Parameterized Synthesis

Swen Jacobs    TU Graz

joint work with Roderick Bloem and Ayrat Khalimov
(and Sasha Rubin, Benjamin Aminof – IST Austria)

24 July 2014 – FRIDA Workshop, Vienna
Problem: Synthesis of Systems with Parametric Size

- Parametric number of components
- Parametric number of communication partners

Hard to get correct manually

Costly to synthesize
AMBA Advanced High-Performance Bus

- Bus controller with locked accesses, bursts, and other features
- „Standard“ synthesis benchmark
- Parameterized in number of masters accessing the bus
Synthesizing AMBA for Increasing Size

AMBA: Communication Bus, „standard“ synthesis benchmark

Runtimes grow very fast

Circuit size grows very fast
Idea: Synthesis of Replicable Building Blocks

• Can we synthesize building blocks for arbitrary size systems?

How to do Parameterized Synthesis?
Parameterized Synthesis [TACAS12,LMCS14]

Builds on:

• Reduction results for parameterized verification
  - cutoffs reduce correctness of parameterized systems to fixed-size systems
• Distributed synthesis (for fixed number of processes)

Challenges:

- synthesis must guarantee that systems satisfy conditions of reduction
- expressiveness of supported specifications and architectures
- scalability
Parameterized Verification

Parameterized verification is **decidable** for certain systems

**Theorem** [EN95]:

In **token rings** with fair token passing, a given process implementation satisfies parameterized specification $\varphi$ in $\text{LTL} \setminus \mathcal{X}$ iff it satisfies $\varphi$ in a ring of small size.

**Corollary:** For parameterized synthesis, it is sufficient to synthesize a process implementation satisfying $\varphi$ (and Thm. conditions) in a small ring.
Cutoff Results – Proof Idea

• $\forall i. \varphi(i)$: establish stuttering bisimulation between every process in a big system and a process in system of size 2

• $\forall i, j. \varphi(i, j)$:
  1. Define finite partition of all pairs of processes in big ring: are i and j neighbors? is i left or right of j?
  2. Define mapping between pairs of processes in big and size 4 system
  3. Establish stuttering bisimulation for every partition

Conditions:
• fair token passing: fair scheduling, and processes must release token
• stuttering-insensitive specifications, i.e., $\text{LTL}\setminus X$
Distributed Synthesis

Problem undecidable under partial information [PR90,FS05]; also in token-passing networks [TACAS12,LMCS14]
Bounded Synthesis [SF07]

LTL Specification

Architecture

(via automata)

SMT (bounded size)

SMT (with quantifiers)

SMT (decidable)

Model of SMT constraints represents implementation of specification

Swen Jacobs
Parameterized Synthesis
Parameterized Synthesis: Procedure

1. **Use cutoff** to reduce parameterized synthesis problem to distributed synthesis problem

2. **Modify SMT encoding** (from bounded synthesis) to ensure satisfaction of reduction conditions:
   - uniform processes
   - interleaving semantics
   - in a token ring architecture
   - with fair scheduling and fair token-passing

3. **Solve** problem with SMT solver (for increasing bounds)
Parameterized Synthesis: Procedure
First Experiments

Can synthesize distributed arbiter in token ring of 4 processes with spec

\[
\forall i: \text{G}(r_i \rightarrow \text{F}g_i)
\]
\[
\forall i \neq j: \text{G} \neg(g_i \land g_j)
\]

This takes Z3 about 10 sec.

**But**: problem gets **hard** very fast.

For extended spec with

\[
\forall i: \neg g_i \text{U} r_i \land \text{G}(g_i \rightarrow \neg g_i \text{U} r_i)
\]

needs about 240 sec.
Benefits of Parameterized Synthesis

![Graph showing the relationship between synthesis time and number of processes for basic and extended methods. The graph indicates that as the number of processes increases, the synthesis time also increases, with the extended method showing a steeper curve compared to the basic method.](image)
Towards Efficient Parameterized Synthesis [VMCAI13]

Optimization: Modular Cutoffs

Standard approach:

\[ \forall i, \varphi_1(i) \land \forall i, j, \varphi_2(i, j) \]

Most of the specification is usually in local part \( \forall i, \varphi_1(i) \)

SMT Constraint \( C_1 \)
SMT Constraint \( C_2 \)
SMT Constraint \( C \)

solve \( C_1 \land C_2 \)
Towards Efficient Parameterized Synthesis [VMCAI13]

• Strengthening (incomplete):

\[ A_S \land A_L \rightarrow G_S \land G_L \]

\[ (A_S \land A_L \rightarrow G_L) \land (A_S \rightarrow G_S) \]

- \( A_S \): safety assumptions
- \( A_L \): liveness assumptions
- \( G_S \): safety guarantees
- \( G_L \): liveness guarantees

Table 2: Effect of general optimizations on solving time (in seconds). Timeout is 7200s.

<table>
<thead>
<tr>
<th></th>
<th>simple4</th>
<th>full2</th>
<th>full3</th>
<th>full4</th>
<th>pnueli2</th>
<th>pnueli3</th>
<th>pnueli4</th>
<th>pnueli5</th>
<th>pnueli1</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottom up</td>
<td>3</td>
<td>24</td>
<td>934</td>
<td>t/o</td>
<td>23</td>
<td>6737</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strengthening</td>
<td>1</td>
<td>6</td>
<td>81</td>
<td>638</td>
<td>2</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>modular</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>13</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Implemented in PARTY [CAV13]
Extensions of Cutoff Results [VMCAI14]

• Several new cutoff results for token-passing networks
• In particular:

**Theorem:**
In token rings with fair token passing, for a parameterized specification
\( \forall i_1, \ldots, i_n, \varphi(i_1, \ldots, i_n) \) in \( \text{LTL} \setminus X \),
**2n is a cutoff.**

In theory, allows properties of tuples of arbitrary size
Parameterized AMBA Case Study [SYNT2014]

- simplified version as considered in existing approaches [J07]
- Synchronous system
- $m_0$ gets the bus by default

```
m_0 \quad \ldots \quad m_n
\quad \text{bus}
\quad \ldots \quad \text{Arbiter}
```

$\text{HBUSREQ}[i]$, $\text{HLOCK}[i]$, $\text{HBURST}$

$\text{HGRANT}[i]$, etc.

$\text{HREADY}$
Parameterized AMBA Case Study [SYNT2014]

Adressed several challenges to make synthesis possible:

• **extension of cutoff results** to fully asynchronous systems
  correctness in asynchronous $\Rightarrow$ correctness in synchronous

• global in- and output signals make PMCP **undecidable**

  **strengthening** by localization of signals and assumptions
  (incomplete)

• different process implementations for 0- and non-0-processes
Scalability:

• domain-specific encoding
• decomposition of synthesis wrt. environment assumptions:

1. Assume all requests are BURST4
2. Add non-locked request
3. Add INCR burst
## Parameterized AMBA Case Study [SYNT2014]

### Table 1: Results for non-0-process.

<table>
<thead>
<tr>
<th>Additional assumptions</th>
<th>time</th>
<th>#states</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GHLOCK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GHBURST = BURST4</strong></td>
<td>16min.</td>
<td>10</td>
</tr>
<tr>
<td><strong>GHBURST = BURST4</strong></td>
<td>13sec.</td>
<td>13</td>
</tr>
<tr>
<td>– (Full Specification)</td>
<td>1min.</td>
<td>14</td>
</tr>
</tbody>
</table>

### Table 2: Results for 0-process (bursts reduced: 3/4 → 2/3).

<table>
<thead>
<tr>
<th>Additional assumptions</th>
<th>time</th>
<th>#states</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GHLOCK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GHBURST = BURST4</strong></td>
<td>3h.</td>
<td>11</td>
</tr>
<tr>
<td><strong>GHBURST = BURST4</strong></td>
<td>1min.</td>
<td>11</td>
</tr>
<tr>
<td>– (Full Specification)</td>
<td>1m30s.</td>
<td>12</td>
</tr>
</tbody>
</table>
Parameterized Synthesis Framework

In general:
obtain **semi-decision procedure for parameterized synthesis** from
  • cutoff result for verification
  • encoding into specialized bounded synthesis problem

Thus far:
  • integrated and extended cutoff results for **token-passing systems**
    [TACAS12,LMCS14,VCMAI14]
  • implemented with **optimizations and complete automation**
    [VMCAI13,CAV13]
  • applied to AMBA case study [SYNT14]
Parameterized Synthesis: Challenges

Make applicable to systems with

• other communication primitives (submitted!)
• failure-resilience (next talk!)

Many cutoff results are essentially restricted to safety properties. Need to be extended to include fairness, liveness properties

Thanks!