

A Framework to Control the Darwin-OP Using CLIPS

Brandon Shrewsbury, Ameen Kazerouni, Kenitra Marrow and Dr. Adel Abunawass
Department of Computer Science, University of West Georgia, Carrollton, Georgia, USA

Abstract - *Procedural Paradigms limit the capabilities of an independent system from tackling novel situations. A rule based expert system allows for dynamic expansion of a knowledge base that can be used by an inference engine to challenge a wide variety of situations. The Dynamic Anthropomorphic Robot with Intelligence (DARwIn) proved to be a practical platform for exploring the possibilities of using an expert system to control an independent agent in an uncontrolled environment. Our research revolves around the development of a framework that allows the seamless integration of the C language integrated production system (CLIPS) with the DARwIn framework. This type of control, that approaches a cognitive model rather than a procedural one, allows researchers to focus on expanding the knowledge base of the system and improving functionality and decision making skills on a more abstract level.*

Keywords: DARwIn-OP, Expert Systems, CLIPS

1 Introduction

In 1954 Isaac Asimov stunned the world with the first book of his robot novel series, introducing the world to the concept of Robots that looked and acted like human beings. For decades the minds of children, sci-fi fans and scientists have been entranced by these possibilities. Fiction novels led to sci-fi movies that were coupled with the dawn of anthropomorphic robots. What started off as the pencil sketches of a mad man can now be found in many robotics labs across the world.

The humanoid-research community today, actively researches bipedal movements, kinematics, prosthetics and a wide array of topics many stemming from Asimov's fictional designs. Along with robotics, another topic that emerged was the study of Artificial Intelligence and Human cognition. The scientific world was convinced that if we could build robots that looked like humans, we should be able to program robots to think like humans.

The framework developed in the course of our research investigates the possibility of constructing a localizationist model that could mimic decision making via the use of an Expert system to control the DARwIn - OP.

2 Motivation

Looking at the anthropomorphic form of the DARwIn raises the question; can the DARwIn be programmed to imitate “thinking” or “perceiving”. There are several approaches within Artificial Intelligence used to explore cognition. One of these research areas is pattern matching using an Expert System [2]. An Expert System allows a computer to make decisions by sifting through and recalling knowledge similar to an “expert”, rather than the procedural forms generally employed [5]. Exploring the possibilities of using an Expert System to control the DARwIn's motion will open various avenues for research in the field of human cognition. We wanted to develop a framework that seamlessly integrates with the DARwIn using a CLIPS expert system. This paper summarizes the implementation of the framework and outlines the communication pathways between various aspects of the project.

3 Approach

In order to provide DARwIn with an unobtrusive use of the inference engine we decided to approach the mimicking of higher functions using a localizationist model as it resembles DARwIn's current architecture. The Clips framework was developed as an external component that interfaces with the Darwin framework using an intermediary controller. The Expert System was designed as a component, and not as the controller itself, in order to keep its priority at the same level with other modules. The intermediary controller houses access to all components of the DARwIn and provides communication pathways throughout.

3.1 DARwIn

The Dynamic Anthropomorphic Robot with Intelligence (DARwIn) is a robotics platform developed at the Robotics and Mechanism Laboratory (RoMeLa) at Virginia Tech for research in kinematics, human-robot interaction, and AI [3]. It comes equipped with a dense array of sensors and expansion capabilities suitable for advanced research an almost any robotics field. DARwIn contains an integrated Fit-PC2i computer running Ubuntu.



Figure 1:
DARwIn

3.2 CLIPS

C Language Integrated Production System (CLIPS) is a rule based system that was developed at the NASA-Johnson Space center. [6] CLIPS is fast, efficient, and can easily be integrated with various languages using an extensive API.

3.3 Integration

It was decided to integrate our solution with the source code provided by Robotis as it is uniformly distributed to versions of the DARwIn purchased through Robotis distributors. Versions using LUA have not been tested but the finite state machine can be adapted to interface with our framework.

4 Implementation

There are several segments that must communicate with each other to facilitate the functionality of the framework. The Expert System (ES) must receive data from the DARwIn, representing the current state from a high level. This is done by asserting either pre-processed sensor data or the running state's current classification into the CLIPS fact base. Once information has been asserted, the ES can analyze and determine the next state needed. Instructions are sent back to the intermediary controller for post processing and execution.

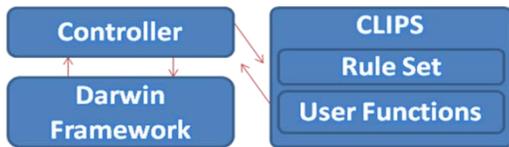


Figure 2: Framework Communication

Development of the framework was broken up into four sections:

- Integrating the CLIPS Library
- Developing communication pathways between CLIPS and the DARwIn Framework
- Creating an output control interface for CLIPS
- Developing a rule set to test the framework

4.1 CLIPS Build

CLIPS version 2.4 was used in this deployment. Installation was not necessary as the source was compiled in library format.

4.2 Communication

Information is transmitted on a pseudo publish-subscribe framework using function pointers. Potential listeners subscribe to “events” by passing function pointers to the controlling classes defined subscribe function. This allows for an easily expandable and lightweight coupling between classes without the overhead of raising events. Function pointers link CLIPS’s user functions and the Darwin’s

framework to the intermediary controller. In both instances, communication is one way.

4.3 Control Interfaces

CLIPS provide a modifiable interface to allow for creating user defined functions. This feature is used for outgoing communications from CLIPS to the intermediate. Declaring user defined functions is facilitated by: DefineFunction(functionName,functionType,functionPointer,actualFunctionName); where the function name is the accessible name used within CLIPS. This links a LISP like call with an external call. Using this, the ES can call the defined function and interact with the intermediary controller to route commands back to the DARwIn framework. In order to decouple the CLIPS library, user defined function can be defined at runtime and stored outside of the library. Code examples can be viewed in Figure 3.

```
C++:
void setDarwinToIdle{
    Walking::GetInstance().X_MOVE_AMPLITUDE = 0;
    Walking::GetInstance().Y_MOVE_AMPLITUDE = 0;
}

int main()
{
    //Initialize CLIPS
    DefineFunction("darwinIdle", "v", PTIF setDarwinToIdle, "setDarwinToIdle");
}

CLIPS:
(defrule SendStop
    "Tell Darwin To Stop"
    ?f2 <- (Walk)
    ?f1 <- (stepsToTake(stepsLeftToTake ?v))
    (test(eq ?v 0))
    =>
    (retract ?f1)
    (retract ?f2)
    (darwinIdle)
    (assert(idle))
)
```

Figure 3: External Call

4.4 Rule Set

The CLIPS Expert System has two major components; the knowledge base and an inference engine. The knowledge base consists of facts that are representative of the current state of the DARwIn and production rules. These rules represent heuristic knowledge and follow an “if-then” protocol coupled with the active facts stored in the knowledge base [4]. This implies that if a certain combination of facts has been asserted; the system “infers” a need to update the knowledge base and push new events to its agenda. A portion of the rule-base is displayed in Figure 4:

```
(defrule SendWalk
    "Assert Walk Intent - Remove idle State"
    ?f1 <- (stepsToTake)
    ?f2 <- (idle)
    =>
    (assert(Walk))
    (retract ?f2))
```

Figure 4: Rule Set

Once the production system identifies intent, rules are instantiated and added to the agenda. A rule is instantiated when all of the patterns that constitute the premise of the rule can be matched with facts present in the knowledge base. The inference engine then executes these rules based off various factors that determine the rules priority. Currently no salience was given to any one rule in our rule set. Depending on the construct of the rule, the output may modify the knowledge base or interact outside the confines of CLIPS.

5 Framework Testing

Our chunk primarily demonstrates the viability of using an Expert System to control the Darwin framework. Ideally, the DARwIn's thought process would be broken up in to numerous ESs. Each ES would be an independent component that dealt with domain specific tasks; such as, walking, vision, or auditory systems [1]. The chunk we developed for testing the framework superficially dealt with controlling motion

5.1 Workflow

Our motion control rule set tracks and manages the walking and turning of the DARwIn using state representation of the motion manager and externally registered intent. A number of steps to take or number of degrees to turn would be added to the agenda and the ES would set movement amplitudes and initiate walking. Once the DARwIn begins to move, the intermediate controller would receive updates on the movement from lower levels and relay data to the ES for monitoring. Once the ES has identified completion of the task, it would adjust the movement amplitudes and stop walking. Figure 4 traces various paths the rule-based system follows when intent has been registered.

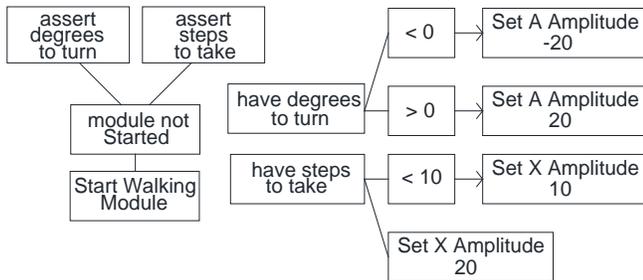


Figure 4: Rule Set – Movement Initiation

5.2 Tracking Movement

Every two steps the DARwIn takes results in a cycle through four phases; Standing -> leg raise -> standing -> leg raise. Following completion of the 1st and 3rd phases, a call is made notifying the controller of a phase change. The controller passes this information to the ES which incorporates it into the knowledge base. Once the information is logged, the ES reruns the agenda to decide when to modify the state of the DARwIn.

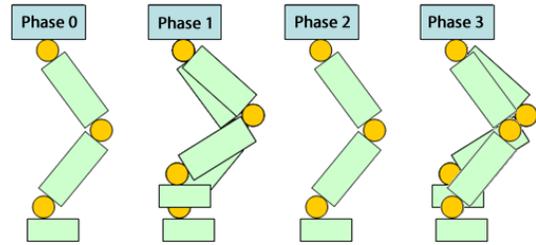


Figure 5: Walking Phase depiction – No forward movement

Table 1: Phase Transitions

Phase 0 - 1: Ankle dorsiflexes, knee is flexed, leg elevates
Phase 1 - 2: pelvis rotates and leg moves forward, knee is extended and foot contacts surface
Phases 2 - 3: Repetition of phases 1 and 2 using opposite leg

5.3 Example Integration

To facilitate our testing, both steps to take and degrees to turn were asserted arbitrarily by the intermediary controller. In order to maintain relative awareness of phase changes efficiently, the motion manager was modified to communicate openly about step phases. The walking module was given an access point, linking to the intermediary controller. With every alternate phase change an event was raised, sending an alert to the ES through the controller.

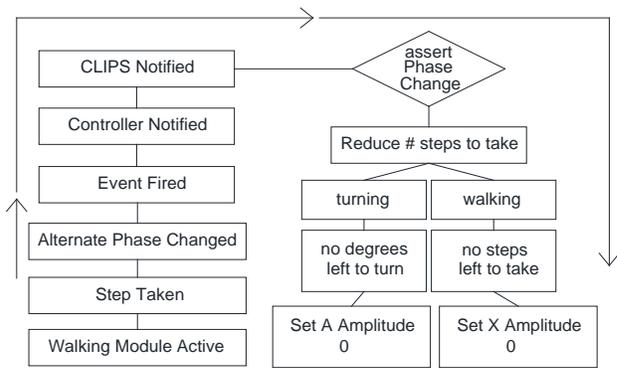


Figure 6: Rule Set – Movement Completion

6 Framework Setup

Clips integration requires minimal modifications to the CLIPS library and the DARwIn Framework. User Defined functions must be defined in the CLIPS library to match the execution requirements of the output space. This step is critical to ensuring proper feedback from the ES. The input space of the ES will pull from an array of data throughout the system and may require modification of the DARwIn's source code to include state listeners. Once communications have been properly defined, the intermediary controller can load the rule set and information can be polled and pushed to the ES through CLIPS's API.

7 Results

A framework was developed that could successfully integrate CLIPS with the DARwIn's framework. The framework was able to control the DARwIn's walking module by sending instructions to the controller based off decisions inferred by the Expert System. The Expert System itself is scalable and can be easily replaced with other Expert Systems of the user's choice. Slight changes to the DARwIn's framework will allow manipulation of almost any aspect of the DARwIn via "expert" control.

8 Acknowledgements

We would like to thank Edwin Rudolph for his guidance and support throughout the development process.

9 References

- [1] John R. Anderson. 1988. The expert module. In M. Polson & J. Richardson (Eds.), *Handbook of Intelligent Training Systems*. Hillsdale, NJ: Erlbaum, 21-53.
- [2] John R. Anderson. 2000. *The Structure of Memory, Learning and Memory: An Integrated Approach*. Hoboken, NJ: Wiley. Print, 203-211
- [3] DARwIn OP: Open Platform Humanoid Robot for Research and Education [Online] http://www.romela.org/main/DARwIn_OP:_Open_Platform_Humanoid_Robot_for_Research_and_Education (2012)
- [4] William J. Clancey. 1981. *The Epistemology of a Rule-Based Expert System: a Framework for Explanation*. Technical Report. Stanford University, Stanford, CA, USA.
- [5] R. M. Wygant. 1989. CLIPS - a powerful development and delivery expert system tool. *Comput. Ind. Eng.* 17, 1 (November 1989), 546-549. DOI=10.1016/0360-8352(89)90121-6