Symmetric Synthesis

Rüdiger Ehlers

University of Bremen & DFKI GmbH

Based on joint work with Bernd Finkbeiner

Towards Scalable Formal Synthesis of Complex Systems
workshop, CDC 2015
(Monolithic) Synthesis of Reactive Systems

\[ GF(u \land v) \rightarrow G(u \leftrightarrow X v) \]
\[ + \]

Input = \{u, \ldots\}
Output = \{v, \ldots\}

⇒ Realizable
⇒ Not realizable

Input

Output
Distributed Synthesis

The specification for synthesis ranges over $u$, $v$, $y$, and $z$. 
Monolithic Reactive Synthesis vs. Distributed Synthesis

<table>
<thead>
<tr>
<th>Monolithic Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity for CTL: EXPTIME [Emerson and Clarke, 1982]</td>
</tr>
<tr>
<td>Complexity for GR(1): EXPTIME</td>
</tr>
<tr>
<td>Complexity for LTL: 2EXPTIME [Pnueli and Rosner, 1989a,b]</td>
</tr>
</tbody>
</table>
Monolithic Reactive Synthesis vs. Distributed Synthesis

Distributed Synthesis [Finkbeiner and Schewe, 2005]

- Complexity for LTL & architectures without information fork: **decidable**

- Complexity for LTL & architectures with an information fork: **undecidable**

![Diagram of distributed synthesis systems](image)
Symmetric Synthesis – An Example Architecture

u \rightarrow a \leftarrow b \rightarrow c \rightarrow v

w \rightarrow d \leftarrow One

s_1 \rightarrow \neg d \rightarrow Two \rightarrow z
Example Application
Why Symmetric Systems?

- They are easier to maintain and easier to implement
- They can often be smaller
- They are often easier to understand.
- Their existence/inexistence tells us something about the application
- They are a big step towards synthesis of *arbitrarily scalable systems*
- They are somewhat robust to failure of individual processes
Central Question

Central question: In which cases is symmetric synthesis decidable and how efficiently can it be performed?
Results in This Talk

Undecidable Symmetric Synthesis

- All architectures that contain the S0 architecture have an undecidable symmetric synthesis problem.
  → These are pretty much all architectures that have internal communication

Decidable Symmetric Synthesis

- All rotation-symmetric architectures have a decidable synthesis problem
  → For GR(1)-like specifications, it can be solved by GR(1) synthesis
Undecidable Symmetric Synthesis
A Simple Undecidable Symmetric Architecture

The S0 Architecture
Another Undecidable Symmetric Architecture (S2)
Another Undecidable Symmetric Architecture (S2)
Compressing an Input/Output Stream
Undecidability of the Example Architecture
Undecidability of the Example Architecture
Decidable Symmetric Synthesis
A Decidable Symmetric Architecture

\[(P_0, y) \quad (P_1, y) \quad (P_2, y)\]
Computation Trees of Rotation-symmetric Systems

\[
\begin{align*}
P_0 & \quad P_1 \\
(0,0) & \quad (0,0) \\
(0,1) & \quad (0,1) \\
(1,0) & \quad (1,0) \\
(1,1) & \quad (1,1)
\end{align*}
\]
Characterisation of Rotation-symmetric Computation Trees

Necessary and Sufficient Conditions
Characterisation of Rotation-symmetric Computation Trees

Necessary and Sufficient Conditions

No spontaneous introduction of asymmetry
Normalization of the system behaviour along different rotations of the input
Characterisation of Rotation-symmetric Computation Trees

Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry
Characterisation of Rotation-symmetric Computation Trees

Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry
Characterisation of Rotation-symmetric Computation Trees

Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry
- Normalization of the system behaviour along different rotations of the input
Enforcing the **Symmetry Property**

Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry
- Normalization of the system behaviour along different rotations of the input
Enforcing the **Symmetry Property**

### Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry $\Rightarrow$ **Regular Property**
- Normalization of the system behaviour along different rotations of the input
Enforcing the *Symmetry Property*

Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry $\Rightarrow$ *Regular Property*
- Normalization of the system behaviour along different rotations of the input $\Rightarrow$ *Non-regular Property!*
Enforcing the Symmetry Property

Necessary and Sufficient Conditions

- No spontaneous introduction of asymmetry $\Rightarrow$ Regular Property
- Normalization of the system behaviour along different rotations of the input $\Rightarrow$ Non-regular Property!
Symmetrizing the specification

Idea

Starting with a specification $\psi$, we modify it to:

$$\psi \land \text{rot}(\psi, 1) \land \ldots \land \text{rot}(\psi, n - 1)$$
Symmetrizing the specification

**Idea**
Starting with a specification $\psi$, we modify it to:

$$\psi \land rot(\psi, 1) \land \ldots \land rot(\psi, n - 1)$$

**Example with Three Processes**
Starting with $G(a_1 \leftrightarrow Xb_2)$, we obtain

$$G(a_1 \leftrightarrow Xb_2) \land G(a_2 \leftrightarrow Xb_0) \land G(a_0 \leftrightarrow Xb_1)$$
Rotation-symmetric Synthesis Algorithm (LTL)

Steps

1. Symmetrize the specification
2. Translate to word automaton $\mathcal{A}$
3. Compose $\mathcal{A}$ with a word automaton to reject words with asymmetry introduction
4. Spread the word automaton to a tree automaton $\mathcal{T}$.
5. Check emptiness of $\mathcal{L}(\mathcal{T})$.
6. *(Not empty:) Symmetrize tree and chop off outputs of other processes*
Complexity of the Symmetric Synthesis Algorithms

LTL and CTL*: EXPTIME in # of processes, 2EXPTIME in size of the specification
Complexity

### Complexity of the Symmetric Synthesis **Algorithms**

| LTL and CTL* | EXPTIME in # of processes, 2EXPTIME in size of the specification |

### Hardness of the Symmetric Synthesis **Problem**

| LTL and CTL* | EXPTIME in # of processes, 2EXPTIME in size of the specification |
More Complex Rotation-Symmetric Architectures

\[ x_0 \]

\[ x_1 x_2 \]

\[ x_3 \]

\[ P_{0,0} \]

\[ P_{1,0} \]

\[ (P_{0,0}, y) \] \( (P_{0,1}, y) \)

\[ (P_{1,0}, y) \] \( (P_{1,1}, y) \)
Symmetric GR(1) Synthesis

Main Steps for Symmetric Synthesis

- Symmetrizing the specification
  → not quite possible, but we can approximate that well
- Adding system guarantees that require the system not to introduce asymmetry in the output
  → easy to encode in GR(1)
Benchmarks Considered (1/2) – Traffic Light
Benchmarks Considered (2/2) – Rotation Sorter
Benchmarks

Setting

- BDD-based game solver
- AMD E-450 1.6GHz, x86-Linux, 4GB RAM

Symmetric Traffic Light

- Basic setting: unrealizable (1.1s)
- With simple symmetry breaking assumption:
  \( F((\text{carSensed}_0 \lor \text{carSensed}_1 \lor \text{carSensed}_2 \lor \text{carSensed}_3) \leftrightarrow \neg(\text{carSensed}_0 \land \text{carSensed}_1 \land \text{carSensed}_2 \land \text{carSensed}_3)) \)
  realizable (1.2)
- With additional exclusiveness guarantee:
  \( G(\neg\text{green}_0 \lor \neg\text{green}_2) \land G(\neg\text{green}_1 \lor \neg\text{green}_3) \)
  unrealizable (1.25s)
## Benchmarks

<table>
<thead>
<tr>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BDD-based game solver</strong></td>
</tr>
<tr>
<td><strong>AMD E-450 1.6GHz, x86-Linux, 4GB RAM</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symmetric Rotation Sorter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic setting:</strong> <strong>unrealizable</strong> (130s)</td>
</tr>
<tr>
<td><strong>With packet consistency assumption:</strong> <strong>realizable</strong> (16 $\frac{1}{3}$ min)</td>
</tr>
</tbody>
</table>
Conclusion

Symmetric Synthesis
- Many architectures have an undecidable synthesis problem
- However, rotation-symmetric architectures have a decidable synthesis problem

Efficiency
- Symmetric synthesis for rotation-symmetric architectures can be performed relatively efficiently
  - time complexity is exponential in the number of processes
References I


