Intelligent, Fault Tolerant Control for Autonomous Systems

Willibald Krenn and Franz Wotawa

Institute for Software Technology, Graz University of Technology
Outline


2. Hardware: “How does the example platform look like?”

3. Methodology: “How do we attempt to solve the problem?”

4. Results: First experiments

5. Q/A
Motivation

• What?
  – Build an autonomous system that reacts flexible and in an intelligent manner and that uses explicit knowledge

• Why?
  – Improved situational awareness
    • Environmental conditions, Internal conditions (e.g. faults, energy)
  – Fault tolerance / robustness
  – ...

We want to build a system that needs less maintenance costs, is more reliable and more flexible than systems designed with “ordinary” approaches.
Example System

- Solar
- GSM/GPS
- Axle driven generator
- Battery
- µC PIC 18F (4K RAM, 128K Flash)
- Sensors (Acceleration, ..)
- Near-Field
Outline


2. Hardware: “How does the example platform look like?”

3. Methodology: “How do we attempt to solve the problem?”

4. Results: First experiments

5. Q/A
Basic Idea

1. Rule Set
   - Describes all possible action sequences (behaviors)
   - Describes all goals the system has to reach (by employing actions)
   - Standard logic (true/false; not multi-valued)

2. Set of Activity Profiles
   - Describe the intended activity of, e.g., a goal, over time
   - Used for calculating a weight that expresses the desirability to reach the associated rule consequent

3. Algorithm
   - Search for all ways (behaviors) to reach goals by evaluating the rule set. (Evaluation must result in ‘true’ over the complete formula)
   - Sort behaviors according to weight and eliminate behaviors already chosen too often
   - Several behaviors can reach the same goal: Eliminate all but one
Basic Idea – Rule Set

- Encodes conditions for actions, action sequences, and goals.
- Assumed that an action will complete successfully whenever conditions are satisfied.
- But execution of an action is monitored:
  - Conditions can be incomplete
  - Unobservable conditions that block execution
- BNF:

  Goal := Label ":: :<=" Expression ";"
  Rule := Proposition "<=" Expression ";"
  Expression :=
    (Expression "|" Expression) or
    (Expression "&" Expression) or
    ("!" Expression) or
    ("Do(" Action ")") or
    ("Test(" Proposition ")") or
    Proposition
Rule Set - Example

```c
// Small rule set, controlling the example device.
LowE <= Test(Battery.Low) | Test(Battery.Crit);
HaveGuesstimatePos <= Do(COMP.GetBearing)
    & Do(BARO.GetHeight) & Do(CLOCK.GetTime);
HaveGpsPos <= not_Test(GPS.Disabled) & Do(GPS.GetPosition);
HavePosition <= HaveGpsPos | HaveGuesstimatePos;
// Goals
Goal2: <= HavePosition & Do(NF.SendPosition);
Goal1: <= HavePosition & Do(GSM.SendPosition);
Powersave: <= LowE & GPS.Disabled & GSM.Off
    & CPU.ClockDown;
SwitchOnGsmPwr: <= not_LowE & Test(GPS.Disabled)
    & GSM.Off;
SaneShutdown: <= Test(Battery.Crit)
    & ExampleSystem.Off;
Goal0: <= not_Test(Battery.Crit) & ExampleSystem.On;
```
Basic Idea – Set of Activity Profiles

• Forms the second order relation ("desirability")
• Specifies a ratio for reaching a certain goal
• Examples of activity profiles:

Weight is calculated by:

\[ \text{SlopeAtActivityFactor} \times \text{DistanceToMax} \times (1 - \text{DampingFactor}) \]

0 \leq \text{Slope, Maximum, Damping factor} \leq 1
Basic Idea – Algorithm (1)

- Tracks a number indicating how often a rule was run during some timeframe: Activity Factor
- Tracks a number indicating how often a rule failed: Damping Factor
- Using these factors and the knowledge base, the algorithm constantly tries to reach goals.
Basic Idea – Algorithm (2)

1. Look at the ordered list of goals: If the weight indicates a goal has to be reached goto next step. (Else look at next goal.)
2. Look at rule set; If there is a behavior that satisfies a goal goto next step. (Else look at next goal.)
3. Execute the behavior. According to execution results increment Activity and Damping Factor.
4. If steps 1 to 3 have been completed for the list of goals, recalculate the weights and goto step 1. (Else goto step 1 without recalculating weights and look at the next-best goal which still has to be examined.)
Basic Idea – Algorithm (3)

Complexity: In worst case (all goals to be reached, no behavior can be satisfied), the algorithm visits all paths encoded in the knowledge base.

Fault Tolerance: As damping factor influences weight, often failing behaviors will be penalized.

Variation of Behavior: As the activity factor influences weight, the system will choose different behaviors to reach a goal. (If the developer wants.)
Extensions

• Rules may change activity profile functions

• Classifiers that map data to propositions

• Details of these and other extensions are described in the paper
Outline


2. Hardware: “How does the example platform look like?”

3. Methodology: “How do we attempt to solve the problem?”

4. Results: First experiments

5. Q/A
First Experiments

• Platform:
  – Microchip PIC 18F device (4K RAM, 128K Flash)
  – FreeRTOS
  – Simulated actions, linear target activity profiles

• Results:
  – System works as intended
  – Difficulties, Open Questions
    • How choose factors so the system reacts as I want it to?
    • Are there any properties the rule set must implement?
    • (Come up with a more space efficient implementation.)
Thank you for your attention