A Domain-Specific Language (DSL) for Integrating Neuronal Networks in Robot Control

G. Hinkel, H. Groenda, L. Vannucci, O. Denninger, N. Cauli, S. Ulbrich
L’Aquila, 21-JUL-2015
Bio-inspired robots

- Robots inspired by nature
- Desire to create biologically plausible robot controllers

stick insect

LAURON V

[Roe14]
Connecting Robots to Neuronal Networks

[Image: extremetech.com]

[Image: FZI]
The Human Brain Project

- Unify understanding of the (human) brain
  - Provide ICT platforms
- Transfer this knowledge into products
  - Future Neuroscience
  - Future Medicine
  - Future Computing
- Interdisciplinary cooperation
The Neuro-Robotics Platform (NRP)

- Motivation
- Platform
- Python DSL
- Formal Model
- Transformation
- Conclusion

The Neuro-Robotics Platform (NRP) is a closed loop engine that integrates neuronal networks into robot control. It includes a Neuronal Simulator (Nest), a World Simulator (Gazebo), and a Closed Loop Engine. The platform supports various inputs such as Camera images, Sensors, Membrane potentials, and Control messages.
A basic example experiment

Motivation
Platform
Python DSL
Formal Model
Transformation
Conclusion

EyeSensorTransmit

WheelSensorTransmit

0 1 4
2 3 5
6 7

synapse excitatory
synapse inhibitory
bias excitatory

[Image: Clearpath robotics]

[Brai86]
Components of a running simulation
Designing a DSL

- General-purpose language inappropriate [Fow10]
- Developers do not voluntarily change their primary language [MR13]
- Python popular among neuroscientists & roboticists
  - Goal to make DSL familiar to Python developers

- No limit for expressiveness in the Transfer Functions
  - Internal Python DSL
import hbp_nrp_cle.tf_framework as nrp
from geometry_msgs.msg import Vector3, Twist

@nrp.MapSpikeSink("left_wheel_neuron",
    nrp.brain.actors[0], nrp.leaky_integrator_alpha)
@nrp.MapSpikeSink("right_wheel_neuron",
    nrp.brain.actors[1], nrp.leaky_integrator_alpha)
@nrp.Neuron2Robot(Topic('/husky/cmd_vel', Twist))

def wheel_transmit(t, left_wheel_neuron, right_wheel_neuron):
    linear = Vector3(20 * min(left_wheel_neuron.voltage, right_wheel_neuron.voltage), 0, 0)

    angular = Vector3(0, 0, 100 * (right_wheel_neuron.voltage - left_wheel_neuron.voltage))

    return Twist(linear=linear, angular=angular)
Approaching a formal model

• Neuroscientists often do not have programming expertise

• Efficient failure management
  – Decreased time for errors reporting
  – Increased locality for errors

• Abstract syntax for graphical editor
Combination of DSLs on multiple abstraction levels

uc Create Transfer Function

Neuro Scientist

Create Transfer Function (Python DSL)

Create Transfer Function (BIBI Model)
Using the formal model

• Transformation of Transfer Functions
  – Simulation component (CLE) only needs single input format
  – Expert users can trace the transformation

• Formal model may contain references to Transfer Functions in Python DSL
import hbp_nrp_cle.tf_framework as nrp
from geometry_msgs.msg import Vector3, Twist

@nrp.MapSpikeSink("left_wheel_neuron",
    nrp.brain.actors[0], nrp.leaky_integrator_alpha)
@nrp.MapSpikeSink("right_wheel_neuron",
    nrp.brain.actors[1], nrp.leaky_integrator_alpha)
@nrp.Neuron2Robot(Topic('/husky/cmd_vel', Twist))
def wheel_transmit(t, left_wheel_neuron, right_wheel_neuron):
    linear = Vector3(20 * min(left_wheel_neuron.voltage,
                        right_wheel_neuron.voltage), 0, 0)
    angular = Vector3(0, 0, 100 * (right_wheel_neuron.voltage -
                        left_wheel_neuron.voltage))
    return Twist(linear=linear, angular=angular)
Conclusion

• Internal DSL in Python for Transfer Functions
  – Raises abstraction level of the platform
  – Familiar for Python users

• Formal model for integrating neuronal networks with robot control
  – Advantages in model validation
  – Enabling artifact for graphical editor

• Support of both abstraction levels
  – Transformation approach
  – Allow expert users to use Python DSL
Outlook

• Validation in more realistic scenarios
  – Platform will go live soon
  – Collection of user feedback

• Graphical DSL
  – Abstract syntax already existing
  – Support users with little programming expertise
Thank you for your attention

neurorobotics@humanbrainproject.eu
Conclusion

• Internal DSL in Python for Transfer Functions
  – Raises abstraction level of the platform
  – Familiar for Python users

• Formal model for Integrating neuronal networks with robot control
  – Advantages in model validation
  – Enabling artifact for graphical editor

• Support of both abstraction levels
  – Transformation approach
  – Allow expert users to use Python DSL
## References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Details</th>
</tr>
</thead>
</table>