On the Mitigation of MultiCore-Induced Behavioral Deviations of an Autonomous Ground Vehicle

Jonathan Sprinkle, ECE
Joint work with Prof. Brandon Eames, Utah State University

This work is supported by the Air Force Research Laboratory, under award #FA8750-08-1-0024, titled “MultiCore Hardware Experiments in Software Producibility”.

Arizona’s First University.
• In the end:
  – We can accomplish more with tools, than with elbow grease
• In the beginning:
  – Our tools come from waste, or leftovers
• In the end:
  – We can use tools to create new tools that are better than the first tools!!

---

**In the beginning...**

---

**THE UNIVERSITY OF ARIZONA.**

Electrical and Computer Engineering
Domain-Specific Modeling: An abstract perspective

Domain Concepts

Unrestricted Implementation
Domain-Specific Modeling: An Abstract Perspective

Domain Concepts

Defns of Domain Assumptions and Givens
Domain-Specific Code Generation

DS Code Generator

Domain “Instance”
Domain-Specific Modeling

- Create *model* of the system
- Perform
  - Analysis
  - Architecture exploration
  - Simulation
- Generate
  - Configuration
  - Code
  - Executables

- *From the same models!*

Example Domains & Environments:
- VLSI Layout (e.g., Altera)
- Engg Drawing (e.g., AutoCAD)
- Physical Modeling (e.g., SolidWorks)
- Signal Processing (e.g., LabVIEW)
- Controls (e.g., Simulink)
While Event $e_i$, and in State, $s_c$
After, $e_i$.delay, and in State, $s_c$,
Stop clock
If exists Transition $t_e$: (src=$s_c$, dst=$s_n$), set $s_c = s_n$
Else if $s_c$.parent=null, set $e_i = e_i$.amSrc.sequence.dst
Else transition through $s_c$.parent
Advance clock
QuickTime™ and a decompressor are needed to see this picture.
Formal Definition of a Domain-Specific Language

\[ L = \langle C, A, S, M_c, M_s \rangle \]
• SEC Capstone Demonstration
• Pursuit/Evasion of fixed-wing aircraft
  – Joint work with Dr. Mike Eklund, Dr. Jin Kim, Prof. Shankar Sastry
Pilot: “Plane reacted just like pilots are trained to do”

Me: “I couldn’t tell whether it was a computer or person controlling the plane”
• SEC Capstone Demonstration
• Landing/Wave-off scenario (safety calculation)
  • Joint work with Dr. Mike Eklund, Dr. Ian Mitchell, Prof. Shankar Sastry


Motivating Example

A UAV is waved off, and then after some time redirected to land.

Can the decision to safely land:
- be made in real time?
- be guaranteed as true?

See Sprinkle et al., ISSE2006
Generative Strategy: Decision Controller Synthesis

See Sprinkle et al., ISSE2006

\[
\theta_\beta(x) = -\left( x_3 \sin(\beta) \sqrt{x_1^2 + x_2^2 + x_3^2} \right) + \theta_\beta \quad (29)
\]

\[
\psi_\beta(x) = -(x_2 - \tan(\psi) x_1) + \psi_\beta \quad (30)
\]

From this, and utilizing (12), we obtain the desired feedback control law:

\[
P_{11}(x) = -u_{\max}(\theta - \text{Sat}_{h_{\max}}(\theta\beta(x))) \quad (31)
\]

\[
Q_{11}(x) = -u_{\max}(\psi - \text{Sat}_{\psi_{\max}}(\psi\beta(x))) \quad (32)
\]

where \(\theta_\beta\) and \(\psi_\beta\) are given as in (29) and (30).

\[
G_0 = \begin{cases} 
\theta_\beta \in [2.85^\circ, 3.15^\circ], \\
\psi_\beta \in [-0.2^\circ, +0.2^\circ], \\
x_2 \in [-100, +100] \text{ ft}, \\
x_3 \in [-15, +15] \text{ ft}, \\
x_1 = 0
\end{cases}
\]

\[
H(\bar{x}, p) = \min_{u \in U} \frac{p^T \int(\bar{x}, u)}{f_{x_1}(\bar{x})}
\]
Including: Autonomous Vehicles

Joint work with Ben Upcroft, Hugh Durrant-Whyte (USyd), Will Uther, Robert Fitch (NICTA) Humberto Gonzalez, Esten Grøtli, Shankar Sastry (Berkeley) and MANY OTHERS!!!!

http://dgc3.eecs.berkeley.edu/
- Model 1: Rigid body dynamics
- Model 2: Kinematic bicycle model
- Model 3: Ackermann four-wheeled model

* Sample points not necessarily representative of models; points are exaggerated to show deviations in behavior.
The goal of our experiment is to determine the robustness of a somewhat fragile cooperation between two data-driven components, capable of running on the same core, two different cores, or two different machines, when those components are deployed on various machines with various single or multicore processors.
What ways are there to see fragility?

This discrete behavioral difference is akin to a condition number, that when accumulated over many different instances (from the same initial conditions) will indicate whether new hardware amplifies this fragility.
### Simulation hardware

<table>
<thead>
<tr>
<th>Machine Name</th>
<th>Processor Type</th>
<th>Processor Details</th>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>macphee</td>
<td>Single Core</td>
<td>Allocated as one processor through VMWare. Hardware is Intel Core 2 Duo, 2.4 MHz</td>
<td>Linux Kubuntu 6.10 (Kernel 2.6.17-10-386) (VMWare running on Mac OS X Host)</td>
</tr>
<tr>
<td>dimble</td>
<td>Dual Core</td>
<td>Intel Core2 Duo CPU, E4500, 2.20GHz</td>
<td>Linux Kubuntu 6.10 (Kernel 2.6.17-10-386)</td>
</tr>
<tr>
<td>beames-desktop</td>
<td>Quad Core</td>
<td>Intel Q6600</td>
<td>Linux Kubuntu 6.10 (Kernel 2.6.17-10-386)</td>
</tr>
<tr>
<td>feverstone</td>
<td>Dual Processor, Quad Core</td>
<td>Intel Xeon CPU, X5460, 3.16GHz, 6MB Cache</td>
<td>Linux Kubuntu 6.10 (Kernel 2.6.17-10-386)</td>
</tr>
<tr>
<td>hardecastle</td>
<td>Dual Processor, Quad Core</td>
<td>Intel Xeon CPU L5420, 2.50GHz, 6MB Cache</td>
<td>Linux Kubuntu 6.10 (Kernel 2.6.17-10-386)</td>
</tr>
</tbody>
</table>
Basic simulation

Model of Computation:
Process Networks (PN)
• Produces a series of (intermediate) waypoints, to guide the vehicle to some location, and to be in a particular heading at that location

• Waypoints are not tracked, so each time highlevelplanner runs, it produces new waypoints for the vehicle

• Waypoints are calculated backward from the ‘next’ destination
Reminder: what does it mean to fail?

Failure

If in 200 seconds we have not passed “dead man’s curve” then we are not going to make it past.

Success
Probability of Success (basic sim)

Number of runs ranges from $n=50$ (1) to $n=400$ (4,8,8)
Visualization across planned paths, and hardware

Planned \((X,Y)\) position, Waypoint 0, Basicsim All machines

- 'dimble_basicsim_wp0_Data'
- 'beames-desktop_basicsim_wp0_Data'
- 'feverstone_basicsim_wp0_Data'
- 'macphee_basicsim_wp0_Data'
- 'hardcastle_basicsim_wp0_Data'

Desired Position

- \(-40\)
- \(-20\)
- \(0\)
- \(20\)
- \(40\)
- \(60\)
- \(80\)
- \(100\)
- \(120\)
- \(140\)
- \(160\)
One machine: lots of waypoints
What’s the deal?

- Highlevelplanner runs TOO FAST!
- We are providing waypoints so fast that the local navigator goes unstable

$$d_{w_1} \gg d_x$$
• We have some system where we have a “race condition” (no, not really, but...) that is exacerbated by an increase in the number of cores

• We see some functional (behavioral) difference, that is (somewhat) easily characterized

• Now:
  – What strategies can mitigate this issue?
  – Can we do this without changing existing code?
  – What experiments should we run to show that it is working?
Strategy: time triggering
• Do not depend on dataflow for firing rules
• Go to sleep, and wake up when they are fired (usually at some period, $T$)

• Can still lead to issues, if the start time is not consistent
• Fix the scheduled time at which inputs are read, and outputs produced

• As long as the component begins and ends between the logical beginning and end of the LET, the behavior is deterministic

• These kinds of schedules are robust to new processors (which execute faster)

• Even lazy versions of this work fairly well
Probability of Success (letsim sim)
Comparison of simulation results
All waypoints, LET simulation
First waypoint, all machines
Planned (X,Y) position, Waypoint 0, Basicsim All machines

- 'dimble_basicsim_wp0_Data'
- 'beanes-desktop_basicsim_wp0_Data'
- 'feverstone_basicsim_wp0_Data'
- 'macphee_basicsim_wp0_Data'
- 'hardcastle_basicsim_wp0_Data'

Reminder: Basic sim, first waypoint, all machines
Brief thoughts:

• Now:
  – What strategies can mitigate this issue?
  – Can we do this without changing existing code?
  – What experiments should we run to show that it is working?

• Answers
  – Strategies: lazy time triggering provides improvement
    • Granted: are improving a very fragile system
  – Changing code: No legacy code changed
  – Experiments:
    • we run from the same initial conditions across lots of hardware platforms
    • automated the experiment runs, and streamlined execution
Where do we go from here?

- A rich research agenda of continued investigation:
  - Hardware substitution:
    - What is the proper way to measure equivalence across multicore platforms when comparing *behaviors* (and not just end results). Unknown.
  - Control: This fishing pole control mechanism has its own dynamics
    - Can we further abstract this problem?
    - Does the dynamics of moving the pole affect stability of the whole system
  - Modeling
    - What pieces of the system can be generated automatically, with a specific semantics for component interaction?
    - Can we generate the new triggers, and get even better equivalence?
  - Software
    - Can we apply to systems that do not use middleware?
  - Analysis
    - What sophisticated comparison metrics can indicate potential pitfalls when mapping? Can we extend the graphs generated?


We are always looking for good graduate students.

http://ece.arizona.edu/