Synthesis from Quantitative Specifications

Arjun Radhakrishna

Institute of Science and Technology, Austria

March 11, 2012

Joint work with Pavol Černý, Thomas A. Henzinger, Sivakanth Gopi, and Nishanth Totla

This research was supported in part by the European Research Council (ERC) Advanced Investigator Grant QUAREM and by the Austrian Science Fund (FWF) project S11402-N23.
Boolean Specifications vs. Quantitative Specifications

Boolean Specifications

Bad | Good
Boolean Specifications vs. Quantitative Specifications

Preference

Boolean Specifications

Quantitative Specifications

Preference

Bad

Good

Many formalisms: Weighted Automata, Quantitative logics, Cost-Register Automata, Software Metrics, etc.

In this talk: Reactive systems + Behavioural metrics
Boolean Specifications vs. Quantitative Specifications

- **Boolean Specifications**
  - Bad
  - Good

- **Quantitative Specifications**

- Yes/No vs. Preference Order

Many formalisms: Weighted Automata, Quantitative logics, Cost-Register Automata, Software Metrics, etc.

In this talk: Reactive systems + Behavioural metrics
Boolean Specifications vs. Quantitative Specifications

- Yes/No vs. Preference Order
- Many formalisms: Weighted Automata, Quantitative logics, Cost-Register Automata, Software Metrics, etc
Boolean Specifications vs. Quantitative Specifications

- **Yes/No vs. Preference Order**
- **Many formalisms:** Weighted Automata, Quantitative logics, Cost-Register Automata, Software Metrics, etc
- **In this talk:** Reactive systems + Behavioural metrics
Our formalism: Simulation distances

- Specification: “Ideal” boolean specification

\[ d_E(I_1, S) \leq d_E(I_2, S), \text{ then } I_1 \text{ is preferred over } I_2 \]

\footnote{[R. Milner. 1971]}
Our formalism: Simulation distances

- Specification: “Ideal” boolean specification + Error Model

\[ d^E(I, S) \] \[ \overset{\text{if}}{\rightarrow} \]

Our formalism: Simulation distances

- Specification: “Ideal” boolean specification + Error Model

- Extend the classical Simulation relation\(^1\) to Simulation distances

\(^1\)[R. Milner. 1971]
Our formalism: Simulation distances

- Specification: “Ideal” boolean specification + Error Model

- Extend the classical Simulation relation\(^1\) to Simulation distances

- Written as \(d_{\mathcal{E}}(\mathcal{I}, S)\)
  - if \(d_{\mathcal{E}}(\mathcal{I}_1, S) < d_{\mathcal{E}}(\mathcal{I}_2, S)\), then \(\mathcal{I}_1\) is preferred over \(\mathcal{I}_2\)

---

\(^1\)[R. Milner. 1971]
Our formalism: Simulation distances

Implementation
Specification
Error Penalty
Our formalism: Simulation distances

Ideal Specification

Error Model

Implementation

Implementation

Specification

Error Penalty

Limit-Average Simulation Distance = 1 / 4 = 0.25
Our formalism: Simulation distances

Ideal Specification

Error Model

Implementation

Implementation

Specification

Error Penalty

0
Our formalism: Simulation distances

**Ideal Specification**

- Transition: $a \rightarrow 0$
- Transition: $b \rightarrow 1$
- Transition: $b \rightarrow 2$
- Transition: $a \rightarrow 1$

**Error Model**

- Transition: $a(0) \rightarrow 0$
- Transition: $b(0) \rightarrow 0$
- Transition: $a(1) \rightarrow 0$
- Transition: $b(1) \rightarrow 0$

**Implementation**

- Transition: $a \rightarrow 0$
- Transition: $b \rightarrow 1$
- Transition: $b \rightarrow 2$
- Transition: $a \rightarrow 3$

Implementation: $b \ b$

Specification: $b \ b$

Error Penalty: 0 0
Our formalism: Simulation distances

Ideal Specification

Error Model

Implementation

Implementation  |  b  |  b  |  b  
Specification    |  b  |  b  |  a  
Error Penalty    |  0  |  0  |  1  

Limit-Average Simulation Distance = 1 / 4 = 0.25
Our formalism: Simulation distances

Ideal Specification

Error Model

Implementation

Implementation

Specification

Error Penalty

Limit-Average Simulation Distance = $\frac{1}{4} = 0.25$
Our formalism: Simulation distances

Ideal Specification

Error Model

Implementation

Implementation: b b b a ...  
Specification: b b a a a ...  
Error Penalty: 0 0 1 0 ...  

Limit-Average Simulation Distance = 1 / 4 = 0.25
Error Models and Properties

Boolean Error Model

\[
\begin{align*}
\frac{a}{a}(0) & \\
\frac{a}{b}(\infty) & \circlearrowleft \\
\frac{b}{b}(0) & \circlearrowright \\
\frac{b}{a}(\infty) & \\
\end{align*}
\]
Delayed Grant Model – penalizes delay in grant

\[ g/g(0) \quad g/\ast(0) \quad \tilde{g}/g(1) \quad \tilde{g}/\ast(1) \]
Grant Efficiency Error Model – penalizes spurious grants
Every request \( req \) to be eventually granted with \( gr \). \( \Phi = G ( req \implies Fgr ) \).

---

\(^2\)Taken from [I. Pill, S. Semprini, R. Cavada, M. Roveri, R. Bloem, and A. Cimatti. FMCAD 2009]
Why use quantitative specifications? – Example

Every request \( \text{req} \) to be \textit{eventually} granted with \( \text{gr} \). \( \Phi = G(\text{req} \implies F\text{gr}) \).

- Additional desire \( D_1 \): We want to minimize spurious grants\(^2\).
  - \( D_1 = \neg\text{gr} \land \text{req} \land G(\text{gr} \implies X(\neg\text{gr} \land \text{req})) \).
  - Simple requirement, but complicated to specify.

\(^2\)Taken from [I. Pill, S. Semprini, R. Cavada, M. Roveri, R. Bloem, and A. Cimatti. FMCAD 2009]
Why use quantitative specifications? – Example

Every request \( \text{req} \) to be eventually granted with \( \text{gr} \). \( \Phi = G(\text{req} \implies F\text{gr}) \).

- Additional desire \( D_1 \): We want to minimize spurious grants\(^2\).
  - \( D_1 = \neg \text{gr} \wedge \text{req} \wedge G(\text{gr} \implies X(\neg \text{gr} \wedge \text{req})) \).
  - Simple requirement, but complicated to specify.

- Changing requirement \( \Phi \) to \( \Phi \wedge GF(\text{gr}) \) – full rewriting of \( D_1 \).

\(^2\)Taken from [I. Pill, S. Semprini, R. Cavada, M. Roveri, R. Bloem, and A. Cimatti. FMCAD 2009]
Why use quantitative specifications? – Example

Every request \( \text{req} \) to be **eventually** granted with \( \text{gr} \). \( \Phi = G(\text{req} \implies \text{Fgr}) \).

- Additional desire \( D_1 \): We want to minimize spurious grants\(^2\).
  - \( D_1 = \neg \text{gr} \land \text{req} \land G(\text{gr} \implies X(\neg \text{gr} \land \text{req})) \).
  - Simple requirement, but **complicated to specify**.

- Changing requirement \( \Phi \) to \( \Phi \land GF(\text{gr}) \) – full rewriting of \( D_1 \).

- Quantitative case: add one ideal specification saying "**no grants**", and error model penalizing each grant.
  - No change is \( \Phi \) changes
  - Specifying **What** instead of **How**

\(^2\)Taken from [I. Pill, S. Semprini, R. Cavada, M. Roveri, R. Bloem, and A. Cimatti. FMCAD 2009]
Given specification-error model pairs \((S_i, E_i)\) and weights \(\mu_i \in (0, 1)\), find \((\epsilon-)\)optimal implementation \(I^*\) such that \(\max_i \{\mu_i \cdot d_{E_i}(I^*, S_i)\}\) is minimized.
Given specification-error model pairs \((S_i, \mathcal{E}_i)\) and weights \(\mu_i \in (0, 1)\), find \((\epsilon\text{-})optimal implementation \(\mathcal{I}^*\) such that \(\max_i \{\mu_i \cdot d_{\mathcal{E}_i}(\mathcal{I}^*, S_i)\}\) is minimized.

\[
\mu_1 = \mu_2
\]
Given specification-error model pairs \( (S_i, \mathcal{E}_i) \) and weights \( \mu_i \in (0, 1) \), find \((\epsilon-)\)optimal implementation \( \mathcal{I}^* \) such that \( \max_i \{ \mu_i \cdot d_{\mathcal{E}_i}(\mathcal{I}^*, S_i) \} \) is minimized.

\[ \mu_1 < \mu_2 \]
Composing Requirements – Synthesis on the Pareto Curve

Given specification-error model pairs \((S_i, E_i)\) and weights \(\mu_i \in (0, 1)\), find \((\epsilon-)\)optimal implementation \(I^*\) such that \(\max_i \{\mu_i \cdot d_{E_i}(I^*, S_i)\}\) is minimized.

\[
\mu_1 > \mu_2
\]

\[
S_2 \quad \mu_2 \quad I \quad \mu_1 \quad S_1
\]
Given specification-error model pairs \((S_i, \mathcal{E}_i)\) and weights \(\mu_i \in (0, 1)\), find \((\epsilon-)\)optimal implementation \(I^*\) such that \(\max_i \{\mu_i \cdot d_{\mathcal{E}_i}(I^*, S_i)\}\) is minimized.
Composing Requirements – Synthesis on the Pareto Curve

Given specification-error model pairs \((S_i, E_i)\) and weights \(\mu_i \in (0,1)\), find \((\epsilon-)optimal implementation \(I^*\) such that \(\max_i \{\mu_i \cdot d_{E_i}(I^*, S_i)\}\) is minimized.

Solution: \(I^*\) is the \((\epsilon-)optimal finite memory strategy in a multi-dimensional mean-payoff game [K. Chatterjee. 30 minutes ago.]
Protocol Trade-offs: Forward Error Correcting Codes

- FECs are protocols for error control in noisy channels.

- Our problem – Send 3 bit integers over a network
  - Say one bit-flip during transmission.
  - Additional complexity: Error in the MSB is worse than an error in LSB.
Protocol Trade-offs: Forward Error Correcting Codes

- FECs are protocols for error control in noisy channels.

- Our problem – Send 3 bit integers over a network
  - Say one bit-flip during transmission.
  - Additional complexity: Error in the MSB is worse than an error in LSB.

- Two conflicting quantitative requirements:
  - Efficiency: Use least bandwidth.
    - Ideal Specification: Only 3 bits transferred.
    - Error model: Penalty 1 per additional bit.
Protocol Trade-offs: Forward Error Correcting Codes

• FECs are protocols for error control in noisy channels.

• Our problem – Send 3 bit integers over a network
  ▶ Say one bit-flip during transmission.
  ▶ Additional complexity: Error in the MSB is worse than an error in LSB.

• Two conflicting quantitative requirements:
  ▶ Efficiency: Use least bandwidth.
    ★ Ideal Specification: Only 3 bits transferred.
    ★ Error model: Penalty 1 per additional bit.
  ▶ Robustness: Decode the integer right.
    ★ Ideal Specification: All bits are decoded properly
    ★ Error model: Penalty of 4, 2 and 1 for getting first, second and third bits wrong.
Protocol Trade-offs: Forward Error Correcting Codes

- **FECs** are protocols for **error control in noisy channels**.

- **Our problem** – Send 3 bit integers over a network
  - Say one bit-flip during transmission.
  - Additional complexity: Error in the MSB is worse than an error in LSB.

- **Two conflicting quantitative requirements:**
  - **Efficiency**: Use least bandwidth.
    - Ideal Specification: Only 3 bits transferred.
    - Error model: Penalty 1 per additional bit.
  - **Robustness**: Decode the integer right.
    - Ideal Specification: All bits are decoded properly
    - Error model: Penalty of 4, 2 and 1 for getting first, second and third bits wrong.
  - Additional boolean specifications for ensuring soundness
Synthesis of FECs: Results

- **Varying** efficiency weight $\mu_{eff}$ and robustness weight $\mu_{rob}$. 

<table>
<thead>
<tr>
<th>Domain</th>
<th>Generated protocol</th>
<th>Correctness guarantee</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hamming code</td>
<td>full error correction</td>
<td>4 bits</td>
</tr>
<tr>
<td></td>
<td>Plain</td>
<td>no error correction</td>
<td>0 bits</td>
</tr>
<tr>
<td></td>
<td>TMR for MSB</td>
<td>MSB is correct</td>
<td>2 bits</td>
</tr>
</tbody>
</table>

Completely different protocols just by varying weights.

Key properties:
- Requirements are kept separate.
- Basic functionality specified exactly.
- Advanced functionality through preferences.
Synthesis of FECs: Results

- **Varying** efficiency weight $\mu_{\text{eff}}$ and robustness weight $\mu_{\text{rob}}$.
  - Case $\mu_{\text{rob}} >>> \mu_{\text{eff}}$. Inefficient, but fully robust.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Generated protocol</th>
<th>Correctness guarantee</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hamming code</td>
<td>full error correction</td>
<td>4 bits</td>
</tr>
<tr>
<td></td>
<td>Plain</td>
<td>no error correction</td>
<td>0 bits</td>
</tr>
<tr>
<td></td>
<td>TMR for MSB</td>
<td>MSB is correct</td>
<td>2 bits</td>
</tr>
</tbody>
</table>

Completely different protocols just by varying weights.

**Key properties:**
- Requirements are kept separate.
- Basic functionality specified exactly.
- Advanced functionality through preferences.
Synthesis of FECs: Results

- **Varying** efficiency weight $\mu_{eff}$ and robustness weight $\mu_{rob}$.
  - Case $\mu_{rob} >>> \mu_{eff}$. Inefficient, but fully robust.
  - Case $\mu_{eff} >>> \mu_{rob}$. Fully efficient, but non-robust.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Generated protocol</th>
<th>Correctness guarantee</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hamming code</td>
<td>full error correction</td>
<td>4 bits</td>
</tr>
<tr>
<td></td>
<td>Plain</td>
<td>no error correction</td>
<td>0 bits</td>
</tr>
<tr>
<td></td>
<td>TMR for MSB</td>
<td>MSB is correct</td>
<td>2 bits</td>
</tr>
</tbody>
</table>

Completely different protocols just by varying weights.

**Key properties:**
- Requirements are kept separate.
- Basic functionality specified exactly.
- Advanced functionality through preferences.
Synthesis of FECs: Results

- **Varying** efficiency weight $\mu_{\text{eff}}$ and robustness weight $\mu_{\text{rob}}$.
  - Case $\mu_{\text{rob}} >>> \mu_{\text{eff}}$. Inefficient, but fully robust.
  - Case $\mu_{\text{eff}} >>> \mu_{\text{rob}}$. Fully efficient, but non-robust.
  - Case $\mu_{\text{eff}} \approx \mu_{\text{rob}}$. In-between efficiency, and gets MSB correct.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Generated protocol</th>
<th>Correctness guarantee</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_{\text{rob}} &gt;&gt; \mu_{\text{eff}}$</td>
<td>Hamming code</td>
<td>full error correction</td>
<td>4 bits</td>
</tr>
<tr>
<td>$\mu_{\text{rob}} &lt;&lt; \mu_{\text{eff}}$</td>
<td>Plain</td>
<td>no error correction</td>
<td>0 bits</td>
</tr>
<tr>
<td>$\mu_{\text{rob}} \approx \mu_{\text{eff}}$</td>
<td>TMR for MSB</td>
<td>MSB is correct</td>
<td>2 bits</td>
</tr>
</tbody>
</table>

- Completely different protocols just by varying weights.

- **Key properties:**
  - Requirements are kept separate.
  - Basic functionality specified exactly.
  - Advanced functionality through preferences.
Incompatible Specifications

- Specifications rarely monolithic.

- Designer reconciles multiple requirements.

- Writes more detailed specifications resolving corner cases and contradictions – whole field of requirements engineering and tracability
Every request $req_1$ must be immediately granted with $gr_1$.

Every request $req_2$ must be immediately granted with $gr_2$.

Grants $gr_1$ and $gr_2$ cannot occur at the same time.
Every request \( \text{req}_1 \) must be immediately granted with \( \text{gr}_1 \).

Every request \( \text{req}_2 \) must be immediately granted with \( \text{gr}_2 \).

Grants \( \text{gr}_1 \) and \( \text{gr}_2 \) cannot occur at the same time.

- Designer resolution \( \implies \) (say) by alternating between requests.
  - \( G(\text{r}_1\text{r}_2 \implies (\text{g}_1 \lor \text{g}_2)) \)
  - \( G(\text{r}_1\text{r}_2\text{g}_1 \implies (\neg \text{r}_1\text{r}_2\text{g}_1 \land \text{r}_1\text{r}_2\text{g}_2)) \)
  - \( G(\text{r}_1\text{r}_2\text{g}_2 \implies (\neg \text{r}_1\text{r}_2\text{g}_2 \land \text{r}_1\text{r}_2\text{g}_1)) \).
Every request $req_1$ must be immediately granted with $gr_1$.

Every request $req_2$ must be immediately granted with $gr_2$.

Grants $gr_1$ and $gr_2$ cannot occur at the same time.

- Designer resolution $\implies$ (say) by alternating between requests.
  - $G(r_1r_2 \implies (g_1 \lor g_2))$
  - $G(r_1r_2g_1 \implies (\neg r_1r_2g_1 \land r_1r_2g_2))$
  - $G(r_1r_2g_2 \implies (\neg r_1r_2g_2 \land r_1r_2g_1))$.

- How instead of What
Requirements now entangled.

Changing $G(req_1 \Rightarrow gr_1)$ to $G(req_1 \Rightarrow (gr_1 \lor Xgr_1))$
  
  - Lots of rewriting.
Requirements now entangled.

Changing \( G(req_1 \implies gr_1) \) to \( G(req_1 \implies (gr_1 \lor Xgr_1)) \)

▶ Lots of rewriting.

In quantitative case, add error models with equal penalties.

Can change one independently without changing anything else.
Requirements now entangled.

Changing \( G(\text{req}_1 \Rightarrow gr_1) \) to \( G(\text{req}_1 \Rightarrow (gr_1 \lor Xgr_1)) \)

- Lots of rewriting.

In quantitative case, add error models with equal penalties.

Can change one independently without changing anything else.

Changing request priorities \( \implies \) vary specification weights
Conclusion

- Quantitative specification formalism – Simulation Distances

- Can lead to more compact specifications – specifying functionality through preferences

- Synthesis from multiple quantitative specifications
  - Trade-offs in protocol synthesis
  - Resolving corner-case incompatibilities