Prolog on Microcontrollers

Samuel Hicks

Supervisor—Dr Oliver Ray

I implemented Warren's Abstract Machine for the ESP32 microcontroller, a small computer without an operating system. I also implemented a Prolog compiler for this system, allowing a user to execute Prolog programs onboard the microcontroller, including hardware interaction. This brings a powerful, easily provable and robust language to low level hardware

Prolog

Prolog is a declarative logic programming language.

A prolog program is expressed in terms of relations, represented as facts and rules.

Computation is initiated by running a query over these relations. A query can be used to both generate and verify solutions.

In the case of multiple solutions, Prolog performs a depth first search, backtracking on failure.

Prolog also natively has a powerful pattern matching system.

Thus complex search problems, such as ones commonly encountered in AI, can be expressed reasonably simply.

It is also far easier to prove correctness in Prolog that many other languages

Warren's Abstract Machine (WAM)

Unfortunately, Prolog in not suitable for executing on hardware.

Prolog is expressed in names, facts, and logical combination (disjunction and conjunction), whereas hardware typically involves memory locations, fixed-size numbers, and arithmetic operations.

In 1983 David Warren designed an abstract machine for the execution of Prolog consisting of a memory architecture and an instruction set.

Even this machine however is too complex to be natively executed on hardware. As such, programs and queries are compiled to a byte code which in interpreted by compiled C++ code running on the microcontroller.

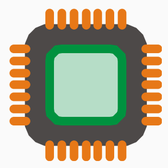
Memory Management

There are a number of hurdles to overcome when writing a WAM for microcontrollers. Most notable is the issue of memory management, particularly garbage collection.

As such, I implemented a memory system based on Professor May's SURE architecture, featuring a concurrent garbage collector.

This also allowed me to combine the separate memory spaces described by the WAM, making more efficient use of the limited onboard memory.

Hardware Interaction



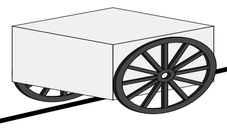
Prolog does not natively support hardware interaction.

I extended the WAM instruction set to support both digital and analog GPIO.

This allows the programmer to describe logical relations between current and future states of the hardware in a logically provable manner

Further Work

- Multiple precision integers, i.e. integers larger than the processor word size. With the new memory system, this task becomes easier.
- User interaction. At the moment, the only user interaction in submitting queries and reading the result. Predicates that request user input from and display messages to the terminal machine would provide a richer experience.
- Dynamic predicates. A standard Prolog extension is to add support for asserting and retracting rules at runtime. There is adequate space in non-volatile storage (i.e. Flash) to store predicates, providing durability



University of BRISTOL

Acknowledgements

- Professor David May
- Dr Steve Moyle