From Safety Analysis to Software Requirements

Kirsten M. Hansen
Anders P. Ravn Member
Introduction

• Leveson, “Software Safety in Embedded Computer Systems”.  
  – Software requirements must satisfy system safety constraints  
  – Software must be formalized in order to raise confidence in the verification

• Demonstrate how fault trees resulting from safety analysis can be interpreted directly as requirements

• Link fault tree analysis to program development

• Give a common semantic model
Introduction

- In particular, failures due to the lack of understanding of the interactions among system,
  - Missing common framework

- To overcome this problem, using
  - Dynamic systems framework
  - Duration calculus formulas by timing diagrams

- Informal descriptions of fault tree gates are ambiguous
  \[\Rightarrow\text{Formalization}\]
Fault Trees

• Deductive safety analysis technique

• Applied during the design phase

• A fault tree is given a particular failure
  – Possible combination of component failures that may lead to this failure
### Fault Trees

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="X" /></td>
<td>A node denoting a failure X, possibly resulting from a combination of basic failures (nodes).</td>
</tr>
<tr>
<td><img src="image" alt="AND-gate" /></td>
<td>AND-gate—the failure in the top node occurs only when all the failures in the children nodes occur.</td>
</tr>
<tr>
<td><img src="image" alt="OR-gate" /></td>
<td>OR-gate—the failure in the top node occurs only when one or more of the failures in the children nodes occur.</td>
</tr>
<tr>
<td><img src="image" alt="INHIBIT-gate" /></td>
<td>INHIBIT-gate—the failure in the top node occurs only when both the failures in the child node occurs and the condition in the oval is true.</td>
</tr>
<tr>
<td><img src="image" alt="EXCLUSIVE OR-gate" /></td>
<td>EXCLUSIVE OR-gate—the failure in the top node occurs only when exactly one of the failures in the children nodes occurs.</td>
</tr>
<tr>
<td><img src="image" alt="PRIORITY AND-gate" /></td>
<td>PRIORITY AND-gate—the failure in the top node occurs only when the failures in the children nodes occur in a left to right order.</td>
</tr>
</tbody>
</table>
Duration Calculus

- Events are rely on some minimal or maximal time of occurrence

- A common thread for in all these possible interpretations
  - Event are observed while time passes
  - i.e.) over finite intervals of time, when certain state patterns occur, suggesting the use of a real-time, interval logic
Fire occurs

- Gas explodes
  - Air present
  - Excess of gas present
- Electrical short in cables
  - Ignition attempted
    - Gas leaks for more than 4 sec
    - Observation interval less than 30 sec
Interval of diagram

• Gas leaks for more then 4 sec
  – Gas, Flame : Time → {0,1}
  – Boolean values are represented by 0(false) and 1(true)
• The gas valve is open and there is no frame
  – \( \text{Leak} \equiv \text{Gas} \land \neg \text{Flame} \)
• \( \int_{b}^{e} \text{Leak}(t) \)
Duration calculus

- Gas of leaks for more than 4 sec
  \[ \int \text{Leak} > 4 \]
- 1, \( \int 1 \) is constant state
  \[ \int 1, l \]
- An interval is not longer than 30 sec
  \[ l \leq 30 \]
- Time interval is not longer than 30 seconds and gas leaks for more than 4 sec
  \[ (l \leq 30) \land (\int \text{Leak} > 4) \]
- Ignition, holds throughout a nonpoint interval
  \[ \int \text{Ignition} = l \land l > 0, [\text{Ignition}] \]
Duration calculus

• “Chop” operator (written in “;”)
  – Subinterval properties
  – $D_1; D_2$ in $[b, e]$
    • $D_1$ hold for the interval $[b, m]$
    • $D_2$ hold for the interval $[m, e]$

• Somewhere $D$
  – $\Diamond D \equiv true; D; true$

• Everywhere $D$
  – $\Box D \equiv \neg \Diamond (\neg D)$
Fault Tree Semantics - Leaves

• Events
  – Leaves in a fault tree
  – Occurrence of a specific system state
  – Example
    • True, false
    • Occurrence of State, i.e., P, \([P]\)
    • Occurrence of an event, i.e., a transition to state P:\(\neg P\); [P]
    • Elapse of a certain time i.e., \(l \geq (30 + \epsilon)\)
    • A threshold of some duration i.e., \(\int P \leq 4 \times \epsilon\)
Intermediate Nodes

- **Edges**
  - A connected to a node, B, by an edge
  - Example)
    - $A \overset{\text{def}}{=} B$
    - System failure A occurs when the failure B occurs
    - Assumes that if something has the possibility of going wrong, then it does go wrong
Intermediate Nodes

- **AND Gate**
  - \( A \equiv B_1 \land \cdots \land B_n \)
  - \( A \) holds iff \( B_1 \) to \( B_n \) simultaneously
  - \( A \equiv \Box B_1 \land \cdots \land \Box B_n \) is rejected
    - Remembers any occurrence of \( B_i \)

- **OR Gate**
  - \( A \equiv B_1 \lor \cdots \lor B_n \)
Intermediate Nodes

• INHIBIT Gate
  – If A occurs then \( B_1 \) has occurred in the past while condition \( B_2 \) was true
  – \( B_1, B_2 \) as inputs, where \( B_1 \) and \( B_2 \) are duration formulas
  – \( A \stackrel{\text{def}}{=} B_1 \land \cdots \land B_n \)

• EXCLUSIVE OR
  – \( A \stackrel{\text{def}}{=} (B_1 \land \neg B_2) \lor (\neg B_1 \land B_2) \)
  – Generalized to
    • \( A \stackrel{\text{def}}{=} (B_1 \land \neg(B_2 \lor \cdots \lor B_n)) \lor \cdots \lor (B_1 \land \neg(B_2 \lor \cdots \lor B_n)) \)
Intermediate Nodes

- **PRIORITY AND**
  - \( A \stackrel{\text{def}}{=} B_1 \land \bigcirc (B_2 \land \bigcirc (B_3 \land \ldots \land B_n) \ldots ) \)
  - Generalized to
    - \( A \stackrel{\text{def}}{=} (B_1 \land \neg (B_2 \lor \ldots \lor B_n)) \lor \ldots \lor (B_1 \land \neg (B_2 \lor \ldots \lor B_n)) \)
Software Safety Requirements

• Design and the safety analysis should proceed concurrently
  – Fault tree analysis influence the design
• Fault tree analysis and the system design must use the same system model
• Deriving safety requirement used during system development in order to validate the design
• E.g) Root is $S$, the system should be build such that $S$ never occurs
  – $\square \neg S$
  – $\square \neg S_1 \land \cdots \land \square \neg S_n$, roots are interpreted as $S_1 \land \cdots \land S_n$
Deriving Component requirements

• **AND-Gates**
  
  \[ A = B_1 \land \cdots \land B_n \]
  \[ \implies \Box \neg A = \Box \neg (B_1 \land \cdots \land B_n) \]
  
  • If \( B_1 \) is not controllable, the worst case is \( B_1 = true \)
  
  Software engineer implements, \( \Box \neg (B_2 \land \cdots \land B_n) \)

• **OR-Gates**

  \[ A = B_1 \lor \cdots \lor B_n \]
  \[ \implies \Box \neg A = \Box \neg (B_1 \lor \cdots \lor B_n) = \Box \neg B_1 \land \cdots \land \Box \neg B_n \]

  • \( B_1 = true \rightarrow \Box \neg B_1 \) is \( false \)
  
  Weaken the requirement specification

  • Assumption(\( Asm \)) and commitment(\( Com \))
  
  • \( Asm \rightarrow Com \) is weakened to \( Asm \land \Box \neg B_1 \rightarrow Com \)
Deriving Component requirements

• PRIORITY AND-Gates

- \( A = B_1 \land \Diamond (B_2 \land (\Diamond B_3 \land \cdots \land \Diamond B_n) \ldots ) \)

- \[ \implies \Box \neg A = \Box \neg (B_1 \land \Diamond (B_2 \land (\Diamond B_3 \land \cdots \land \Diamond B_n) \ldots )) \]

- \[ = \Box \neg B_1 \lor \Box \neg B_2 \lor \cdots \lor \Box \neg B_n \]

• If \( B_1 \) is not controllable, the worst case is \( B_1 = true \)

Software engineer implements, \( \Box \neg (B_2 \land (\Diamond B_3 \land \cdots \land \Diamond B_n) \ldots ) \)
Example

- Formalization of fault tree, derivation of safety requirements
- Railway interlocking system
  - Prevent trains from colliding and derailing while allowing train movements
  - tpo-station topology
    - Track segment, points, signal
  - sst-Station state
    - Track segment
  - intlck-state of interlocking system
    - Train routes currently set
Example

- **Safety Commitment**
  - \( \text{Safe}_{\text{com}} \overset{\text{def}}{=} \Box \neg \text{Collision}(\text{tpo}, \text{sst}, \text{intlck}) \)

- **Fault Tree Analysis**
  - \( \text{Collision}(\text{tpo}, \text{sst}, \text{intlck}) \overset{\text{def}}{=} \)
    - \( \text{Overlap}_{\text{areas}}(\text{top}, \text{sst}, \text{intlck}) \)
    - \( \lor \text{Signal}_{\text{bypass}}(\text{tpo}, \text{sst}) \)

- **Safety Requirements**
  - Chosen not to implement \( \Box \neg \text{Signal}_{\text{bypass}}(\text{tpo}, \text{sst}) \)
  - Deduced
    - \( \text{Safe} \overset{\text{def}}{=} \Box \neg \text{Signal}_{\text{bypass}}(\text{top}, \text{sst}) \)
    - \( \Rightarrow \Box \neg \text{Overlap}_{\text{bypass}}(\text{tpi}, \text{sst}, \text{intlck}) \)
Fault Tree analysis

- \( \text{Collision}(t\text{po, sst, intlck}) \overset{\text{def}}{=} \) \\
  \( \text{Signal}_{\text{bypass}}(t\text{po, sst}) \) \\
  \lor \text{Outsie}\_\text{route}(sst, intlck) \\
  \lor (\text{On}\_\text{routes}(sst, intlck) \\
  \land (\text{Overlap}\_\text{routes}(sst, intlck) \\
  \lor \text{Reverses}(t\text{po, sst, intlck}) \\
  \lor \text{Error}\_\text{point}(t\text{po, sst, intlck}) \\
  \lor \text{Error}\_\text{signaling}(t\text{po, sst, intlck}))) \)
Safety Requirements

- $\text{Safe\_com} \overset{\text{def}}{=} \square \neg (\text{On\_route}(\text{sst}, \text{intlck}) \land \text{Overlap\_route}(\text{intlck}) \lor (\text{Error\_point}(\text{tpo}, \text{sst}, \text{intlck}) \lor \text{Error\_signaling}(\text{tpo}, \text{sst}, \text{intlck})))$

- $\text{Asm} \overset{\text{def}}{=} \square \neg \text{Signal\_bypass}(\text{tpo}, \text{sst}) \land \square \neg \text{Outside\_route}(\text{sst}, \text{intlck}) \land \square \neg (\text{On\_routes}(\text{tpo}, \text{sst}, \text{intlck}) \land \text{Reverses}(\text{tpo}, \text{sst}, \text{intlck}))$

- Deduced safety requirements are
  - $\text{Safe} \overset{\text{def}}{=} \text{Asm} \Rightarrow \text{Safe\_com}$
Conclusion

• Software may be proven to satisfy the system safety requirements
• Logic is capable of expressing both the semantics of the intermediate events
• Fault tree analysis and system design can be made to interact much more effectively