Adaptive System for Autonomous Driving

Franz Wotawa and Martin Zimmermann

Christian Doppler Laboratory for
Quality Assurance Methodologies for Cyber-Physical Systems (QAMCAS)
Institute for Software Technology
Graz University of Technology, Inffeldgasse 16b/2, 8010 Graz, Austria

{wotawa, mzimmerm}@ist.tugraz.at
Agenda

• Introduction
• Rule Based Language
• Case Study
• Conclusion and Further Work
Introduction

- 32,000 deaths per year in US traffic accidents
  - 90% caused by humans
- Major challenge for autonomous driving: No human backup
- Solution: Redundancy and fail safe operation
- Introduction of a domain specific language
Rule Based Language

• Rule based
• Small scope
• Easily combinable with Java
• Weight model
  • Number of executions and failures per rule
• Path finding
  • Best rule predicted from the past
Syntax

• Memory
  • Contains all the current facts about the model in the form of propositions

• Rule
  • [Pre conditions]
  • [Additive post condition]/[Goal]
  • [Subtractive post conditions]
  • Action

• Pre and post conditions are propositions
  • Removed from or added to the memory based on the rule

(proposition.)*
pre* -> [(+post/#goal)] (-post)* action.
Syntax Example

1. running.
2. running -> +objectDetected detectObjectWithCamera.
3. running -> +objectDetected detectObjectWithLiDAR.
4. running, objectDetected -> #objectProcessed -objectDetected processObject.

<table>
<thead>
<tr>
<th>After Line</th>
<th>Memory</th>
<th>Possible next lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>running</td>
<td>2, 3</td>
</tr>
<tr>
<td>2</td>
<td>running, objectDetected</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>4</td>
<td>running</td>
<td>2, 3</td>
</tr>
</tbody>
</table>
Weight Model

- 2 parts
  - Activity (how often selected)
  - Damping (how often failed/succeeded)

- Activity
  \[
  \text{activity}_n = \frac{\text{chosen}_{n-1} + \text{activity}_{n-1}}{2}
  \]
  Chosen more often => higher activity

- Damping
  - ✔️: \(damp_n = \begin{cases} 0.1, & damp_{n-1} - 0.1 < 0.1 \\ damp_{n-1}, & \text{otherwise} \end{cases}\)
  - ✗: \(damp_n = \begin{cases} 0.9, & damp_{n-1} + 0.1 > 0.9 \\ damp_{n-1} + 0.1, & \text{otherwise} \end{cases}\)
  Fails more often => higher damping

- Weight
  \[
  \text{weight(activity, damp)} = (1 - \text{activity}) \times (1 - \text{damp})
  \]
  Activity and damp is high => lower weight
Path Finding

1. Construct all paths (a rule can only be once in a path)
2. Remove all reducible paths
3. Choose plan with highest weight

1. running.
2. running → +objectDetected detectObjectWithLiDAR.
3. running → +objectDetected detectObjectWithRADAR.
4. objectDetected → #objectProcessed processObject.
Case Study

• Simulate camera, radar and LiDAR to detect objects
  • Probability based on sensor and current condition

• Every time detectObjectWithX is called
  • Generate random number between 0 and 100
  • If number < probability success else failure

lidar_ok.
radar_ok.
camera_ok.
camera_ok -> +objectDetected actions.detectObjectWithCamera.
lidar_ok -> +objectDetected actions.detectObjectWithLiDAR.
radar_ok -> +objectDetected actions.detectObjectWithRADAR.
objectDetected -> #objectProcessed -objectDetected actions.processObject.
Scenarios

• **Goal:** Most successful object detections

• **Weather simulation**
  - Clear
  - Rain
  - Storm

• **Total fault simulation**
  - L/R
  - C/R
  - C/L
Results Weather Simulation

Improvement from 73.3% to 75.5% (maximum success 88%)
Results Total Fault Simulation

Improvement from 66.6% to 96% (maximum success 100%)
Conclusion and Further Work

• We developed a domain specific language with which we can model fault tolerant systems easily

• Interesting results from case study
  • Bigger deviation => better results

• Further work
  • Improvement of the language (e.g.: aging of the damping)
  • Create more scenarios
Thank you