BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

Fuzzy Regulator for Inverted Pendulum

Intelligent Systems project

Ondřej Lengál, xlenga00@stud.fit.vutbr.cz
Libor Polčák, xpolca03@stud.fit.vutbr.cz
Boris Procházka, xproch63@stud.fit.vutbr.cz
Petr Zemek, xzemek02@stud.fit.vutbr.cz

2008

Contents

1 Introduction **3**

2 Project design **3**

2.1 Experimental frame design 3

2.2 Fuzzy regulator design 4

2.3 Cart and pole design 5

3 Project demonstration **5**

4 Conclusion **6**

1 Introduction

This is a project documentation for Fuzzy Regulator for Inverted Pendulum, an Intelligent Systems project. The task was to design an intelligent component for inverted pendulum regulation using the SmallDEVS framework [2]. This documentation describes project design and the steps necessary in order to run the simulation demonstration.

2 Project design

Our solution is based on the Cart and Pole example presented in [2]. During our development, [1] was a great resource we used to harness the Squeak environment.

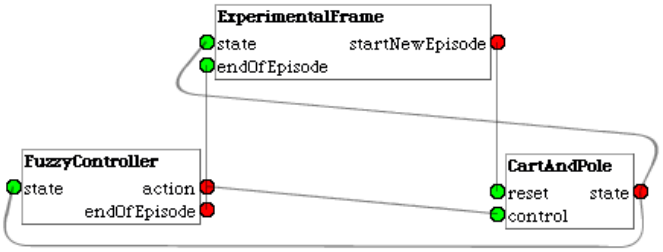

Figure 1: Overall project design.

Figure 1 shows the overall project design. *ExperimentalFrame* and *CartAndPole* are modified components introduced in [2]. The original version was using reinforcement learning to manage the pendulum regulation, so we had to remove rewards from these components and change the force direction computation to allow us sending continous values (to the cart) from the interval $\langle -1; 1 \rangle$ instead of just discrete values from $\{-1, 0, 1\}$. Also, the assignment required us to transform these components (and their subcomponents) to *FSAPrototype* (see [2]).

2.1 Experimental frame design

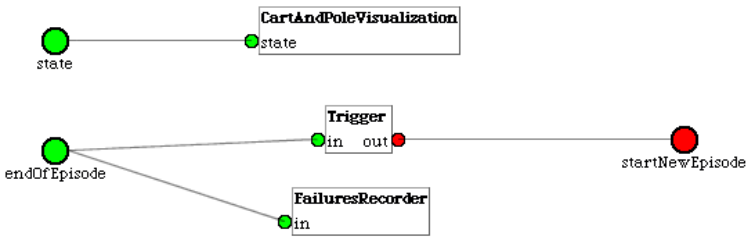

Figure 2: Experimental frame design.

Figure 2 shows the experimental frame design (it is based on [2]). *FailuresRecorder* counts the number of cart and pole failures, *CartAndPoleVisualization* implements visualization for our simulation and *Trigger* just delegates (with a short delay) the "end of episode" request to the output. Formal models for these components were ommited because of their simplicity.

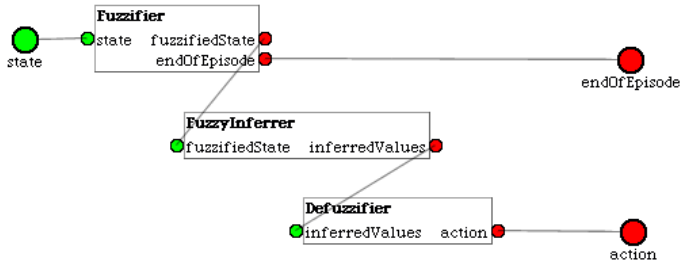

Figure 3: Fuzzy regulator design.

2.2 Fuzzy regulator design

FuzzyController (see figure 3) is composed of three subcomponents. The first one, *Fuzzifier*, transforms the state of the cart into a "fuzzified state", so that each state variable (cart position, cart speed, pendulum angle and pendulum angular velocity) is mapped into a degree of membership for the corresponding linguistic term.

We used the following fuzzy sets:

position (left, centre, right)

speed (left, stopped, right)

angle (left, centre, right)

angular velocity (left, zero, right)

Also, when the cart state is detected to be invalid (i.e. the pendulum has fallen down or the cart has moved out of the permitted range), it sends the "end of episode" signal, the current episode finishes and the simulation is restarted. Figure 4 shows the formal model of this component.

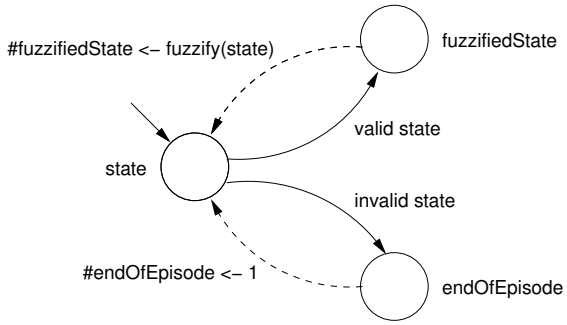

Figure 4: Model of fuzzifier.

The second subcomponent, *FuzzyInferrer*, uses rules to compute the way in which every linguistic term value (from the input of 4) affects the resulting force direction of the cart. We used 13 rules in total. Figure 5 shows the formal model of this component.

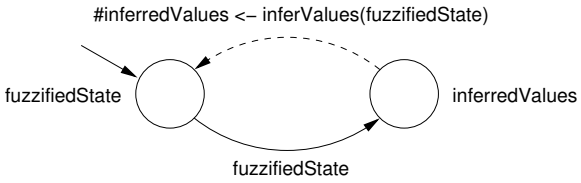

Figure 5: Model of fuzzy inferrer.

The last subcomponent, *Defuzzifier*, merges the results from *FuzzyInferrer* and computes the resulting force direction for the cart that is sent to the output. Figure 6 shows the formal model of this component.

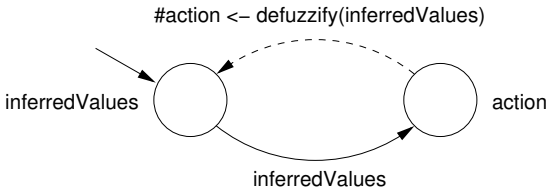

Figure 6: Model of defuzzifier.

2.3 Cart and pole design

This component (it is based on [2]) stores the actual state of the cart and pole. It has two inputs (reset and control) and one output (state). The control value can be from the interval $\langle -1; 1 \rangle$ (force direction, i.e. where should the cart move) and whenever it is changed, the component computes its new state and sends it to the output, which is connected with the fuzzy regulator and with the visualization. Figure 7 shows the formal model of this component.

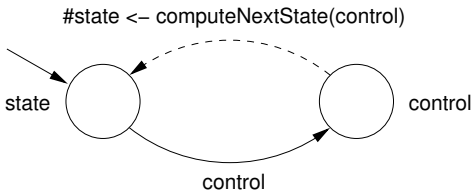

Figure 7: Model of cart and pole.

3 Project demonstration

To run our project demonstration, perform the following tasks please:

1. Follow information in [2] to install the latest SmallDEVS version.
2. Download and install FSA Trait & Prototype, also from [2].
3. Install our simulation file `Root.Simulations.SINProjekt.sar`.
4. File in `SINPprojekt.st`.
5. Execute `SINProjekt demonstration` in a workspace.

Three windows (see figure 8) should immediately pop up — episode durations, cart-pole visualization and demonstration log. During the demonstration, several test cases will be executed, each with different simulation parameters. You can view the demonstration progress in the demonstration log window.

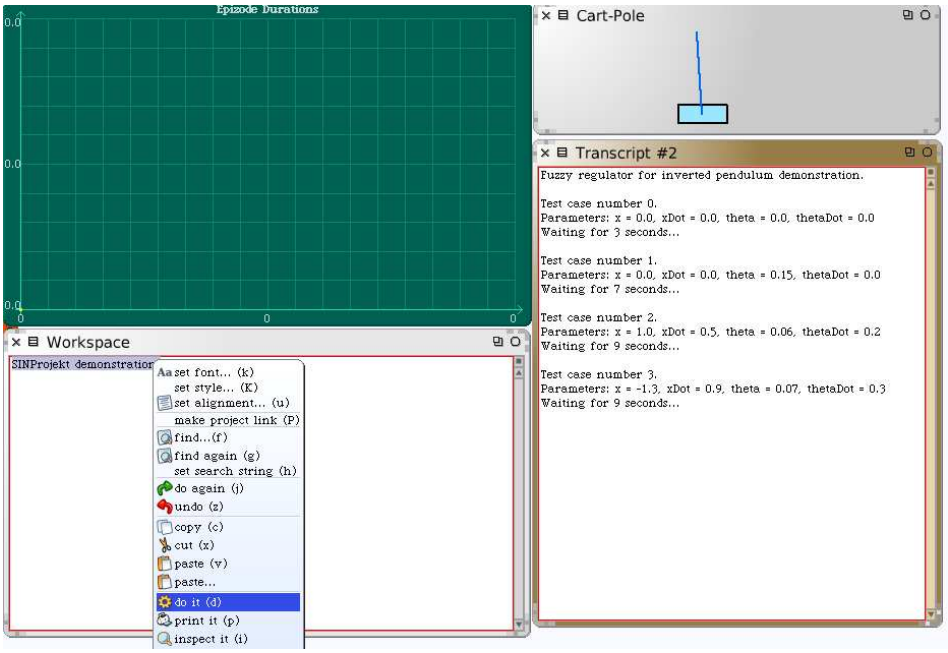

Figure 8: Overall simulation demonstration view.

4 Conclusion

All specifications from the assignment were observed during the project development. All members of our team were interested in the project and critical parts of the design were discussed together. We were focusing on the regulator quality, so that it would be able to control the pendulum in most situations.

The application was successfully tested on the faculty computers (GNU/Linux, x86) and on our systems (GNU/Linux, x86 and x86-64).

References

[1] Oscar Nierstrasz et al. *Squeak by Example*. Square Bracket Publishing, 2004.

[2] Vladimír Janoušek. SmallDEVS: Dynamic DEVS in Dynamic Language. [online]. Available on URL: http://www.fit.vutbr.cz/~janousek/smalldevs/.