \leq \langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle \leq 2^{31} - 1 \quad \land \quad -2^{31} \leq \langle \vec{r}_1 \rangle \leq 1 \]

\[ g_{O(1), O(4), E(6)} = 2^{31} + 1 \leq \langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle \leq 2^{32} \]

Redundant inequalities are omitted for clarity of presentation.
Synthesising updates
Consider ADD R0 R1; LSL R0 in exact modes

We want an update to map octagonal input constraints with symbolic constants to octagonal outputs with symbolic constants:

\[
\begin{align*}
\langle \vec{r}_0 \rangle & \leq d_1 \\
\langle \vec{r}_1 \rangle & \leq d_2 \\
-\langle \vec{r}_0 \rangle & \leq d_3 \\
-\langle \vec{r}_1 \rangle & \leq d_4 \\
\langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle & \leq d_5 \\
-\langle \vec{r}_0 \rangle - \langle \vec{r}_1 \rangle & \leq d_6 \\
-\langle \vec{r}_0 \rangle + \langle \vec{r}_1 \rangle & \leq d_7 \\
\langle \vec{r}_0 \rangle - \langle \vec{r}_1 \rangle & \leq d_8 \\
\end{align*}
\]  
\[
\begin{align*}
\langle \vec{r}_0' \rangle & \leq 2d_5 \\
\langle \vec{r}_1' \rangle & \leq d_2 \\
-\langle \vec{r}_0' \rangle & \leq 2d_6 \\
-\langle \vec{r}_1' \rangle & \leq d_4 \\
\langle \vec{r}_0' \rangle + \langle \vec{r}_1' \rangle & \leq 2d_5 + d_2 \\
-\langle \vec{r}_0' \rangle - \langle \vec{r}_1' \rangle & \leq 2d_6 + d_4 \\
-\langle \vec{r}_0' \rangle + \langle \vec{r}_1' \rangle & \leq 2d_6 + d_2 \\
\langle \vec{r}_0' \rangle - \langle \vec{r}_1' \rangle & \leq 2d_5 + d_4 \\
\end{align*}
\]
Consider $\langle \vec{r} \vec{0}' \rangle \leq d'_1$ and the problem of discovering a relationship between $d'_1$ and $d_1, \ldots, d_8$

- Let $\vec{d}_1, \ldots, \vec{d}_8$ denote signed 34-bit vectors that represent the symbolic constants $d_1, \ldots, d_8$
- Let $\kappa$ denote a formula that holds iff the 8 inequalities $\langle \vec{r} \vec{0} \rangle \leq \langle \vec{d}_1 \rangle, \ldots, \langle \vec{r} \vec{0} \rangle - \langle \vec{r} \vec{1} \rangle \leq \langle \vec{d}_8 \rangle$ simultaneously hold
- Let $\pi$ denote a propositional encoding for ADD R0 R1; LSL R0 operating in exact mode
- Let $\rho$ encode the equality $\langle \vec{r} \vec{0}' \rangle = \langle \vec{d}'_1 \rangle$ where $\vec{d}'_1$ is a signed bit-vector representing $d'_1$
Step i: solving and maximisation

- Present $\kappa \land \pi \land \rho$ to a SAT solver and find a model:

$$m_1 = \left\{ \langle \vec{d}'_1 \rangle = 1, \langle \vec{d}_1 \rangle = 1, \langle \vec{d}_2 \rangle = 1, \ldots, \langle \vec{d}_7 \rangle = 1, \langle \vec{d}_8 \rangle = 1 \right\}$$

- $\langle \vec{d}'_1 \rangle = 1$ may not be maximum for $\langle \vec{d}_1 \rangle = 1, \ldots, \langle \vec{d}_8 \rangle = 1$

- Let $\zeta$ denote a formula that holds iff $\langle \vec{d}_1 \rangle = 1, \ldots, \langle \vec{d}_8 \rangle = 1$

- Apply dichotomic search to find the maximal value of $\langle \vec{d}'_1 \rangle$ subject to $\kappa \land \pi \land \rho \land \zeta$.

- This gives the model:

$$m'_1 = \left\{ \langle \vec{d}'_1 \rangle = 2, \langle \vec{d}_1 \rangle = 1, \langle \vec{d}_2 \rangle = 1, \ldots, \langle \vec{d}_7 \rangle = 1, \langle \vec{d}_8 \rangle = 1 \right\}$$
Suppose the matrix $M_1$ is constructed from $m_1'$ by using the variable ordering $\langle d_1', d_1, \ldots, d_8 \rangle$ on columns:

$$M_1 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Let $\mu$ denote a formula that holds iff $\langle \overrightarrow{d_8} \rangle \neq 1$ holds
Step ii: solving and maximisation

- Present $\kappa \land \pi \land \rho \land \mu$ to a SAT solver and find a model:

$$m_2 = \{ \langle \vec{d}_1' \rangle = 8, \langle \vec{d}_1 \rangle = 3, \langle \vec{d}_2 \rangle = 3, \ldots, \langle \vec{d}_7 \rangle = 2, \langle \vec{d}_8 \rangle = 0 \}$$

- $\langle \vec{d}_1' \rangle = 8$ may not be maximum for $\langle \vec{d}_1 \rangle = 3, \ldots, \langle \vec{d}_8 \rangle = 0$

- Let $\zeta$ denote a formula that holds iff $\langle \vec{d}_1 \rangle = 3, \ldots, \langle \vec{d}_8 \rangle = 0$

- Apply dichotomic search to find the maximal value of $\langle \vec{d}_1' \rangle$ subject to $\kappa \land \pi \land \rho \land \zeta$.

- This gives the model:

$$m'_2 = \{ \langle \vec{d}_1' \rangle = 10, \langle \vec{d}_1 \rangle = 3, \langle \vec{d}_2 \rangle = 3, \ldots, \langle \vec{d}_7 \rangle = 2, \langle \vec{d}_8 \rangle = 0 \}$$
Merging [Karr’76]

The model $m'_2$ is interpreted as a matrix:

$$M_2 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

The merge $M_1 \sqcup M_2$ is as follows:

$$M_1 \sqcup M_2 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}$$
Step iii and iv: solving and maximisation

- Let $\mu$ now denote a formula that holds iff $\langle\vec{d}_7\rangle + \langle\vec{d}_8\rangle \neq 1$
- Presenting $\kappa \land \pi \land \rho \land \mu$ to a solver gives:
  
  $$m_3 = \begin{cases} 
  \langle\vec{d}_1'\rangle = 22, \langle\vec{d}_1\rangle = 8, \ldots, \langle\vec{d}_8\rangle = 0 
  \end{cases}$$

- Maximising $\langle\vec{d}_1'\rangle$ then gives:
  
  $$m_3' = \begin{cases} 
  \langle\vec{d}_1'\rangle = 26, \langle\vec{d}_1\rangle = 8, \ldots, \langle\vec{d}_8\rangle = 0 
  \end{cases}$$

- Form $M_3$ and calculate another merge:
  
  $$M_1 \sqcup M_2 \sqcup M_3 = \begin{b matrix} 
  1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\
  0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
  \end{b matrix}$$

- Repeating

  $$M_1 \sqcup M_2 \sqcup M_3 \sqcup M_4 = \begin{b matrix} 
  1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\
  \end{b matrix}$$

- Conclude $d_1' = 2d_5$
**“With” versus “without” for intervals**

<table>
<thead>
<tr>
<th>block</th>
<th>insts</th>
<th>bits</th>
<th>guards/SAT</th>
<th>affine/SAT</th>
<th>overall</th>
<th>SAS’10</th>
</tr>
</thead>
<tbody>
<tr>
<td>inc</td>
<td>1</td>
<td>8</td>
<td>0.2s / 40</td>
<td>0.1s / 5</td>
<td>0.3s</td>
<td>0.2s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>0.5s / 136</td>
<td>0.2s / 5</td>
<td>1.0s</td>
<td>23.0s</td>
</tr>
<tr>
<td>shifter</td>
<td>2</td>
<td>8</td>
<td>0.3s / 60</td>
<td>0.1s / 8</td>
<td>0.4s</td>
<td>0.3s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>0.8s / 216</td>
<td>0.2s / 8</td>
<td>1.0s</td>
<td>∞</td>
</tr>
<tr>
<td>swap</td>
<td>3</td>
<td>8</td>
<td>—</td>
<td>0.1s / 3</td>
<td>0.1s</td>
<td>0.1s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>—</td>
<td>0.1s / 3</td>
<td>0.1s</td>
<td>0.2s</td>
</tr>
<tr>
<td>flipper</td>
<td>4</td>
<td>8</td>
<td>0.2s / 40</td>
<td>0.2s / 5</td>
<td>0.4s</td>
<td>0.5s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>0.9s / 216</td>
<td>0.3s / 5</td>
<td>1.2s</td>
<td>∞</td>
</tr>
<tr>
<td>abs</td>
<td>5</td>
<td>8</td>
<td>2.5s / 216</td>
<td>0.3s / 8</td>
<td>2.8s</td>
<td>0.8s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>6.5s / 792</td>
<td>0.3s / 8</td>
<td>6.8s</td>
<td>∞</td>
</tr>
<tr>
<td>isign</td>
<td>7</td>
<td>8</td>
<td>4.1s / 360</td>
<td>0.2s / 18</td>
<td>4.3s</td>
<td>4.5s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>10.7s / 1320</td>
<td>0.4s / 18</td>
<td>11.1s</td>
<td>∞</td>
</tr>
<tr>
<td>absolute</td>
<td>10</td>
<td>8</td>
<td>2.8s / 216</td>
<td>0.3s / 8</td>
<td>3.1s</td>
<td>9.5s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>7.2s / 792</td>
<td>0.3s / 8</td>
<td>7.5s</td>
<td>∞</td>
</tr>
</tbody>
</table>