THE UNIVERSITY OF BIRMINGHAM
SCHOOL OF COMPUTER SCIENCE

The Cognition and Affect Project

The SimAgent TOOLKIT
For Philosophers and Engineers
(And Some Biologists, Psychologists and Social Scientists)
(Previously known as SIM_AGENT)

Aaron Sloman

Slide Presentation on SimAgent

Demonstration movies

Apologies for some broken links:
we failed to ensure that some files whose owners left had been preserved.

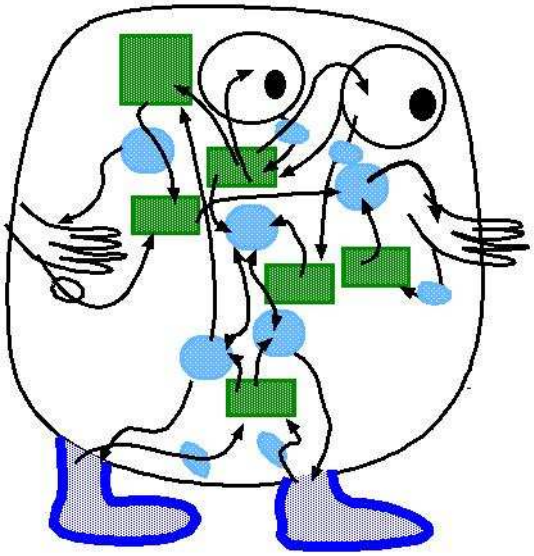
Some External Pointers to the Toolkit

This toolkit is referenced at various web sites. A sample found with the help of google is listed here.
(Apologies for broken links: Probably quite out of date now.)

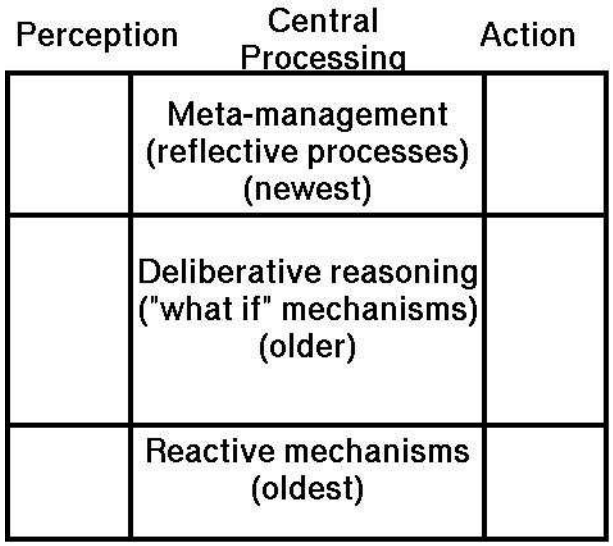
INTRODUCTION

The SimAgent toolkit (originally called SIM_AGENT) provides a range of resources for research and teaching related to the development of interacting agents in environments of various degrees and kinds of complexity. It can be run as a pure simulation tool, or installed in a robot with a sufficiently powerful on-board computer, e.g. running linux. It was originally developed to support exploratory research on human-like intelligent agents, but has also been used for student projects developing a variety of interactive games and simulations.

Unlike many so-called 'agent toolkits', like PRS/Jack, Mozart, Alice, and several more, that are aimed mainly at development of systems involving large numbers of highly distributed fairly homogeneous relatively 'small' agents, SimAgent can be used for such purposes (and was used in that way for a while by Matthias Scheutz at Notre Dame University) but (like ACT-R, COGENT, and the original SOAR) SimAgent is primarily designed to support design and implementation of very complex agents, each composed of very different interacting components (like a human mind) where the whole thing is embedded in an environment that could be a mixture of physical objects and other agents of many sorts, as half-jokingly depicted here:



For example the main motivation behind the development of SimAgent was originally to support research on an increasingly complex sequence of agent types, that led to the development of the biologically inspired CogAff architecture Schema depicted in the image below, showing layers of sophistication superimposed on different types of functionality represented by the columns:



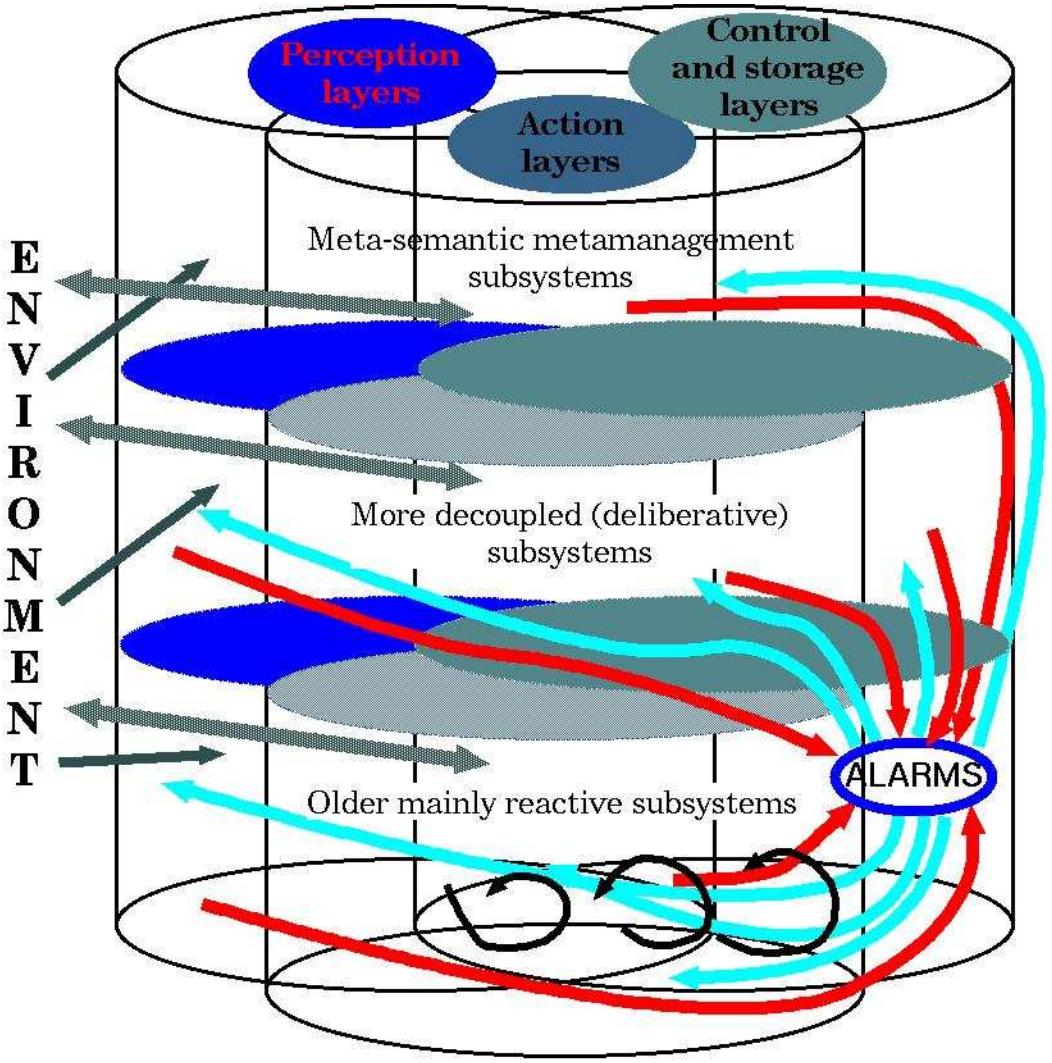
The CogAff Architecture Schema

That schema accommodates a wide variety of specific architecture types, which differ according to which mechanisms and information structures occur in which boxes, and how they are connected to one another and to the environment, as described in this overview.

The simplest architectures (e.g. microbe-like or insect-like agents) use only the bottom (purely reactive) layer of the CogAff schema, whereas more sophisticated architectures use all the layers (the top ones are the ones that evolved latest).

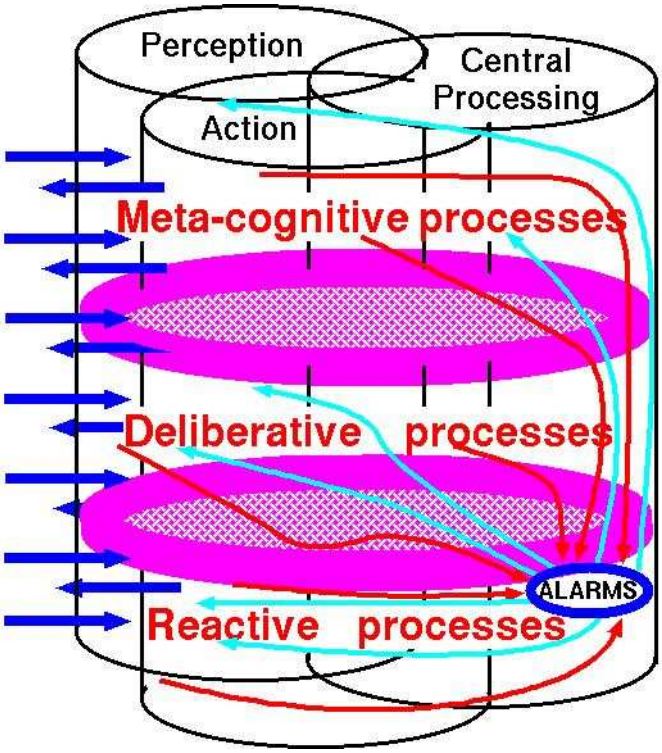
The above diagram is misleading in various ways because it suggest that the perception mechanisms and action mechanisms are separate from each other and can only communicate via the 'central' mechanisms, whereas it is clear (as James Gibson pointed out in his 1966 book *The Senses Considered as Perceptual Systems*) action and perception are deeply integrated, e.g. the constant use of saccades, changes of vergence, changes of focus in vision, and the need to move your hand when it is used to perceive shape, texture, weight, flexibility, hardness, etc. of objects.

So a more accurate, but less clear depiction of the ideas in the CogAff architecture schema is the following (with thanks to Dean Petters, for help with this diagram):



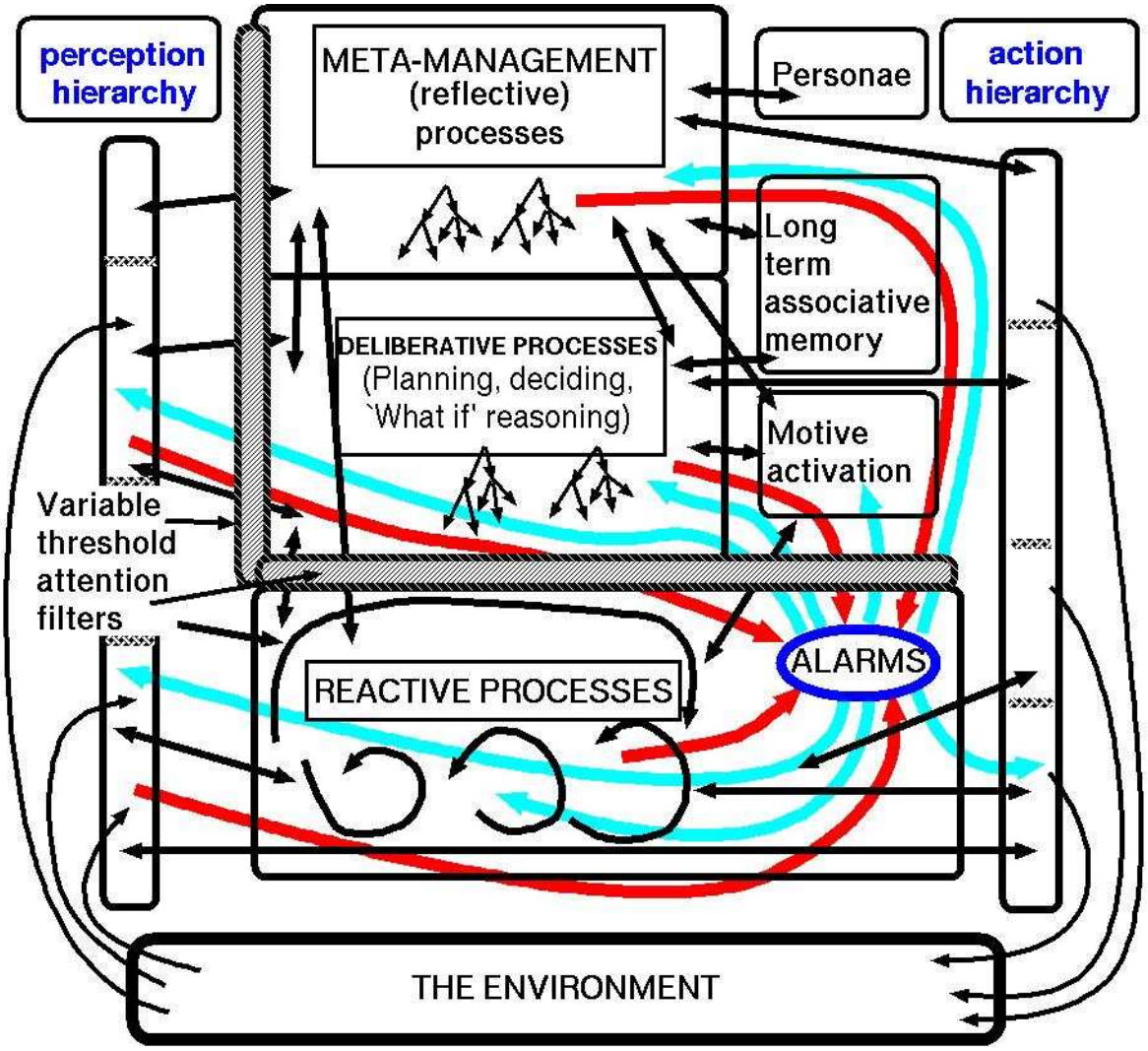
Revised, more realistic CogAff Architecture Schema,
e.g. with deeper integration between action and perception
The horizontal discs represent (usually "fuzzy" boundaries between different levels of functionality. It is possible for some of the information-processing mechanisms to straddle two or more layers.

A different view showing how action and perception are integrated, as pointed out by J.J.Gibson in 1966



A slightly different, possibly clearer presentation of the same ideas as above.

A special case is the conjectured human-like H-CogAff architecture depicted in a sketchy manner below, with components described in more detail in the overview and in papers and presentations on the CogAff web site:



So far nothing as complex as that has been implemented, though various fragments have been, by collaborators, research fellows and PhD students. Some of our undergraduates and MSc students have also used various parts of the toolkit, e.g. the 'Popracer' undergraduate project:
http://www.cs.bham.ac.uk/research/projects/poplog/popracer/

Examples of the 2-D graphical interface tools available can be found here
http://www.cs.bham.ac.uk/research/poplog/figs/rclib/
Examples of movies demonstrating the toolkit can be found here:
http://www.cs.bham.ac.uk/research/poplog/figs/simagent.

MECHANISMS AND DESIGN PHILOSOPHY

The mechanisms are described in detail in the online documents and slide presentations listed below.

AVAILABILITY AND RESOURCES REQUIRED

The toolkit is available as part of Poplog in the FreePoplog directory
http://www.cs.bham.ac.uk/research/poplog/freepoplog.html
The most widely used version is for 32-bit Linux (though there is also a 64 bit version).

http://www.cs.bham.ac.uk/research/projects/poplog/latest-poplog
(includes download instructions).

Also available for non-linux users (e.g. MSWindows, or Mac) in a Virtual Box Package
http://www.cs.bham.ac.uk/research/projects/poplog/ova

Because of its implementation using Pop11 in Poplog it is relatively light weight, and teaching demos and student projects have been run on linux machines with only 64MB RAM. Of course, the amount of memory required at run time depends on what is built with it.

The download package is under 20MB and when installed occupies a directory of less than 90MB -- the exact size depending on the number and type of saved images created. (The defaults include images for running Prolog, Common Lisp, and Standard ML.)

The total size can be much smaller, if unwanted languages in Poplog, the system sources, and documentation are removed, e.g. for running on a small robot.

External libraries, e.g. for vision, or for speech generation (e.g. using 'espeak' on linux) can add to the size required.

Like everything else in Poplog, SimAgent is free of charge with open source, including tutorial examples and documentation. It runs in the Poplog multi-language environment, on a variety of unix and linux systems. A version without graphics will work on Windows PCs. The full system can run on Windows using VirtualBox.

The OpenPoplog project aims to port the full functionality to Windows+PC. However, with the use of virtualisation, e.g. using the VmWare environment or VMFusion or VirtualBox, full Linux Poplog, including SimAgent with graphics, can run under both Windows and Apple OSX (on Intel-based Macs), as illustrated here: http://www.cs.bham.ac.uk/research/projects/poplog/ova/

SOME BACKGROUND

The toolkit was developed within the Cognition and Affect project, at the University of Birmingham, partly in collaboration with users at DERA Malvern (especially Jeremy Baxter and Richard Hepplewhite), who made significant suggestions for improvements during the early stages. Some of the ideas were inspired by Jonathan Cunningham, with whom I interacted on a user-interface design tools project circa 1991-1993. Many others have contributed since then. The list of users, below, includes contributors.

The CogAff project paper directory includes papers reporting on theoretical work and preliminary designs. There are also slide presentations (in PDF and Postscript) presenting the theoretical ideas including a slide presentation on the SimAgent toolkit:
http://www.cs.bham.ac.uk/~axs/misc/talks#simagent

MOVIES
Some mpeg movies illustrating simple uses of the toolkit can be found here
http://www.cs.bham.ac.uk/research/poplog/figs/simagent/

Although SimAgent was originally developed to simulate multiple agents and/or individual agents with multiple concurrently active interacting sub-mechanisms, it was mainly used on a single CPU (although some users haVE distributed processes communicating via sockets). More recently there have been two developments supporting distributed versions of SimAgent

- A paper on the use of HLA to distribute SimAgent,by Mike Lees, and Brian Logan (University of Nottingham) and Georgios Theodoropoulos and Ton Oguara (University of Birmingham), was voted 8th best out of 78 at the 2003 Euro-SIW conference.

- the SWAGES package, developed at the University of Notre Dame by Matthias Scheutz (now at Tufts University) and students allowed SimAgent to be distributed over different computers and interfaced with other packages.

OBJECTIVES AND DESIGN PHILOSOPHY

For our work exploring architectural design requirements for intelligent human-like agents, and other kinds, we need a facility for rapidly implementing and testing out different agent architectures, including scenarios where each agent is composed of several different sorts of concurrent interacting sub-systems, in an environment where there are other agents and objects. Some agents should have sensors and effectors, and some should be allowed to communicate with others. Some agents should have hybrid architectures including, for example, symbolic mechanisms communicating with neural nets. We also wanted to be able to use the toolkit for exploring evolutionary processes, as in the "Blind and Lazy" scenario mentioned below.

It was important for us that the toolkit did not embody a commitment to a particular type of agent architecture, since the whole point was to support exploration of a variety of architectures.

We also required a development environment supporting rapid prototyping (to help with clarifying ill-defined problems), incremental compilation, changes to code without having to restart a process, support for a wide variety of AI techniques and programming styles, and tools for graphical tracing via displays of a running simulation, and graphical interaction via a mouse -- sometimes described as "direct manipulation".

We wanted it to be easy to do fairly simple things, while also supporting more complex and difficult programming tasks.

As a result of analysing requirements implicit in some of our theoretical work (described in papers in the "Cogaff" project directory, and in some of our conference and seminar presentations) a set of ideas for a toolkit emerged. The first draft was made available in October 1994, and a paper about it presented at ATAL95. Since then there have been several extensions. Some of these arose from suggestions made by Jeremy Baxter and Richard Hepplewhite who were using the toolkit for their work on simulated army agents at DERA Malvern. Their work is briefly summarised in a panel alongside a paper on Sim_agent, in *Communications of the ACM*, March 1999.

PROGRAMMING PARADIGMS USED IN SIM_AGENT

It was found useful to combine a variety of programming paradigms and mechanisms within the toolkit. This is why Pop-11 in the Poplog multi-language environment was chosen as the core implementation language.

Pop-11 is a multi-paradigm, extendable, incrementally compiled language, very similar in power and diversity to Common Lisp, but with a more conventional syntax and some other differences. E.g. it has no interpreter, only an incremental compiler.

Here are some of the programming paradigms we felt were necessary for our architectural explorations:

- Object oriented programming (including generic functions and multiple inheritance)
  Based on the ObjectClass extension to Pop-11 (similar to CLOS).

- Rule-based programming and pattern matching
  Based on the Pop-11 pattern matcher and the Poprulebase library

- Concurrent processes (threads)
  Based on a rule interpreter that runs for a given number of "cycles" then suspends itself.

- List processing (as provided in Pop-11 -- similar to Common Lisp lists),

- Event driven programming (e.g. handling mouse events, or new percepts)
  Based on the rich interface between Pop-11 and the X window system, e.g.
  http://www.cs.bham.ac.uk/research/poplog/figs/rclib
  http://www.cs.bham.ac.uk/research/poplog/doc/popref/x
  http://www.cs.bham.ac.uk/research/poplog/doc/popxref/xtoolkit

- Conventional procedural and functional programming (provided in Pop-11)

- Reflection and meta-management, allowing programs to have easy access to their own rules and to monitor and alter their own internal processing.
  In particular, an agent's mental mechanisms may use rules that are in one of the agent's databases on which rules operate. This was used by Catriona Kennedy in her work on architectures for collaborative intrusion-detection agents.

  Allowing rules that operate on internal databases within agents to be *contained in* those databases also makes it easier to design agents that develop new capabilities, or agents which can be dynamically extended via new "plug-ins").

- Other computational paradigms needed for particular applications,
  e.g. neural nets, logic programming
  (Using the ability of Poprulebase conditions or actions to invoke arbitrary mechanisms, and the fact that Poplog supports an implementation of Prolog.)

- Extendable syntax and semantics (macros and beyond)
  Using Pop-11's powerful syntactic extension facilities, possible because the compiler sub-routines are all available at run time.

- Incremental compilation with dynamic replacement of modules
  (essential for rapid prototyping and interactive testing and debugging).

- Automatic store management and garbage collection. (particularly fast in Pop-11).

The use of the language Pop-11 makes it far easier to combine all these paradigms than if we had used any other language, apart from Common Lisp.

The current version of the toolkit is very general and flexible, though perhaps not as easy to use as a toolkit dedicated to a particular type of architecture.

For applications requiring a graphical interface, e.g. displaying moving simulated agents, or with buttons, sliders, menus, control-panels, etc. the RCLIB package is available. There is a library which links RCLIB to Sim_agent, called SIM_PICAGENT.

(Note: RCLIB is an object-oriented extension of the old RC_GRAPHIC package providing 'Relative Coordinate' Graphical facilities. All drawing is relative to a changeable coordinate frame, allowing scale-changes, zooming in and out, moving windows into a simulation, etc.)

By default sim_agent runs as a single process within which multiple "threads" manage different objects and agents and also different components within the "minds" of individual agents. It can also be used in distributed mode, using the Pop-11 socket mechanism to support communication between instances of sim_agent.

Pop-11 and Poplog

The toolkit uses the Pop-11 language in the Poplog software development environment. Pop-11, like Common Lisp, is a powerful extendable multi-purpose programming language supporting multiple paradigms. Within the Poplog environment Pop-11 also supports programs written in Prolog, Common Lisp or Standard ML.

Poplog and Pop-11 previously formed part of an expensive commercial product, but they are now available, free of charge, with full sources, courtesy of the Sussex University. Sim_agent is available at the same site.
Information about teaching materials for programming and AI in Poplog can be found here. ("Poplog" is a trade mark of the University of Sussex.)

Some Birmingham users of SimAgent (Also contributors)

- Darryl Davis, who worked with A.Sloman during 1994-5 has been using the toolkit to explore more complex hierarchical agent architectures, also including mixtures of reactive and deliberative components.
- Brian Logan used and contributed to SimAgent while he was at Birmingham, and continued using it for research and teaching after moving to Nottingham University, including linking SimAgent to the MASSIV Virtual Reality system, and investigating distribution mechanisms in the PDES-MAS project.
- After spending a year here, and contributing ideas to the toolkit, Matthias Scheutz used it for teaching and research at The University of Notre Dame (USA). (He is now at Indiana University). His Simworld package, used, for example, in this online paper (The Role of Signaling Action Tendencies in Conflict Resolution, in *Journal of Artificial Societies and Social Simulation* vol. 7, no. 1) is available at the Poplog web site.
- Catriona Kennedy
- Nick Hawes
- Manuela Viezzer
- Dean Petters

Other users include undergraduates doing AI projects, MSc students and PhD students, including contributions demonstrated in SimAgent movies.

There are additional users in the list of referrers to SimAgent

DOCUMENTATION AND PUBLICATIONS

Descriptions of the motivation for the toolkit, some uses of the toolkit, and the toolkit itself, can be found in the following documents:

1. A detailed (long) plain text description of the Pop-11 sim_agent library.

2. A (long) plain text overview of the "rulesystems" mechanisms available for specifying internal agent architectures (which we have plans to modify extensively in order to increase flexibility including self-monitoring and self-control by agents),

3. A (long) plain text overview of the Poprulebase library, which implements a very flexible and extendable forward chaining production system language, for specifying condition action rules in symbolic or hybrid architectures.

4. http://www.cs.bham.ac.uk/research/poplog/newkit/sim/help/newkit This gives an overview of the changes introduced in July 1999 to allow greater self-reference (reflective capabilities) in agents. I.e. they can now more easily monitor and change their own processing architecture.

   The news files report extensions and bug-fixes.

   E.g. extensions in July 1999 made it easier to use the toolkit to create agents which can observe and alter their own processing architecture. These changes are described in the NEWKIT document listed below.

   More recent extensions proposed partly by Steve Allen and Catriona Kennedy allowed rules more easily to access databases of other agents, and extended the facilities for self-monitoring. See the NEWS files listed below.
5. http://www.cs.bham.ac.uk/research/poplog/newkit/sim/help/sim_agent_news
   http://www.cs.bham.ac.uk/research/poplog/newkit/prb/help/prb_news
   These two files give information about the latest changes to Sim_agent and Poprulebase
   http://www.cs.bham.ac.uk/research/poplog/rclib/help/rclib_news
   gives information about changes to the graphical subset of the toolkit, RCLIB.

6. Building cognitively rich agents using the SIM_AGENT toolkit,
   (in **Communications of the Association of Computing Machinery**, March 1999, vol 43, no 2, pp. 71-77) by Aaron Sloman and Brian Logan.

7. Three papers by Jeremy Baxter and Richard Hepplewhite, Brian Logan and Aaron Sloman presented at the AAAI-98 Workshop on Software Tools for Developing Agents at AAAI-98 Madison, Wisconsin, July, 1998

8. The RCLIB 2-D graphical toolkit used for control panels and interactive graphical displays of simulated agents. Examples of RCLIB displays can be found in
   http://www.cs.bham.ac.uk/research/poplog/figs/rclib/

   Some older documents:

9. A set of seminar slides, presented in 1994, (PDF)

10. A PDF version of a conference paper on the toolkit, by Aaron Sloman and Riccardo Poli, presented at the ATAL-95, Workshop on Agent Theories, Architectures, and Languages, IJCAI-95, Montreal, August 1995. The final version appeared in: Sloman, A and Poli R, (1996) SIM_AGENT: A toolkit for exploring agent designs, in **Intelligent Agents Vol II (ATAL-95)** , Eds. Mike Wooldridge, Joerg Mueller, Milind Tambe, Springer-Verlag pp 392--407

11. One of the early examples of the use of the toolkit: the "Blind and Lazy" robot scenario, an experiment in co-evolution, by Riccardo Poli (now at Essex University). See
Aaron Sloman and Riccardo Poli, (1996)
SIM_AGENT: A toolkit for exploring agent designs, in *Intelligent Agents Vol II (ATAL-95)*, Eds. M. Wooldridge and J. Mueller and M. Tambe, Springer-Verlag, pp. 392--407, (PDF)

The program code, including a great deal of online documentation is available in the FreePoplog ftp directory, at the University of Birmingham, both in browsable directories and in compressed tar files. See below for more information.

Tutorial file

A tutorial introduction to the SIM_AGENT toolkit is provided in the SIM_FEELINGS teach file, which is part of the online documentation included in the package. This file is an executable tutorial demonstration of the use of the toolkit in a simple scenario involving (very shallow) "emotional" agents with a simple architecture, moving around in a 2-D world containing other immobile objects.

It is a plain text file, designed for viewing in the Poplog editor VED. It includes code that can be copied and run, if you have a Poplog system and have fetched the toolkit from the Birmingham Poplog ftp site.

A different tutorial, the the TEACH SIM_DEMO code produced some of the demonstration "movies", described below.

THE SIM_AGENT TOOLKIT HAS THREE MAIN COMPONENTS

The components are the rule interpreter, the sim_agent library and the rclib graphical package. The first and third can be used independently of sim_agent.

1. Poprulebase:
a very general, flexible and extendable interpreter for condition-action rules, written in Pop-11. It is based on the idea of a forward-chaining production system interpreter, but provides a collection of unusual facilities, including a smooth interface to neural net or other "sub-symbolic" mechanisms, through so-called "filter" conditions, a wide variety of condition types and action types (both user-extendable), and a variety of control options and higher level structuring facilities.

Poprulebase allows rulesets to be combined into rulefamilies. Within a rulefamily, control can be transferred between rulesystems as the context changes. This also allows SOAR-like pushing and popping of contexts, as well as other sorts of transitions. Rulefamilies can be combined into rulesystems. Each agent has a rulesystem, which may be an arbitrarily complex collection of interacting rulesets and rulefamilies, with associated databases providing memory stores and communication channels.

2. The Sim_agent library
provides a set of base classes and scheduling mechanisms for running simulations involving a number of objects and agents whose internal mechanisms are implemented as multiple rulesets or rulefamilies, using Poprulebase. Sim_agent also makes use of the object oriented facilities provided in the Pop-11 Objectclass package, a CLOS-like extension to Pop-11 providing multiple-inheritance, generic functions, etc. Objectclass, which was designed and implemented by Steve Knight at HP Research Labs, Bristol, turns Pop-11 into a powerful language for designing re-usable extendable software modules, as CLOS does for Lisp.

In sim_agent, each agent has its own rulesystem which may consist of a collection of rulesets and rulefamilies implementing a variety of mechanisms (perception, memory, motive generation, planning, self-monitoring, etc.). A feature of the toolkit is provision for differing resource allocation between components of an agent, and between agents. E.g. some agents may plan quickly relative to their perceptual processes, others slowly.

3. The RCLIB package: This extends the Pop-11 RC_GRAPHIC facility and provides flexible new tools for building graphical demonstrations and complex graphical control panels.

Examples of displays produced by the "RCLIB" Graphic Library can be found in
http://www.cs.bham.ac.uk/research/poplog/figs/rclib/

MPEG MOVIES PRODUCED BY TOOLKIT USERS

A collection of Mpeg movie demonstrations showing some of the features of SimAgent can be found here (with thanks to Mike Lees at Nottingham University for the Boids and Tileworld examples):
http://www.cs.bham.ac.uk/research/poplog/figs/simagent/

RUNNING THE TOOLKIT

The packages can be run only in a Poplog Pop-11 environment, though anyone who has Poplog may freely copy and use the code (but please let me know if you do). It requires Poplog Version 15.0 or later.

For more information on Pop-11 and Poplog see:
http://www.cs.bham.ac.uk/research/poplog/primer
http://www.cogs.susx.ac.uk/users/adrianh/pop11.html
http://www.cogs.susx.ac.uk/users/adrianh/poplog.html
http://www.cs.bham.ac.uk/research/poplog/poplog.info.html

Downloadable versions of Poplog and the toolkit

are available here: http://www.cs.bham.ac.uk/research/poplog/freepoplog.html

A user of linux on a PC (x86 or x86_64) can fetch everything required from here.

CODE AND DOCUMENTATION

In order to run SimAgent you need to have poplog, various extensions to Poplog produced in Birmingham (e.g. extending the Pop11 pattern matcher), the Poprulebase package and the SimAgent package. Use of the graphical features of SimAgent and the ability to run the graphical demos also requires the RCLIB package. The simplest way to get everything if you are using a PC running linux is to download the whole Birmingham Linux poplog tar file mentioned above and described in

http://www.cs.bham.ac.uk/research/poplog/freepoplog.html#pclinuxversions The toolkit is now included in the linux poplog package by default, in the $usepop/pop/packages/newkit library.

The following sub-packages are available for download separately.

1. Poprulebase
Code and documentation for the Poprulebase package can be found in a browsable directory..

The main documentation files for Poprulebase (in plain ascii text) are in the PRB/HELP sub-directory.

The tutorial documentation (in plain ascii text) is in the PRB/TEACH sub-directory.

There is also a compressed tar file for Poprulebase.

Most of poprulebase can be used without SimAgent or RCLIB.

2. Sim_agent
Code and documentation for SIM_AGENT is in a browsable directory, and also in a compressed tar file. This cannot be used without Poprulebase. Some of the examples with graphical interfaces also require RCLIB.

The main overview documentation for SIM_AGENT (in plain ascii text) is in the SIM/HELP sub-directory.

The tutorial documentation (in plain ascii text) is in the SIM/TEACH sub-directory.

3. The RCLIB graphical library
The graphical libraries used in some of the demonstration code can be found in the "rclib" extension to Pop-11's rc_graphic library, which provides a sophisticated object oriented graphical extension to Pop-11 including 2-D drawing, movable objects, buttons, sliders (not just horizontal and vertical) etc.

Code and documentation for RCLIB can be found in a browsable directory, and also in a compressed tar file.

The main overview documentation for RCLIB (in plain ascii text) is in the RCLIB/HELP sub-directory, especially the HELP RCLIB file.

The tutorial documentation for RCLIB (in plain ascii text) is in the RCLIB/TEACH sub-directory, especially the introductory overview tutorial.

ACKNOWLEDGEMENTS

The bulk of the design and implementation was done by Aaron Sloman who is also responsible for any major design flaws. However, many individuals contributed ideas, code, demonstrations, etc., including Luc Beaudoin and Tim Read who suffered from the lack of something like this when they were doing their PhDs and (roughly in chronological order) Riccardo Poli, Jeremy Baxter, Richard Hepplewhite, Darryl Davis, Ian Wright, Brian Logan, Peter Waudby, Tom Carter, Catriona Kennedy, and other users at the University of Birmingham, where it is used for undergraduate and MSc student projects as well as for research.

The Pop-11 Objectclass system was crucial for the development of both RCLIB and the SIM_AGENT mechanisms. Objectclass provides multiple inheritance and generic functions, extending Pop-11 roughly as CLOS extends Common Lisp. It was originally designed and implemented by Steve Leach

when he was at Hewlett Packard research labs. He is now a freelance software engineer. He made many other contributions to the design as a user, while at Hewlett Packard, some of which are part of poplog, others in libraries at the Poplog web site.

Objectclass makes it easy to introduce new types of agents sharing some default methods with the top level agent class while providing their own extensions. This was also a requirement for the graphical tools. Objectclass also made it easy to combine agent classes with graphical object classes to produce agents which both act in a simulated world and also display their activity in a 2-D graphical display, where they can be manipulated with the mouse.

Code and documentation for objectclass are part of the Poplog system.

Thanks to Chris Glur for pointing out typos.

FUNDING

Some of the early work on the toolkit was funded by a grant from the UK Joint Research Council Initiative on HCI and Cognitive Science (1992-5)

Additional funding came from the University of Birmingham (1994-5).

Our collaborators at DERA provided funding which partly contributed to development of the toolkit between 1994 and 1998.

Disclaimer

Poprulebase and Sim_agent form part of the Cognition and Affect Project.

The whole package continues to improve (partly on the basis of suggestions from users), while every effort is made to retain ‘backward compatibility’. Comments and criticisms are welcome and may be addressed to me.

Aaron Sloman
School of Computer Science, The University of Birmingham,
Birmingham, B15 2TT,
England

EMAIL: A.Sloman AT cs.bham.ac.uk

This file is maintained by:
Aaron Sloman.
Last updated: 30 May 2005; 27 Jan 2010; 31 Aug 2014 (reformat, and some minor corrections and additions.)