Qualitative Analysis of SEFTs for Supporting the Certification Process of Software-Intensive Systems

Michael Roth and Peter Liggesmeyer
University of Kaiserslautern
Motivation

Embedded System

Network

HW  SW
Combines traditional fault trees with state-charts:

**State/Event Fault Trees**

- **AND_Event_State**
  - **T0:Fire**
  - **No fire detection possible**

- **OR_State**
  - **S1**
  - **S2**

- **Sensor**
  - **T0:Init**
  - **T1:Fail**
  - **Failed**
  - **λ = 1 per a**

- **Software unit**

**Watchdog**

- **T1:Trigger**
  - **δ = 1 h**
  - **Init**

**Fire alarm unit**

- **T1:Fail**
  - **Failed**
  - **λ = 1/10 per h**

**Detection**

- **T0:Init**
  - **Detection_OK**
  - **T3:Start**
  - **δ = 0.1 h**

**Software unit**

State/Event Fault Tree Properties

Combines traditional fault trees with state-charts:

- Modeling the decomposition structure of the underlying system
- Fault tree similar gates offers a **familiar syntax** for safety engineers
  - Boolean and state-based gates
- State charts covering the timing behavior of the components
- Well appropriate for software intensive systems*
- **No qualitative analysis methods**

*) M. Steiner, P. Keller, P. Liggesmeyer: Modeling the Effects of Software on Safety and Reliability in Complex Embedded Systems. Computer Safety, Reliability and Security (SafeComp Workshops), 2012
State/Event Fault Tree Translation

undetected fire

\[ \text{E} \rightarrow \text{E} \rightarrow \text{E} \rightarrow \text{E} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{Software unit} \]

\[ \text{E}_{\text{in}} = \text{Fire} \rightarrow \text{P} = \text{high} \rightarrow \text{S}_{\text{out}}(\text{OR State}) \]

\[ \text{S}_{\text{out}}(\text{OR State}) \rightarrow \text{S}_{\text{in}}(\text{AND Event State}) \]

\[ \text{S}_{\text{in}}(\text{AND Event State}) \rightarrow \text{S}_{\text{in1}} \rightarrow \text{S}_{\text{in2}} \rightarrow \text{Sensor} \rightarrow \lambda = 1/a \rightarrow \text{S. Fail} \]

\[ \text{S. Fail} \rightarrow \text{P} = \text{low} \rightarrow \text{E} \rightarrow \text{S} \rightarrow \text{E} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{S} \rightarrow \text{Software unit} \]
State/Event Fault Tree Analysis

Quantitative Analysis:

TimeNet*:
• Transient simulation
• Stationary simulation
• Transient analysis
• Stationary analysis

Qualitative Analysis:

Develop a Reachability Graph (RG) for the translated DSPN:
• Find *Minimal Event Sequences*
• RG is always finite because of boundedness of the underlying DSPN

*) A. Zimmermann, R. German, J. Freiheit, G. Hommel: TimeNET 3.0 Tool Description. Int. Conf. on Petri Nets and Performance Models (PNPM’99), 1999
Qualitative Analysis: Reachability Graph

- How is it possible to reduce the state explosion?
- How can the resulting graph be simplified?

➢ 4 step reduction process
1st: Transition Priorization

- Timed transitions: $P=\text{low}$
- Immediate transitions: $P=\text{medium}$
- Gate transitions: $P=\text{high}$

$\text{RG}_{\text{Prio}}(\text{DSPN})$
2nd: TLE Termination

Stop the path execution by reaching a TLE-place!
**3rd: Mute Transitions**

Substitution through successor state

- **S0**: F.Fail → S1
- **S1**: (T1) S.Fail
- **S2**: (T4) W.Trigger
- **S3**: S.Fail → W.Trigger
- **S18**: F.Fail (T8)
- **S19**: Fire

- **S0**: F.Fail → S3
- **S3**: S.Fail → Fire
- **S19**: Fire

- **RG_{Mute(RG_{P&T})}**
Remove all paths which never end in a TLE-state (non-trivial cycles)
Remove all trivial cycles in the RG
Merge similar TLE-states
Results & Evaluation

<table>
<thead>
<tr>
<th></th>
<th>SEFT</th>
<th>DSPN</th>
<th>RG\textsubscript{Full}</th>
<th>RG\textsubscript{Prio}</th>
<th>RG\textsubscript{Term}</th>
<th>RG\textsubscript{Mute}</th>
<th>RG\textsubscript{Min}</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>6</td>
<td>13</td>
<td>192</td>
<td>63</td>
<td>49</td>
<td>17</td>
<td>7</td>
</tr>
<tr>
<td>Transitions</td>
<td>9</td>
<td>14</td>
<td>880</td>
<td>90</td>
<td>64</td>
<td>32</td>
<td>13</td>
</tr>
<tr>
<td>Gates</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Reduction</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>85%</td>
<td>90%</td>
<td>95.5%</td>
<td>98%</td>
</tr>
</tbody>
</table>
Summary

• We show a 4 step process to perform a qualitative SEFT analysis
• We applied it on a example system
• We could show a significant reduction of the underlying state space
  ➢ This holds the RG understandable and suitable to support the verification process of the system

Perspectives

• Apply different measures on MES-Sets equivalent to FTA
  • e. g. importance analysis
• Improve the analysis: (performance, shortest path, …)