



Università degli Studi di Bari
Dipartimento di Informatica



LACAM
Machine Learning

Rule and Knowledge-Based Systems

A brief introduction

Antonio Vergari

April 10, 2015

Context & Background

Rule-Based Systems

Different kinds of use for rules in the programming world¹:

derivation as in deductive systems and theorem provers

transformation as in rewriting systems, grammars

constraint declaration as in Business Rules Management Systems (BRMS)

reaction as in the ECA paradigm, e.g. db triggers

Rule-based programming falls into the **definitional programming approach** (or declarative programming paradigm, like *logical programming* and *purely functional programming*). Programmers write rules, demanding a rule **inference engine** to manage, activate, process them.

There are also meta-languages to express and serialize rules: **RuleML**.

¹ <http://www.w3.org/2000/10/swap/doc/rule-systems>

Rule Engine Architectures

what you are expected to know

Knowledge Base
(Rules)

Inference Engine

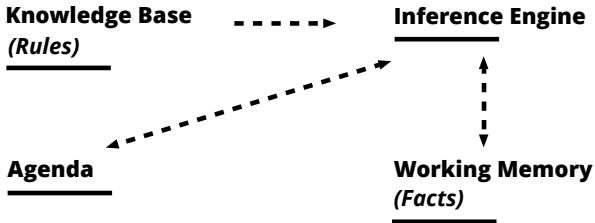
?

Agenda

Working Memory
(Facts)

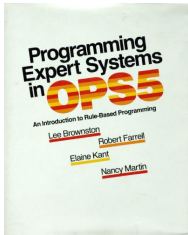
Architectures

what you are expected to know

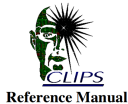


Expert System Shells

A little bit of history



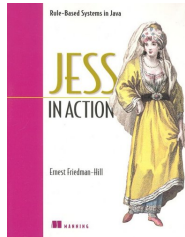
1981



Volume I
Basic Programming Guide

QuickSilver Beta
December 31st 2007

1985



1995



2001

CLIPS

C Language Integrated Production System.

A **forward chaining** rule language based on the **Rete** algorithm, created in 1985 at NASA's Johnson Space Center.

It became an **expert system shell**, i.e. an environment to write expert systems (it can also be used for fast prototyping).

Written in C, but resembling LISP (*embrace parentheses!*).

Multi-paradigmatic: rule programming + functional programming aspects + oop.

Embeddable: C APIs (but also wrappers in Java, python, .Net...).

Current version 6.30, released 2015/03/25, 6.24 is fine as well (installed in labs).

Applications

Formal Systems

Formal languages and **grammars** are an example of formal systems you already met. Rules encode how to produce or *rewrite* symbolic knowledge.

$$S \rightarrow ASb, A \rightarrow a, A \rightarrow \lambda$$

The first attempt to embed *all mathematical truths* into a single formal system is to be found in **Principia Mathematica** (1910-1927). We know **how it ended...**

Nowadays' applications are software theorem provers and model checking, exploited in formal software verification.

Expert Systems

Interpretation	Hearsay (Speech Recognition), PROSPECTOR
Prediction	Preterm Birth Risk Assessment
Diagnosis	CADUCEUS, MYCIN, PUFF, Mistral, Eydenet, Kaleidos
Design	Dendral, Mortgage Loan Advisor, R1
Planning	Mission Planning for Autonomous Underwater Vehicle
Monitoring	REACTOR
Debugging	SAINT, MATHLAB, MACSYMA
Repair	Toxic Spill Crisis Management
Intruction	MH.PAL, Intelligent Clinical Training, STEAMER
Control	Real Time Process Control, Space Shuttle Mission Control

Not all are rule-based.

Soar

Soar is a cognitive architecture, created by Laird, *Newell*, and Rosenbloom.

It is the embodiment of an intelligent agent system.

It is both a view of what cognition is and an implementation of that view through a programming architecture for AI modeling different aspects of human behavior.

It is based on a production system, it uses explicit **production rules** to govern its behavior. An example rule to model plan intentions²:

```
sp {top-ps*elaborate*task*belief*intend*true
  (state <s> ^problem-space.name top-ps ^agent-name <me> ^plan <task>)
  (<task> ^intend true ^responsibility <me> ^authorized yes)
-->
  (<task> ^belief true)
}
```

²<http://people.ict.usc.edu/~traum/Talks/ict-dm-tutorial5.pdf>

Rule based Game AIs



Resources

Books

Please do not study from these slides.

Expert Systems: Principles and Programming

J. Giarratano & G. Riley. Course Technology. 4th Edition. 2004.

From the programmers of CLIPS, useful and general enough to get confidence with the language.

Chapters 6-10, 12

Introduction to Expert Systems

Peter Jackson. Addison-Wesley. Third Edition. 1998.

Less CLIPS-centric, but heavier on expert system design and implementation issues. It will come handy for the exam.

Chapters 10-12, 16-17

CLIPS Programming

Assuming version 6.30, there are equivalent for version 6.24.

CLIPS User's Guide

Most of the basic arguments we will face can be found in this tutorial. A must.

[documentation/v630/ug.pdf](#)

CLIPS Basic Programming Guide

It contains the documentation and examples for each shell and language construct.

[documentation/v630/bpg.pdf](#)

CLIPS Advanced Programming Guide

Explaining in depth the source code, the modules functioning and how to use CLIPS APIs from a wrapper program (in C).

[documentation/v360/apg.pdf](#)

CLIPS Programming

Projects for embedding CLIPS into external frameworks. Beware outdated software.

clipsmm

C++ wrapper of CLIPS C APIs.

clipsmm@sourceforge

CLIPSnet

Embedding CLIPS in to .NET applications (bleargh).

clipsnet@sourceforge

DROID-CLIPS

Porting CLIPS to Android.

droid-clips@github

pyCLIPS

Python 2.X wrapper.

pyclips@sourceforge

CLIPS JNI 0.4

Official Java Native Interface for CLIPS. We'll use it later in the course.

CLIPS-JNI@sourceforge

Additional Resources

CLIPS Forum

Sourceforge project discussion boards

CLIPS@sourceforge

CLIPSESG

CLIPS Expert System Group, a much more updated forum for users to ask for help and interact.

CLIPSESG@googlegroups

Exam