Experiments with Diversified Models for Fault-Tolerant Planning

Benjamin Lussier, Matthieu Gallien, Jérémie Guiochet, Félix Ingrand, Marc-Olivier Killijian, David Powell

Dependable Computing Group
Robotics Group

IARP DRHE, Rome, April 14-15, 2007
Motivation

Introduction

- Dependability of autonomous robots in critical applications
  - Space exploration
  - Medical assistance
  - Service

- Planning is central to complex autonomous systems: selection & organization of actions to achieve high-level goals
Motivation

Planning basics

Declarative domain model
• Domain objects
• Actions on domain objects
• Constraints on objects
• Constraints on actions

Goals

Search engine

Current state

Heuristics

NP-complete

Accuracy/tractability compromise

Model & heuristics incrementally constructed

Small changes have dramatic consequences

Validation challenges:
  – execution context variability
  – need for integrated testing
  – oracle problem

Plan execution control

Functional layer
(control of physical devices)

Environment
Fault-Tolerant Planning

Redundancy

A

Domain model

Heuristics

Search engine

Goals

B

Domain model

Heuristics

Search engine

Current state
Fault-Tolerant Planning

Redundancy with diversity

A

DOMAIN MODEL

HEURISTICS

Goals

Search engine

B

Domain model

Heuristics

Current state
Fault-Tolerant Planning

Error detection mechanisms

- **Watchdog timer**: time limit on search process
  - search too slow or deadlocked
    - fault in model, search engine or search heuristics
- **Plan analyzer**: acceptance test on output of planner
  - plan does not respect constraints/properties in *on-line oracle*
    - fault in model, search engine or search heuristics
- **Plan failure detector**: monitors actions during plan execution
  - classic execution control: an action can fail due to:
    - adverse environmental situation, or
    - inadequate plan (due to fault in model, etc.)
  - plan repair can be attempted before declaring failed plan
- **On-line goal checker**: monitors goals during plan execution
  - system state analyzed following each action execution
  - some goal not achieved when plan execution terminates
Fault-Tolerant Planning

Recovery paradigms

- **Sequential execution** of redundant planners
  - choose a planner to provide initial plan
  - each time an error is detected
    - ask a different planner to re-plan
    - until all achievable goals reached (or all planners fail in a row)

- **Concurrent execution** of redundant planners
  - ask all planners to provide initial plan; choose a correct one
  - each time an error is detected
    - ask all planners to re-plan; choose a correct one
    - until all achievable goals reached (or all planners fail in a row)
Fault-Tolerant Planning

Sequential planner execution

1. begin mission
2. failed_planners ← Ø;
3. \textbf{while} (goals ≠ Ø) \textbf{do}
4. candidates ← planners;
5. \textbf{while} (candidates ≠ Ø \& goals ≠ Ø) \textbf{do}
6. choose k such as (k ∈ candidates) \& (k \notin failed_planners);
7. candidates ← candidates \setminus k;
8. init\_watchdog(max\_duration);
9. send (plan\_request) to k;
10. \textbf{wait} % for either of these two events
11. receive (plan) from k
12. stop\_watchdog;
13. \textbf{if} analyze(plan) = OK \textbf{then}
14. failed_planners ← Ø;
15. k.execute\_plan();
16. \% if the plan fails goals != empty
17. \% and then we loop to line 5 or 3
18. else
19. send(k.invalid\_plan) to operator;
20. failed_planners ← failed\_planners \cup k;
21. end if
22. \textbf{wait}
23. \textbf{if} failed\_planners = planners \textbf{then}
24. raise exception "no valid plan found in time";
25. \% no remaining planner, \% the mission has failed
26. end if
27. end while
28. end mission

© LAAS-CNRS 2007
Fault-Tolerant Planning

Concurrent planner execution

1. begin mission
   while (goals ≠ Ø)
     candidates ← planners;
     init_watchdog(max_duration);
     send (plan_request) to candidates;
     while (candidates ≠ Ø)
       wait % for either of these two events
       □ receive (plan) from k ∈ candidates
       candidates ← candidates \ k;
       pause_watchdog;
       if analyze(plan)=OK then
         stop_watchdog;
         send (cancel_planning) to candidates;
         candidates ← Ø;
         k.execute_plan();
         % if the plan fails goals /= empty
         % and then we loop to line 3
       else
         resume_watchdog;
         send(k.invalid_plan) to operator;
         if (candidates = Ø)
           raise exception "no valid plan
           % no remaining planner,
           % the mission has failed
           found in time";
       end if
   end if
   watchdog_timeout
   □ raise exception "no valid plan
   % no remaining planner,
   % the mission has failed
   found in time";
23. end wait
24. end while
25. end mission
Fault-Tolerant Planning

LAAS Architecture for Autonomous Systems

Temporal planner & executive (IxTeT)

Procedural executive (OpenPRS)

Functional layer (GenoM)

Environment

- Temporal constraint planner that instantiates and arranges high level actions into a plan that is conflict-free and fulfills mission goals + action execution control
- Decomposition and refinement of plan actions into elementary tasks + elementary task execution control
- Set of automatically generated GenoM modules performing computation (e.g., trajectory computation) or communication with physical devices
Fault-Tolerant Planning

Implementation

FTplan

Temporal planner & executive (IxTeT2)

Temporal planner & executive (IxTeT1)

Procedural executive (OpenPRS)

Functional layer (GenoM)

Environment

Detection mechanisms
- watchdog timer ✓
- plan analyzer ✗
- plan failure detection ✓
- on-line goal checker ✓

Recovery paradigms
- sequential ✓
- concurrent ✗
Goals

- Procedural executive (OpenPRS)
- Functional layer (GenoM)
- Pocosim Library
- Gazebo robot simulator
- Temporal planner & executive (IxTeT2)
- Temporal planner & executive (IxTeT1)
- FTplan
- watchdog

Validation Experiments
Fault-injection and simulation environment

- Domain model
- Faults
- Environment (world)
- Goals (mission)

Efficacy
- %Goals achieved

Performance
- Mission duration
- Distance covered

Internal
- #Replans, …
Validation Experiments

Missions

M1

M2

M3

M4

photos
comms
return
Validation Experiments

Worlds

W1

W2

W3

W4
Validation Experiments

Workload (missions x worlds)

M1 W1  M2 W1  M1 W2  M2 W2
M3 W1  M4 W1  M3 W2  M4 W2
M1 W3  M2 W3  M1 W4  M2 W4
M3 W3  M4 W3  M3 W4  M4 W4
Results

Fault-free performance

IARP DRHE, Rome, April 14-15, 2007

© LAAS-CNRS 2007
Results

Fault-free performance (with forced replanning)
Validation Experiments

Faultload

- Syntactic mutations using SESAME tool
  - Substitutions of numerical values
    - «-oo» <----> «+oo», «0», «60», «1», «-1», «-4», «26.3»
  - Substitutions of variables
    - «?obj» <----> «?x», «?y»
    - «t_start» <----> «t_end»
  - Substitutions of attribute values
    - «downward» <----> «straight», «other»
    - «none» <----> «done», «doing»
  - Substitutions of operators
    - «nonPreemptive» <----> «latePreemptive», «»
    - «-» <----> «+», «*», «/»

- 600 line plan model gives 3000 mutants
  - Random selection of mutants to be executed (amongst 1300 executable mutants)
## Results

### Fault-tolerance efficacy: mutation 1-39

<table>
<thead>
<tr>
<th></th>
<th>%_Photos</th>
<th>%_Comms</th>
<th>%_Return</th>
<th>%Bad Missions</th>
<th>#Replan</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 W1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M2 W1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M3 W1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M4 W1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IARP DRHE, Rome, April 14-15, 2007
Results

Fault-tolerance efficacy: mutation 1-589

<table>
<thead>
<tr>
<th>%¬Photos</th>
<th>%¬Comms</th>
<th>%¬Return</th>
<th>%Bad Missions</th>
<th>#Replan</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 W1</td>
<td>W2</td>
<td>W3</td>
<td>W4</td>
<td>M2 W1</td>
</tr>
</tbody>
</table>

IARP DRHE, Rome, April 14-15, 2007
Results
Fault-tolerance efficacy: mutation 1-583

| %
| %
| %
| %
| %
| %
| %
| %
| %
| %

IARP DRHE, Rome, April 14-15, 2007

© LAAS-CNRS 2007
Results

Fault-tolerance efficacy: overall results (28 mutations)

<table>
<thead>
<tr>
<th>Improvement</th>
<th>Photos</th>
<th>Comms</th>
<th>Return</th>
<th>Missions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Including W4</td>
<td>50%</td>
<td>64%</td>
<td>58%</td>
<td>29%</td>
</tr>
<tr>
<td>Excluding W4</td>
<td>62%</td>
<td>70%</td>
<td>80%</td>
<td>41%</td>
</tr>
</tbody>
</table>
Conclusion

Summary and future directions

- FT-plan concept
  - watchdog - plan analyzer - failure detection - goal-checking
  - sequential or concurrent planning

- Implementation of sequential FT-Plan with dual IxTeTs
  - diverse models: Cartesian / cellular
  - mutation-based fault-injection on declarative models
  - useful 30-40% reliability improvement in presence of faults

- Ongoing and future work
  - diverse parameters of depth-first search cost functions
  - definition of a plan analyzer (on-line test oracle)
  - multiple parallel simulations (~20 minutes / experiment)
  - definition of an off-line oracle for automated planner testing
  - on-line checking of safety properties (PHRIENDS)