GNU/Linux AI & Alife HOWTO

Table of Contents

GNU/Linux AI & Alife HOWTO

by John Eikenberry

v1.5, 08 Dec 2000

This howto mainly contains information about, and links to, various AI related software libraries, applications, etc. that work on the GNU/Linux platform. All of it is (at least) free for personal use. The new master page for this document is http://zhar.net/gnu−linux/howto/

1. Introduction

2. Traditional Artificial Intelligence

3. Connectionism

4. Evolutionary Computing

5. Alife & Complex Systems

6. Agents

7. Programming languages

1. Introduction

1.1 Purpose

The GNU/Linux OS has evolved from its origins in hackerdom to a full blown UNIX, capable of rivaling any commercial UNIX. It now provides an inexpensive base to build a great workstation. It has shed its hardware dependencies, having been ported to DEC Alphas, Sparcs, PowerPCs, and many others. This potential speed boost along with its networking support will make it great for workstation clusters. As a workstation it allows for all sorts of research and development, including artificial intelligence and artificial life.

The purpose of this Mini−Howto is to provide a source to find out about various software packages, code libraries, and anything else that will help someone get started working with (and find resources for) artificial intelligence, artificial life, etc. All done with GNU/Linux specifically in mind.

1.2 Where to find this software

All this software should be available via the net (ftp || http). The links to where to find it will be provided in the description of each package. There will also be plenty of software not covered on these pages (which is usually platform independent) located on one of the resources listed on the links section of the Master Site (given above).

1.3 Updates and comments

If you find any mistakes, know of updates to one of the items below, or have problems compiling any of the applications, please mail me at: jae@NOSPAM−zhar.net and I'll see what I can do.

If you know of any AI/Alife applications, class libraries, etc. **Please** email me about them. Include your name, ftp and/or http sites where they can be found, plus a brief overview/commentary on the software (this info would make things a lot easier on me... but don't feel obligated ;).

I know that keeping this list up to date and expanding it will take quite a bit of work. So please be patient (I do have other projects). I hope you will find this document helpful.

1.4 Copyright/License

---

2. Traditional Artificial Intelligence

Traditional AI is based around the ideas of logic, rule systems, linguistics, and the concept of rationality. At its roots are programming languages such as Lisp and Prolog. Expert systems are the largest successful example of this paradigm. An expert system consists of a detailed knowledge base and a complex rule system to utilize it. Such systems have been used for such things as medical diagnosis support and credit checking systems.

2.1 AI class/code libraries

These are libraries of code or classes for use in programming within the artificial intelligence field. They are not meant as stand alone applications, but rather as tools for building your own applications.

ACL2

- Web site: www.telent.net/cliki/ACL2

ACL2 (A Computational Logic for Applicative Common Lisp) is a theorem prover for industrial applications. It is both a mathematical logic and a system of tools for constructing proofs in the logic. ACL2 works with GCL (GNU Common Lisp).

AI Search II

- WEB site: www.bell−labs.com/topic/books/ooai−book/

Submitted by: Peter M. Bouthoorn

Basically, the library offers the programmer a set of search algorithms that may be used to solve all kind of different problems. The idea is that when developing problem solving software the programmer should be able to concentrate on the representation of the problem to be solved and should not need to bother with the implementation of the search algorithm that will be used to actually conduct the search. This idea has been realized by the implementation of a set of search classes that may be incorporated in other software through **C++**'s features of derivation and inheritance. The following search algorithms have been implemented:

− depth−first tree and graph search. − breadth−first tree and graph search. − uniform−cost tree and graph search. − best−first search. − bidirectional depth−first tree and graph search. − bidirectional breadth−first tree and graph search. − AND/OR depth tree search. − AND/OR breadth tree search.

This library has a corresponding book, " Object−Oriented Artificial Instelligence, Using C++".

Chess In Lisp (CIL)

- FTP site: chess.onenet.net/pub/chess/uploads/projects/

The CIL (Chess In Lisp) foundation is a Common Lisp implementaion of all the core functions needed for development of chess applications. The main purpose of the CIL project is to get AI researchers interested in using Lisp to work in the chess domain.

DAI

- Web site: starship.python.net/crew/gandalf/DNET/AI/

A library for the Python programming language that provides an object oriented interface to the CLIPS expert system tool. It includes an interface to COOL (CLIPS Object Oriented Language) that allows:

- Investigate COOL classes
- Create and manipulate with COOL instances
- Manipulate with COOL message−handler's
- Manipulate with Modules

LK

- Web site: www.cs.utoronto.ca/~neto/research/lk/

LK is an implementation of the Lin−Kernighan heuristic for the Traveling Salesman Problem and for the minimum weight perfect matching problem. It is tuned for 2−d geometric instances, and has been applied to certain instances with up to a million cities. Also included are instance generators and Perl scripts for munging TSPLIB instances.

This implementation introduces ``efficient cluster compensation'', an experimental algorithmic technique intended to make the Lin−Kernighan heuristic more robust in the face of clustered data.

Nyquist

- Web site: www.cs.cmu.edu/afs/cs.cmu.edu/project/music/web/music.html

The Computer Music Project at CMU is developing computer music and interactive performance technology to enhance human musical experience and creativity. This interdisciplinary effort draws on Music Theory, Cognitive Science, Artificial Intelligence and Machine Learning, Human Computer Interaction, Real−Time Systems, Computer Graphics and Animation, Multimedia, Programming Languages, and Signal Processing. A paradigmatic example of these interdisciplinary efforts is the creation of interactive performances that couple human musical improvisation with intelligent computer agents in real−time.

PDKB

- Web site: lynx.eaze.net/~pdkb/web/
- SourceForge site: sourceforge.net/project/pdkb

Public Domain Knowledge Bank (PDKB) is an Artificial Intelligence Knowledge Bank of common sense rules and facts. It is based on the Cyc Upper Ontology and the MELD language.

Python Fuzzy Logic Module

- FTP site: ftp://ftp.csh.rit.edu/pub/members/retrev/

A simple python module for fuzzy logic. The file is 'fuz.tar.gz' in this directory. The author plans to also write a simple genetic algorithm and a neural net library as well. Check the 00_index file in this directory for release info.

QUANT1

- Web site: linux.irk.ru/projects/QUANT/

QUANT/1 stands for type QUANTifier. It aims to be an alternative to Prolog−like (Resulutional−like) systems. Main features include a lack of necessity for eliminating Quantifiers, scolemisation, ease of comprehension, large scale formulae operation, acceptance of nonHorn formulaes, and Iterative deeping. The actual library implemented in this project is called ATPPCF (Automatic Theorem Prover in calculus of Positively Constructed Formulae).

ATPPCF will be a library (inference engine) and an extension of the Predicate Calculus Language as a new logical language. The library will be incorporable in another software such as TCL, Python, Perl. The engine's primary inference method will be the "search of inference in language of Positively Constructed Formulas (PCFs)" (a subset of Predicate Calculus well translated in both directions). The language will be used as scripting language to the engine. But there will be possibility to replace it with extensions languages of main software.

Screamer

- Web site: www.cis.upenn.edu/~screamer−tools/home.html

Screamer is an extension of Common Lisp that adds support for nondeterministic programming. Screamer consists of two levels. The basic nondeterministic level adds support for backtracking and undoable side effects. On top of this nondeterministic substrate, Screamer provides a comprehensive constraint programming language in which one can formulate and solve mixed systems of numeric and symbolic constraints. Together, these two levels augment Common Lisp with practically all of the functionality of both Prolog and constraint logic programming languages such as CHiP and CLP(R). Furthermore, Screamer is fully integrated with Common Lisp. Screamer programs can coexist and interoperate with other extensions to Common Lisp such as CLOS, CLIM and Iterate.

ThoughtTreasure

- Web site: www.signiform.com/tt/htm/tt.htm

ThoughtTreasure is a project to create a database of commonsense rules for use in any application. It consists of a database of a little over 100K rules and a C API to integrate it with your applications. Python, Perl, Java and TCL wrappers are already available.

2.2 AI software kits, applications, etc.

These are various applications, software kits, etc. meant for research in the field of artificial intelligence. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

ASA − Adaptive Simulated Annealing

- Web site: www.ingber.com/#ASA−CODE
- FTP site: ftp.ingber.com/

ASA (Adaptive Simulated Annealing) is a powerful global optimization C−code algorithm especially useful for nonlinear and/or stochastic systems.

ASA is developed to statistically find the best global fit of a nonlinear non−convex cost−function over a D−dimensional space. This algorithm permits an annealing schedule for 'temperature' T decreasing exponentially in annealing−time k, T = T_0 exp(−c k^1/D). The introduction of re−annealing also permits adaptation to changing sensitivities in the multi−dimensional parameter−space. This annealing schedule is faster than fast Cauchy annealing, where T = T_0/k, and much faster than Boltzmann annealing, where T = T_0/ln k.

Babylon

- FTP site: ftp.gmd.de/gmd/ai−research/Software/Babylon/

BABYLON is a modular, configurable, hybrid environment for developing expert systems. Its features include objects, rules with forward and backward chaining, logic (Prolog) and constraints. BABYLON is implemented and embedded in Common Lisp.

cfengine

- Web site: www.iu.hioslo.no/cfengine/

Cfengine, or the configuration engine is a very high level language for building expert systems which administrate and configure large computer networks. Cfengine uses the idea of classes and a primitive form of intelligence to define and automate the configuration of large systems in the most economical way possible. Cfengine is design to be a part of computer immune systems.

CLEARS

- Web site: www.coli.uni−sb.de/~clears/

The CLEARS system is an interactive graphical environment for computational semantics. The tool allows exploration and comparison of different semantic formalisms, and their interaction with syntax. This enables the user to get an idea of the range of possibilities of semantic construction, and also where there is real convergence between theories.

CLIG

- Web site: www.ags.uni−sb.de/~konrad/clig.html

CLIG is an interactive, extendible grapher for visualizing linguistic data structures like trees, feature structures, Discourse Representation Structures (DRS), logical formulas etc. All of these can be freely mixed and embedded into each other. The grapher has been designed both to be stand−alone and to be used as an add−on for linguistic applications which display their output in a graphical manner.

CLIPS

- Web site: www.jsc.nasa.gov/~clips/CLIPS.html
- FTP site: cs.cmu.edu/afs/cs.cmu.edu/project/ai−repository/ai/areas/expert/systems/clips

CLIPS is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object based expert systems.

CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule−based, object−oriented and procedural. Rule−based programming allows knowledge to be represented as heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation. Object−oriented programming allows complex systems to be modeled as modular components (which can be easily reused to model other systems or to create new components). The procedural programming capabilities provided by CLIPS are similar to capabilities found in languages such as C, Pascal, Ada, and LISP.

EMA−XPS − A Hybrid Graphic Expert System Shell

- Web site: wmwap1.math.uni−wuppertal.de:80/EMA−XPS/

EMA−XPS is a hybrid graphic expert system shell based on the ASCII−oriented shell Babylon 2.3 of the German National Research Center for Computer Sciences (GMD). In addition to Babylon's AI−power (object oriented data representation, forward and backward chained rules − collectible into sets, horn clauses, and constraint networks) a graphic interface based on the X11 Window System and the OSF/Motif Widget Library has been provided.

FOOL & FOX

- FTP site: ntia.its.bldrdoc.gov/pub/fuzzy/prog/

FOOL stands for the Fuzzy Organizer OLdenburg. It is a result from a project at the University of Oldenburg. FOOL is a graphical user interface to develop fuzzy rulebases. FOOL will help you to invent and maintain a database that specifies the behavior of a fuzzy−controller or something like that.

FOX is a small but powerful fuzzy engine which reads this database, reads some input values and calculates the new control value.

FUF and SURGE

- Web site: www.dfki.de/lt/registry/generation/fuf.html
- FTP site: ftp.cs.columbia.edu/pub/fuf/

FUF is an extended implementation of the formalism of functional unification grammars (FUGs) introduced by Martin Kay specialized to the task of natural language generation. It adds the following features to the base formalism:

- Types and inheritance.
- Extended control facilities (goal freezing, intelligent backtracking).
- Modular syntax.

These extensions allow the development of large grammars which can be processed efficiently and can be maintained and understood more easily. SURGE is a large syntactic realization grammar of English written in FUF. SURGE is developed to serve as a black box syntactic generation component in a larger generation system that encapsulates a rich knowledge of English syntax. SURGE can also be used as a platform for exploration of grammar writing with a generation perspective.

The Grammar Workbench

- Web site: www.cs.kun.nl/agfl/GWB.html

The Grammar Workbench, or GWB for short, is an environment for the comfortable development of Affix Grammars in the AGFL−formalism. Its purposes are:

- to allow the user to input, inspect and modify a grammar;

- to perform consistency checks on the grammar;
- to compute grammar properties;
- to generate example sentences;
- to assist in performing grammar transformations.

GSM Suite

- Web site: www.slip.net/~andrewm/gsm/

The GSM Suite is a set of programs for using Finite State Machines in a graphical fashion. The suite consists of programs that edit, compile, and print state machines. Included in the suite is an editor program, gsmedit, a compiler, gsm2cc, that produces a C++ implementation of a state machine, a PostScript generator, gsm2ps, and two other minor programs. GSM is licensed under the GNU Public License and so is free for your use under the terms of that license.

Illuminator

- Web site: documents.cfar.umd.edu/resources/source/illuminator.html

Illuminator is a toolset for developing OCR and Image Understanding applications. Illuminator has two major parts: a library for representing, storing and retrieving OCR information, heretofore called dafslib, and an X−Windows "DAFS" file viewer, called illum. Illuminator and DAFS lib were designed to supplant existing OCR formats and become a standard in the industry. They particularly are extensible to handle more than just English.

The features of this release:

- 5 magnification levels for images
- flagged characters and words
- unicode support −− American, British, French, German, Greek, Italian, MICR, Norwegian, Russian, Spanish, Swedish, keyboards
- reads DAFS, TIFF's, PDA's (image only)
- save to DAFS, ASCII/UTF or Unicode
- Entity Viewer − shows properties, character choices, bounding boxes image fragment for a selected entity, change type, change content, hierarchy mode

Isabelle

- Web site: isabelle.in.tum.de

Isabelle is a popular generic theorem prover developed at Cambridge University and TU Munich. Existing logics like Isabelle/HOL provide a theorem proving environment ready to use for sizable applications. Isabelle may also serve as framework for rapid prototyping of deductive systems. It comes with a large library including Isabelle/HOL (classical higher−order logic), Isabelle/HOLCF (Scott's Logic for Computable Functions with HOL), Isabelle/FOL (classical and intuitionistic first−order logic), and Isabelle/ZF (Zermelo−Fraenkel set theory on top of FOL).

Jess, the Java Expert System Shell

- Web site: herzberg.ca.sandia.gov/jess/

Jess is a clone of the popular CLIPS expert system shell written entirely in Java. With Jess, you can conveniently give your applets the ability to 'reason'. Jess is compatible with all versions of Java starting with version 1.0.2. Jess implements the following constructs from CLIPS: defrules, deffunctions, defglobals, deffacts, and deftemplates.

learn

- FTP site: sunsite.unc.edu/pub/Linux/apps/cai/

Learn is a vocable learning program with memory model.

NICOLE

- Web site: nicole.sourceforge.net

NICOLE (Nearly Intelligent Computer Operated Language Examiner) is a theory or experiment that if a computer is given enough combinations of how words, phrases and sentences are related to one another, it could talk back to you. It is an attempt to simulate a conversation by learning how words are related to other words. A human communicates with NICOLE via the keyboard and NICOLE responds back with its own sentences which are automatically generated, based on what NICOLE has stored in it's database. Each new sentence that has been typed in, and NICOLE doesn't know about, is included into NICOLE's database, thus extending the knowledge base of NICOLE.

Otter: An Automated Deduction System

- Web site: www−unix.mcs.anl.gov/AR/otter/

Our current automated deduction system Otter is designed to prove theorems stated in first−order logic with equality. Otter's inference rules are based on resolution and paramodulation, and it includes facilities for term rewriting, term orderings, Knuth−Bendix completion, weighting, and strategies for directing and restricting searches for proofs. Otter can also be used as a symbolic calculator and has an embedded equational programming system.

NICOLE

- Web site: nicole.sourceforge.net

It is an attempt to simulate a conversation by learning how words are related to other words. A Human communicates with NICOLE via the keyboard and NICOLE responds back with its own sentences which are automatically generated, based on what NICOLE has stored in it's database. Each new sentence that has been typed in, and NICOLE doesn't know about it, it is included into NICOLE's database, thus extending the knowledge base of NICOLE.

PVS

- Web site: pvs.csl.sri.com/

PVS is a verification system: that is, a specification language integrated with support tools and a theorem prover. It is intended to capture the state−of−the−art in mechanized formal methods and to be sufficiently rugged that it can be used for significant applications. PVS is a research prototype: it evolves and improves as we develop or apply new capabilities, and as the stress of real use exposes new requirements.

RIPPER

- Web site: www.research.att.com/~wcohen/ripperd.html

Ripper is a system for fast effective rule induction. Given a set of data, Ripper will learn a set of rules that will predict the patterns in the data. Ripper is written in ASCI C and comes with documentation and some sample problems.

SNePS

- Web site: www.cs.buffalo.edu/pub/sneps/WWW/
- FTP site: ftp.cs.buffalo.edu/pub/sneps/

The long−term goal of The SNePS Research Group is the design and construction of a natural−language−using computerized cognitive agent, and carrying out the research in artificial intelligence, computational linguistics, and cognitive science necessary for that endeavor. The three−part focus of the group is on knowledge representation, reasoning, and natural−language understanding and generation. The group is widely known for its development of the SNePS knowledge representation/reasoning system, and Cassie, its computerized cognitive agent.

Soar

- Web site: bigfoot.eecs.umich.edu/~soar/
- FTP site: cs.cmu.edu/afs/cs/project/soar/public/Soar6/

Soar has been developed to be a general cognitive architecture. We intend ultimately to enable the Soar architecture to:

- work on the full range of tasks expected of an intelligent agent, from highly routine to extremely difficult, open−ended problems
- represent and use appropriate forms of knowledge, such as procedural, declarative, episodic, and possibly iconic
- employ the full range of problem solving methods
- interact with the outside world and
- learn about all aspects of the tasks and its performance on them.

In other words, our intention is for Soar to support all the capabilities required of a general intelligent agent. http://wwwis.cs.utwente.nl:8080/ tcm/index.html

TCM

- Web site: wwwis.cs.utwente.nl:8080/~tcm/index.html
- FTP site: ftp.cs.vu.nl/pub/tcm/

TCM (Toolkit for Conceptual Modeling) is our suite of graphical editors. TCM contains graphical editors for Entity−Relationship diagrams, Class−Relationship diagrams, Data and Event Flow diagrams, State Transition diagrams, Jackson Process Structure diagrams and System Network diagrams, Function Refinement trees and various table editors, such as a Function−Entity table editor and a Function Decomposition table editor. TCM is easy to use and performs numerous consistency checks, some of them immediately, some of them upon request.

WEKA

- Web site: lucy.cs.waikato.ac.nz/~ml/

WEKA (Waikato Environment for Knowledge Analysis) is an state−of−the−art facility for applying machine learning techniques to practical problems. It is a comprehensive software "workbench" that allows people to analyse real−world data. It integrates different machine learning tools within a common framework and a uniform user interface. It is designed to support a "simplicity−first" methodology, which allows users to experiment interactively with simple machine learning tools before looking for more complex solutions.

---

3. Connectionism

Connectionism is a technical term for a group of related techniques. These techniques include areas such as Artificial Neural Networks, Semantic Networks and a few other similar ideas. My present focus is on neural networks (though I am looking for resources on the other techniques). Neural networks are programs designed to simulate the workings of the brain. They consist of a network of small mathematical−based nodes, which work together to form patterns of information. They have tremendous potential and currently seem to be having a great deal of success with image processing and robot control.

3.1 Connectionist class/code libraries

These are libraries of code or classes for use in programming within the Connectionist field. They are not meant as stand alone applications, but rather as tools for building your own applications.

ANSI−C Neural Networks

- Web site: www.geocities.com/CapeCanaveral/1624/

This site contains ANSC−C source code for 8 types of neural nets, including:

- Adaline Network
- Backpropagation
- Hopfield Model

- (BAM) Bidirectional Associative Memory
- Boltzmann Machine
- Counterpropagation
- (SOM) Self−Organizing Map
- (ART1) Adaptive Resonance Theory

They were designed to help turn the theory of a particular network model into the design for a simulator implementation , and to help with embeding an actual application into a particular network model.

BELIEF

- Web site: www.cs.cmu.edu/afs/cs/project/ai−repository/ai/areas/reasonng/probabl/belief/

BELIEF is a Common Lisp implementation of the Dempster and Kong fusion and propagation algorithm for Graphical Belief Function Models and the Lauritzen and Spiegelhalter algorithm for Graphical Probabilistic Models. It includes code for manipulating graphical belief models such as Bayes Nets and Relevance Diagrams (a subset of Influence Diagrams) using both belief functions and probabilities as basic representations of uncertainty. It uses the Shenoy and Shafer version of the algorithm, so one of its unique features is that it supports both probability distributions and belief functions. It also has limited support for second order models (probability distributions on parameters).

bpnn.py

- Web site: www.enme.ucalgary.ca/~nascheme/python/

A simple back−propogation ANN in Python.

CONICAL

- Web site: strout.net/conical/

CONICAL is a C++ class library for building simulations common in computational neuroscience. Currently its focus is on compartmental modeling, with capabilities similar to GENESIS and NEURON. A model neuron is built out of compartments, usually with a cylindrical shape. When small enough, these open−ended cylinders can approximate nearly any geometry. Future classes may support reaction−diffusion kinetics and more. A key feature of CONICAL is its cross−platform compatibility; it has been fully co−developed and tested under Unix, DOS, and Mac OS.

IDEAL

- Web site: www.rpal.rockwell.com/ideal.html

IDEAL is a test bed for work in influence diagrams and Bayesian networks. It contains various inference algorithms for belief networks and evaluation algorithms for influence diagrams. It contains facilities for creating and editing influence diagrams and belief networks.

IDEAL is written in pure Common Lisp and so it will run in Common Lisp on any platform. The emphasis in writing IDEAL has been on code clarity and providing high level programming abstractions. It thus is very suitable for experimental implementations which need or extend belief network technology.

At the highest level, IDEAL can be used as a subroutine library which provides belief network inference and influence diagram evaluation as a package. The code is documented in a detailed manual and so it is also possible to work at a lower level on extensions of belief network methods.

IDEAL comes with an optional graphic interface written in CLIM. If your Common Lisp also has CLIM, you can run the graphic interface.

Jet's Neural Architecture

- Web site: www.voltar−confed.org/jneural/

Jet's Neural Architecture is a C++ framework for doing neural net projects. The goals of this project were to make a fast, flexible neural architecture that isn't stuck to one kind of net and to make sure that end users could easily write useful applications. All the documentation is also easily readable.

Matrix Class

- FTP site: ftp.cs.ucla.edu/pub/

A simple, fast, efficient C++ Matrix class designed for scientists and engineers. The Matrix class is well suited for applications with complex math algorithms. As an demonstration of the Matrix class, it was used to implement the backward error propagation algorithm for a multi−layer feed−forward artificial neural network.

nunu

- Web site: ruby.ddiworld.com/jreed/web/software/nn.html

nunu is a multi−layered, scriptable, back−propagation neural network. It is build to be used for intensive computation problems scripted in shell scripts. It is written in C++ using the STL. nn is based on material from the "Introduction to the Theory of Neural Computation" by John Hertz, Anders Krogh, and Richard G. Palmer, chapter 6.

Pulcinella

- Web site: iridia.ulb.ac.be/pulcinella/Welcome.html

Pulcinella is written in CommonLisp, and appears as a library of Lisp functions for creating, modifying and evaluating valuation systems. Alternatively, the user can choose to interact with Pulcinella via a graphical interface (only available in Allegro CL). Pulcinella provides primitives to build and evaluate uncertainty models according to several uncertainty calculi, including probability theory, possibility theory, and Dempster−Shafer's theory of belief functions; and the possibility theory by Zadeh, Dubois and Prade's. A User's Manual is available on request.

S−ElimBel

- Web site (???): www.spaces.uci.edu/thiery/elimbel/

S−ElimBel is an algorithm that computes the belief in a Bayesian network, implemented in MIT−Scheme. This algorithm has the particularity of being rather easy to understand. Moreover, one can apply it to any kind of Bayesian network − it being singly connected or muliply connected. It is, however, less powerful than the standard algorithm of belief propagation. Indeed, the computation has to be reconducted entirely for each new evidence added to the network. Also, one needs to run the algorithm as many times as one has nodes for which the belief is wanted.

Software for Flexible Bayesian Modeling

- Web site: www.cs.utoronto.ca/~radford/fbm.software.html

This software implements flexible Bayesian models for regression and classification applications that are based on multilayer perceptron neural networks or on Gaussian processes. The implementation uses Markov chain Monte Carlo methods. Software modules that support Markov chain sampling are included in the distribution, and may be useful in other applications.

Spiderweb2

- Web site: www.cs.nyu.edu/~klap7794/spiderweb2.html

A C++ artificial neual net library. Spiderweb2 is a complete rewrite of the original Spiderweb library, it has grown into a much more flexible and object−oriented system. The biggest change is that each neuron object is responsible for its own activations and updates, with the network providing only the scheduling aspect. This is a very powerful change, and it allows easy modification and experimentation with various network architectures and neuron types.

Symbolic Probabilistic Inference (SPI)

- FTP site: ftp.engr.orst.edu/pub/dambrosi/spi/
- Paper (ijar−94.ps): ftp.engr.orst.edu/pub/dambrosi/

Contains Common Lisp function libraries to implement SPI type baysean nets. Documentation is very limited. Features:

- Probabilities, Local Expression Language Utilities, Explanation, Dynamic Models, and a TCL/TK based GUI.

TresBel

- FTP site: iridia.ulb.ac.be/pub/hongxu/software/

Libraries containing (Allegro) Common Lisp code for Belief Functions (aka. Dempster−Shafer evidential reasoning) as a representation of uncertainty. Very little documentation. Has a limited GUI.

Various (C++) Neural Networks

- Web site: www.dontveter.com/nnsoft/nnsoft.html

Example neural net codes from the book, The Pattern Recognition Basics of AI. These are simple example codes of these various neural nets. They work well as a good starting point for simple experimentation and for learning what the code is like behind the simulators. The types of networks available on this site are: (implemented in C++)

- The Backprop Package
- The Nearest Neighbor Algorithms
- The Interactive Activation Algorithm
- The Hopfield and Boltzman machine Algorithms
- The Linear Pattern Classifier
- ART I
- Bi−Directional Associative Memory
- The Feedforward Counter−Propagation Network

3.2 Connectionist software kits/applications

These are various applications, software kits, etc. meant for research in the field of Connectionism. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

Aspirin − MIGRAINES

(am6.tar.Z on ftp site)

- FTP site: sunsite.unc.edu/pub/academic/computer−science/neural−networks/programs/Aspirin/

The software that we are releasing now is for creating, and evaluating, feed−forward networks such as those used with the backpropagation learning algorithm. The software is aimed both at the expert programmer/neural network researcher who may wish to tailor significant portions of the system to his/her precise needs, as well as at casual users who will wish to use the system with an absolute minimum of effort.

DDLab

- Web site: www.santafe.edu/~wuensch/ddlab.html
- FTP site: ftp.santafe.edu/pub/wuensch/

DDLab is an interactive graphics program for research into the dynamics of finite binary networks, relevant to the study of complexity, emergent phenomena, neural networks, and aspects of theoretical biology such as gene regulatory networks. A network can be set up with any architecture between regular CA (1d or 2d) and "random Boolean networks" (networks with arbitrary connections and heterogeneous rules). The network may also have heterogeneous neighborhood sizes.

GENESIS

- Web site: www.bbb.caltech.edu/GENESIS/
- FTP site: genesis.bbb.caltech.edu/pub/genesis/

GENESIS (short for GEneral NEural SImulation System) is a general purpose simulation platform which was developed to support the simulation of neural systems ranging from complex models of single neurons to simulations of large networks made up of more abstract neuronal components. GENESIS has provided the basis for laboratory courses in neural simulation at both Caltech and the Marine Biological Laboratory in Woods Hole, MA, as well as several other institutions. Most current GENESIS applications involve realistic simulations of biological neural systems. Although the software can also model more abstract networks, other simulators are more suitable for backpropagation and similar connectionist modeling.

JavaBayes

- Web site: www.cs.cmu.edu/People/javabayes/index.html/

The JavaBayes system is a set of tools, containing a graphical editor, a core inference engine and a parser. JavaBayes can produce:

- the marginal distribution for any variable in a network.
- the expectations for univariate functions (for example, expected value for variables).
- configurations with maximum a posteriori probability.
- configurations with maximum a posteriori expectation for univariate functions.

Jbpe

- Web site: cs.felk.cvut.cz/~koutnij/studium/jbpe.html

Jbpe is a back−propagation neural network editor/simulator.

Features

- Standart back−propagation networks creation.
- Saving network as a text file, which can be edited and loaded back.
- Saving/loading binary file
- Learning from a text file (with structure specified below), number of learning periods / desired network energy can be specified as a criterion.
- Network recall

neuralnets

- Web site: members.home.net/neuralnets/neuralnets/

neuralnets is a text−based program that allows someone to build, configure, train and run a Neural Network application. The code is written in Java and is easily extended or included within other code. The application comes ready to go with a Back Prop algorithm included. Well known applications...stock market, weather prediction, scheduling, image recognition, expert systems, research...basically anywhere you may need to make a complicated decision.

Neural Network Generator

- Web site: www.idsia.ch/~rafal/research.html
- FTP site: ftp.idsia.ch/pub/rafal

The Neural Network Generator is a genetic algorithm for the topological optimization of feedforward neural networks. It implements the Semantic Changing Genetic Algorithm and the Unit−Cluster Model. The Semantic Changing Genetic Algorithm is an extended genetic algorithm that allows fast dynamic adaptation of the genetic coding through population analysis. The Unit−Cluster Model is an approach to the construction of modular feedforward networks with a "backbone" structure.

NOTE: To compile this on Linux requires one change in the Makefiles. You will need to change '−ltermlib' to '−ltermcap'.

Neureka ANS (nn/xnn)

- Web site: www.bgif.no/neureka/
- FTP site: ftp.ii.uib.no/pub/neureka/

nn is a high−level neural network specification language. The current version is best suited for feed−forward nets, but recurrent models can and have been implemented, e.g. Hopfield nets, Jordan/Elman nets, etc. In nn, it is easy to change network dynamics. The nn compiler can generate C code or executable programs (so there must be a C compiler available), with a powerful command line interface (but everything may also be controlled via the graphical interface, xnn). It is possible for the user to write C routines that can be called from inside the nn specification, and to use the nn specification as a function that is called from a C program. Please note that no programming is necessary in order to use the network models that come with the system (`netpack').

xnn is a graphical front end to networks generated by the nn compiler, and to the compiler itself. The xnn graphical interface is intuitive and easy to use for beginners, yet powerful, with many possibilities for visualizing network data.

NOTE: You have to run the install program that comes with this to get the license key installed. It gets put (by default) in /usr/lib. If you (like myself) want to install the package somewhere other than in the /usr directory structure (the install program gives you this option) you will have to set up some environmental variables (NNLIBDIR & NNINCLUDEDIR are required). You can read about these (and a few other optional variables) in appendix A of the documentation (pg 113).

NEURON

- Web site: www.neuron.yale.edu/neuron.html
- FTP site: ftp.neuron.yale.edu/neuron/unix/

NEURON is an extensible nerve modeling and simulation program. It allows you to create complex nerve models by connecting multiple one−dimensional sections together to form arbitrary cell morphologies, and allows you to insert multiple membrane properties into these sections (including channels, synapses, ionic concentrations, and counters). The interface was designed to present the neural modeler with a intuitive environment and hide the details of the numerical methods used in the simulation.

PDP++

- Web site: www.cnbc.cmu.edu/PDP++/
- FTP site (US): cnbc.cmu.edu/pub/pdp++/
- FTP site (Europe): unix.hensa.ac.uk/mirrors/pdp++/

As the field of Connectionist modeling has grown, so has the need for a comprehensive simulation environment for the development and testing of Connectionist models. Our goal in developing PDP++ has been to integrate several powerful software development and user interface tools into a general purpose simulation environment that is both user friendly and user extensible. The simulator is built in the C++ programming language, and incorporates a state of the art script interpreter with the full expressive power of C++. The graphical user interface is built with the Interviews toolkit, and allows full access to the data structures and processing modules out of which the simulator is built. We have constructed several useful graphical modules for easy interaction with the structure and the contents of neural networks, and we've made it possible to change and adapt many things. At the programming level, we have set things up in such a way as to make user extensions as painless as possible. The programmer creates new C++ objects, which might be new kinds of units or new kinds of processes; once compiled and linked into the simulator, these new objects can then be accessed and used like any other.

RNS

- Web site: www.cs.cmu.edu/afs/cs/project/ai−repository/ai/areas/neural/systems/rns/

RNS (Recurrent Network Simulator) is a simulator for recurrent neural networks. Regular neural networks are also supported. The program uses a derivative of the back−propagation algorithm, but also includes other (not that well tested) algorithms.

Features include

- freely choosable connections, no restrictions besides memory or CPU constraints
- delayed links for recurrent networks
- fixed values or thresholds can be specified for weights
- (recurrent) back−propagation, Hebb, differential Hebb, simulated annealing and more
- patterns can be specified with bits, floats, characters, numbers, and random bit patterns with Hamming distances can be chosen for you
- user definable error functions
- output results can be used without modification as input

Simple Neural Net (in Python)

- Web site: starship.python.net/crew/amk/unmaintained/

Simple neural network code, which implements a class for 3−level networks (input, hidden, and output layers). The only learning rule implemented is simple backpropagation. No documentation (or even comments) at all, because this is simply code that I use to experiment with. Includes modules containing sample datasets from Carl G. Looney's NN book. Requires the Numeric extensions.

SCNN

- Web site: apx00.physik.uni−frankfurt.de/e_ag_rt/SCNN/

SCNN is an universal simulating system for Cellular Neural Networks (CNN). CNN are analog processing neural networks with regular and local interconnections, governed by a set of nonlinear ordinary differential equations. Due to their local connectivity, CNN are realized as VLSI chips, which operates at very high speed.

Semantic Networks in Python

- Web site: strout.net/info/coding/python/ai/index.html

The semnet.py module defines several simple classes for building and using semantic networks. A semantic network is a way of representing knowledge, and it enables the program to do simple reasoning with very little effort on the part of the programmer.

The following classes are defined:

- **Entity**: This class represents a noun; it is something which can be related to other things, and about which you can store facts.
- **Relation**: A Relation is a type of relationship which may exist between two entities. One special relation, "IS_A", is predefined because it has special meaning (a sort of logical inheritance).
- **Fact**: A Fact is an assertion that a relationship exists between two entities.

With these three object types, you can very quickly define knowledge about a set of objects, and query them for logical conclusions.

SNNS

- Web site: www.informatik.uni−stuttgart.de/ipvr/bv/projekte/snns/
- FTP site: ftp.informatik.uni−stuttgart.de/pub/SNNS/

Stuttgart Neural Net Simulator (version 4.1). An awesome neural net simulator. Better than any commercial simulator I've seen. The simulator kernel is written in C (it's fast!). It supports over 20 different network architectures, has 2D and 3D X−based graphical representations, the 2D GUI has an integrated network editor, and can generate a separate NN program in C. SNNS is very powerful, though a bit difficult to learn at first. To help with this it comes with example networks and tutorials for many of the architectures. ENZO, a supplementary system allows you to evolve your networks with genetic algorithms.

There is a debian package of SNNS available. So just get it (and use alien to convert it to RPM if you need to).

SPRLIB/ANNLIB

- Web site: www.ph.tn.tudelft.nl/~sprlib/

SPRLIB (Statistical Pattern Recognition Library) was developed to support the easy construction and simulation of pattern classifiers. It consist of a library of functions (written in C) that can be called from your own program. Most of the well−known classifiers are present (k−nn, Fisher, Parzen, ...), as well as error estimation and dataset generation routines.

ANNLIB (Artificial Neural Networks Library) is a neural network simulation library based on the data architecture laid down by SPRLIB. The library contains numerous functions for creating, training and testing feed−forward networks. Training algorithms include back−propagation, pseudo−Newton, Levenberg−Marquardt, conjugate gradient descent, BFGS.... Furthermore, it is possible − due to the datastructures' general applicability − to build Kohonen maps and other more exotic network architectures using the same data types.

TOOLDIAG

- Web site: www.inf.ufes.br/~thomas/www/home/tooldiag.html
- FTP site: ftp.inf.ufes.br/pub/tooldiag/

TOOLDIAG is a collection of methods for statistical pattern recognition. The main area of application is classification. The application area is limited to multidimensional continuous features, without any missing values. No symbolic features (attributes) are allowed. The program in implemented in the 'C' programming language and was tested in several computing environments.

XNBC

- Web site: www.b3e.jussieu.fr/xnbc/

XNBC v8 is a simulation tool for the neuroscientists interested in simulating biological neural networks using a user friendly tool.

XNBC is a software package for simulating biological neural networks.

Four neuron models are available, three phenomenologic models (xnbc, leaky integrator and conditional burster) and an ion−conductance based model. Inputs to the simulated neurons can be provided by experimental data stored in files, allowing the creation of `hybrid'' networks.

4. Evolutionary Computing

Evolutionary computing is actually a broad term for a vast array of programming techniques, including genetic algorithms, complex adaptive systems, evolutionary programming, etc. The main thrust of all these techniques is the idea of evolution. The idea that a program can be written that will *evolve* toward a certain goal. This goal can be anything from solving some engineering problem to winning a game.

4.1 EC class/code libraries

These are libraries of code or classes for use in programming within the evolutionary computation field. They are not meant as stand alone applications, but rather as tools for building your own applications.

daga

- Web site: GARAGe.cps.msu.edu/software/software−index.html

daga is an experimental release of a 2−level genetic algorithm compatible with the GALOPPS GA software. It is a meta−GA which dynamically evolves a population of GAs to solve a problem presented to the lower−level GAs. When multiple GAs (with different operators, parameter settings, etc.) are simultaneously applied to the same problem, the ones showing better performance have a higher probability of surviving and "breeding" to the next macro−generation (i.e., spawning new "daughter"−GAs with characteristics inherited from the parental GA or GAs. In this way, we try to encourage good problem−solving strategies to spread to the whole population of GAs.

Ease

- Web site: www.sprave.com/Ease/Ease.html

Ease − Evolutionary Algorithms Scripting Evironment − is an extension to the Tcl scripting language, providing commands to create, modify, and evaluate populations of individuals represented by real number vectors and/or bit strings.

EO

- Web site: geneura.ugr.es/~jmerelo/EO.html

EO is a templates−based, ANSI−C++ compliant evolutionary computation library. It contains classes for any kind of evolutionary computation (specially genetic algorithms) you might come up to. It is component−based, so that if you don't find the class you need in it, it is very easy to subclass existing abstract or concrete class.

FORTRAN GA

- Web site: www.staff.uiuc.edu/~carroll/ga.html

This program is a FORTRAN version of a genetic algorithm driver. This code initializes a random sample of individuals with different parameters to be optimized using the genetic algorithm approach, i.e. evolution via survival of the fittest. The selection scheme used is tournament selection with a shuffling technique for choosing random pairs for mating. The routine includes binary coding for the individuals, jump mutation, creep mutation, and the option for single−point or uniform crossover. Niching (sharing) and an option for the number of children per pair of parents has been added. More recently, an option for the use of a micro−GA has been added.

GAGS

- Web site: kal−el.ugr.es/gags.html
- FTP site: kal−el.ugr.es/GAGS/

Genetic Algorithm application generator and class library written mainly in C++. As a class library, and among other thing, GAGS includes:

- A *chromosome hierarchy* with variable length chromosomes. *Genetic operators*: 2−point crossover, uniform crossover, bit−flip mutation, transposition (gene interchange between 2 parts of the chromosome), and variable−length operators: duplication, elimination, and random addition.
- *Population level operators* include steady state, roulette wheel and tournament selection.
- *Gnuplot wrapper*: turns gnuplot into a `iostreams`−like class.
- Easy sample file loading and configuration file parsing.

As an application generator (written in `PERL`), you only need to supply it with an ANSI−C or C++ fitness function, and it creates a C++ program that uses the above library to 90% capacity, compiles it, and runs it, saving results and presenting fitness thru `gnuplot`.

GAlib: Matthew's Genetic Algorithms Library

- Web Site: lancet.mit.edu/ga/
- FTP site: lancet.mit.edu/pub/ga/
- Register GAlib at: lancet.mit.edu/ga/Register.html

GAlib contains a set of C++ genetic algorithm objects. The library includes tools for using genetic algorithms to do optimization in any C++ program using any representation and genetic operators. The documentation includes an extensive overview of how to implement a genetic algorithm as well as examples illustrating customizations to the GAlib classes.

GALOPPS

- Web site: GARAGe.cps.msu.edu/software/software−index.html
- FTP site: garage.cps.msu.edu/pub/GA/galopps/

GALOPPS is a flexible, generic GA, in 'C'. It was based upon Goldberg's Simple Genetic Algorithm (SGA) architecture, in order to make it easier for users to learn to use and extend.

GALOPPS extends the SGA capabilities several fold:

- (optional) A new Graphical User Interface, based on TCL/TK, for Unix users, allowing easy running of GALOPPS 3.2 (single or multiple subpopulations) on one or more processors. GUI writes/reads "standard" GALOPPS input and master files, and displays graphical output (during or after run) of user−selected variables.

- 5 selection methods: roulette wheel, stochastic remainder sampling, tournament selection, stochastic universal sampling, linear−ranking−then−SUS.
- Random or superuniform initialization of "ordinary" (non−permutation) binary or non−binary chromosomes; random initialization of permutation−based chromosomes; or user−supplied initialization of arbitrary types of chromosomes.
- Binary or non−binary alphabetic fields on value−based chromosomes, including different user−definable field sizes.
- 3 crossovers for value−based representations: 1−pt, 2−pt, and uniform, all of which operate at field boundaries if a non−binary alphabet is used.
- 4 crossovers for order−based reps: PMX, order−based, uniform order−based, and cycle.
- 4 mutations: fast bitwise, multiple−field, swap and random sublist scramble.
- Fitness scaling: linear scaling, Boltzmann scaling, sigma truncation, window scaling, ranking.
- **Plus** a whole lot more....

GAS

- Web site: starship.skyport.net/crew/gandalf
- FTP site: ftp.coe.uga.edu/users/jae/ai

GAS means "Genetic Algorithms Stuff".

GAS is freeware.

Purpose of GAS is to explore and exploit artificial evolutions. Primary implementation language of GAS is Python. The GAS software package is meant to be a Python framework for applying genetic algorithms. It contains an example application where it is tried to breed Python program strings. This special problem falls into the category of Genetic Programming (GP), and/or Automatic Programming. Nevertheless, GAS tries to be useful for other applications of Genetic Algorithms as well.

GECO

- FTP site: ftp://ftp.aic.nrl.navy.mil/pub/galist/src/

GECO (Genetic Evolution through Combination of Objects), an extendible object−oriented tool−box for constructing genetic algorithms (in Lisp). It provides a set of extensible classes and methods designed for generality. Some simple examples are also provided to illustrate the intended use.

GPdata

- FTP site: ftp.cs.bham.ac.uk/pub/authors/W.B.Langdon/gp−code/
- Documentation (GPdata−icga−95.ps): cs.ucl.ac.uk/genetic/papers/

GPdata−3.0.tar.gz (C++) contains a version of Andy Singleton's GP−Quick version 2.1 which has been extensively altered to support:

- Indexed memory operation (cf. teller)
- multi tree programs
- Adfs
- parameter changes without recompilation
- populations partitioned into demes
- (A version of) pareto fitness

This ftp site also contains a small C++ program (ntrees.cc) to calculate the number of different there are of a given length and given function and terminal set.

gpjpp Genetic Programming in Java

- [Dead Link] Web site: http://www.turbopower.com/~kimk/gpjpp.asp
- Anyone who knows where to find gpjpp, please let me know.

gpjpp is a Java package I wrote for doing research in genetic programming. It is a port of the gpc++ kernel written by Adam Fraser and Thomas Weinbrenner. Included in the package are four of Koza's standard examples: the artificial ant, the hopping lawnmower, symbolic regression, and the boolean multiplexer. Here is a partial list of its features:

- graphic output of expression trees
- efficient diversity checking
- Koza's greedy over−selection option for large populations
- extensible GPRun class that encapsulates most details of a genetic programming test
- more robust and efficient streaming code, with automatic checkpoint and restart built into the GPRun class
- an explicit complexity limit that can be set on each GP
- additional configuration variables to allow more testing without recompilation
- support for automatically defined functions (ADFs)
- tournament and fitness proportionate selection
- demetic grouping
- optional steady state population
- subtree crossover
- swap and shrink mutation

GP Kernel

- Web site (???): www.emk.e−technik.th−darmstadt.de/~thomasw/gp.html

The GP kernel is a C++ class library that can be used to apply genetic programming techniques to all kinds of problems. The library defines a class hierarchy. An integral component is the ability to produce automatically defined functions as found in Koza's "Genetic Programming II". Technical documentation (postscript format) is included. There is also a short introduction into genetic programming.

Functionality includes; Automatically defined functions (ADFs), tournament and fitness proportionate selection, demetic grouping, optional steady state genetic programming kernel, subtree crossover, swap and shrink mutation, a way of changing every parameter of the system without recompilation, capacity for multiple populations, loading and saving of populations and genetic programs, standard random number generator, internal parameter checks.

jaga

- Web site: cs.felk.cvut.cz/~koutnij/studium/jaga/jaga.html

Simple genetic algorithm package written in Java.

lil−gp

- Web site: GARAGe.cps.msu.edu/software/software−index.html#lilgp
- FTP site: garage.cps.msu.edu/pub/GA/lilgp/

patched lil−gp *

- Web site: www.cs.umd.edu/users/seanl/gp/

lil−gp is a generic 'C' genetic programming tool. It was written with a number of goals in mind: speed, ease of use and support for a number of options including:

- Generic 'C' program that runs on UNIX workstations
- Support for multiple population experiments, using arbitrary and user settable topologies for exchange, for a single processor (i.e., you can do multiple population gp experiments on your PC).
- lil−gp manipulates trees of function pointers which are allocated in single, large memory blocks for speed and to avoid swapping.

* The patched lil−gp kernel is strongly−typed, with modifications on multithreading, coevolution, and other tweaks and features.

Lithos

- Web site: www.esatclear.ie/~rwallace/lithos.html

Lithos is a stack based evolutionary computation system. Unlike most EC systems, its representation language is computationally complete, while also being faster and more compact than the S−expressions used in genetic programming. The version presented here applies the system to the game of Go, but can be changed to other problems by simply plugging in a different evaluation function. ANSI C source code is provided.

PGAPack

Parallel Genetic Algorithm Library

- Web site: www.mcs.anl.gov/~levine/PGAPACK/
- FTP site: ftp.mcs.anl.gov/pub/pgapack/

PGAPack is a general−purpose, data−structure−neutral, parallel genetic algorithm library. It is intended to provide most capabilities desired in a genetic algorithm library, in an integrated, seamless, and portable manner. Key features are in PGAPack V1.0 include:

- ♦ Callable from Fortran or C.
- ♦ Runs on uniprocessors, parallel computers, and workstation networks.
- ♦ Binary−, integer−, real−, and character−valued native data types.
- ♦ Full extensibility to support custom operators and new data types.
- ♦ Easy−to−use interface for novice and application users.
- ♦ Multiple levels of access for expert users.
- ♦ Parameterized population replacement.
- ♦ Multiple crossover, mutation, and selection operators.
- ♦ Easy integration of hill−climbing heuristics.
- ♦ Extensive debugging facilities.
- ♦ Large set of example problems.
- ♦ Detailed users guide.

PIPE

- ♦ Web site: www.idsia.ch/~rafal/research.html
- ♦ FTP site: ftp.idsia.ch/pub/rafal

Probabilistic Incremental Program Evolution (PIPE) is a novel technique for automatic program synthesis. The software is written in C. It

- ♦ is easy to install (comes with an automatic installation tool).
- ♦ is easy to use: setting up PIPE_V1.0 for different problems requires a minimal amount of programming. User−written, application− independent program parts can easily be reused.
- ♦ is efficient: PIPE_V1.0 has been tuned to speed up performance.
- ♦ is portable: comes with source code (optimized for SunOS 5.5.1).
- ♦ is extensively documented(!) and contains three example applications.
- ♦ supports statistical evaluations: it facilitates running multiple experiments and collecting results in output files.
- ♦ includes testing tool for testing generalization of evolved programs.
- ♦ supports floating point and integer arithmetic.
- ♦ has extensive output features.
- ♦ For lil−gp users: Problems set up for lil−gp 1.0 can be easily ported to PIPE_v1.0. The testing tool can also be used to process programs evolved by lil−gp 1.0.

Sugal

- ♦ Web site: www.trajan−software.demon.co.uk/sugal.htm

Sugal [soo−gall] is the SUnderland Genetic ALgorithm system. The aim of Sugal is to support research and implementation in Genetic Algorithms on a common software platform. As such, Sugal supports a large number of variants of Genetic Algorithms, and has extensive features to support customization and extension.

4.2 EC software kits/applications

These are various applications, software kits, etc. meant for research in the field of evolutionary computing. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

ADATE

- Web site: www−ia.hiof.no/~rolando/adate_intro.html

ADATE (Automatic Design of Algorithms Through Evolution) is a system for automatic programming i.e., inductive inference of algorithms, which may be the best way to develop artificial and general intelligence.

The ADATE system can automatically generate non−trivial and novel algorithms. Algorithms are generated through large scale combinatorial search that employs sophisticated program transformations and heuristics. The ADATE system is particularly good at synthesizing symbolic, functional programs and has several unique qualities.

esep & xesep

- Web site(esep): www.iit.edu/~linjinl/esep.html
- Web site(xesep): www.iit.edu/~linjinl/xesep.html

This is a new scheduler, called Evolution Scheduler, based on Genetic Algorithms and Evolutionary Programming. It lives with original Linux priority scheduler.This means you don't have to reboot to change the scheduling policy. You may simply use the manager program esep to switch between them at any time, and esep itself is an all−in−one for scheduling status, commands, and administration. We didn't intend to remove the original priority scheduler; instead, at least, esep provides you with another choice to use a more intelligent scheduler, which carries out natural competition in an easy and effective way.

Xesep is a graphical user interface to the esep (Evolution Scheduling and Evolving Processes). It's intended to show users how to start, play, and feel the Evolution Scheduling and Evolving Processes, including sub−programs to display system status, evolving process status, queue status, and evolution scheduling status periodically in as small as one mini−second.

Corewars

- Web site: corewars.sourceforge.net/
- SourceForge site: sourceforge.net/project/corewars/

Corewars is a game which simulates a virtual machine with a number of programs. Each program tries to crash the others. The program that lasts the longest time wins. A number of sample programs are provided and new programs can be written by the player. Screenshots are available at the Corewars homepage.

Corewar VM

- Web site: www.jedi.claranet.fr/

This is a virtual machine written in Java (so it is a virtual machine for another virtual machine !) for a Corewar game.

FSM−Evolver

- Web site (???): pages.prodigy.net/czarneckid

A Java (jdk−v1.0.2+) code library that is used to evolve finite state machines. The problem included in the package is the Artificial Ant problem. You should be able to compile the .java files and then run: java ArtificialAnt.

GPsys

- Web site: www.cs.ucl.ac.uk/staff/A.Qureshi/gpsys.html

GPsys (pronounced gipsys) is a Java (requires Java 1.1 or later) based Genetic Programming system developed by Adil Qureshi. The software includes documentation, source and executables.

Feature Summary:

- Steady State engine
- ADF support
- Strongly Typed
    1. supports generic functions and terminals
    2. has many built−in primitives
    3. includes indexed memory
- Save/Load feature
    1. can save/load current generation to/from a file
    2. data stored in GZIP compression format to minimise disk requirements
    3. uses serialisable objects for efficiency
- Fully Documented
- Example Problems
    1. Lawnmower (including GUI viewer)
    2. Symbolic Regression
- Totally Parameterised
- Fully Object Oriented and Extensible
- High Performance
- Memory Efficient

Grany−3

- Web site: www.mandrakesoft.com/~gc/html/personal−homepage/html/grany.html

Grany−3 is a full−featured cellular automaton simulator, made in C++ with Gtk−−, flex++/bison++, doxygen and gettext, useful to granular media physicists.

JGProg

- Web site: www.linuxstart.com/~groovyjava/JGProg/

Genetic Programming (JGProg) is an open−source Java implementation of a strongly−typed Genetic Programming experimentation platform. Two example "worlds" are provided, in which a population evolves and solves the problem.

---

5. Alife & Complex Systems

Alife takes yet another approach to exploring the mysteries of intelligence. It has many aspects similar to EC and Connectionism, but takes these ideas and gives them a meta−level twist. Alife emphasizes the development of intelligence through *emergent* behavior of *complex adaptive systems*. Alife stresses the social or group based aspects of intelligence. It seeks to understand life and survival. By studying the behaviors of groups of 'beings' Alife seeks to discover the way intelligence or higher order activity emerges from seemingly simple individuals. Cellular Automata and Conway's Game of Life are probably the most commonly known applications of this field. Complex Systems (abbreviated CS) are very similar to alife in the way the are approached, just more general in definition (ie. alife is a type of complex system). Usually complex system software takes the form of a simulator.

5.1 Alife & CS class/code libraries

These are libraries of code or classes for use in programming within the artificial life field. They are not meant as stand alone applications, but rather as tools for building your own applications.

CASE

- Web site: www.iu.hioslo.no/~cell/
- FTP site: ftp.iu.hioslo.no/pub/

CASE (Cellular Automaton Simulation Environment) is a C++ toolkit for visualizing discrete models in two dimensions: so−called cellular automata. The aim of this project is to create an integrated framework for creating generalized cellular automata using the best, standardized technology of the day.

Integrating Modelling Toolkit

- Web site: flock.cbl.umces.edu/imt/

The Integrating Modelling Toolkit (IMT) is a generic, comprehensive, and extensible set of abstractions allowing definition and use of interoperable model components. Modellers create an IMT "world" made of IMT "agents" that will perform each a particular phase of a modelling task. The core set of IMT agents can describe generic, modular, distributed model components, either native to the IMT or integrating existing simulation toolkits, specialized for tasks that range from simple calculation of functions in an interpreted language to spatially explicit simulation, model optimization, GIS analysis, visualization and advanced statistical analysis. IMT agents are designed to easily "glue" together in higher−level simulations integrating different modelling paradigms and toolkits. The IMT can be easily extended by users and developers through a convenient plug−in mechanism

John von Neumann Universal Constructor

- Web site: alife.santafe.edu/alife/software/jvn.html
- FTP site: alife.santafe.edu/pub/SOFTWARE/jvn/

The universal constructor of John von Neumann is an extension of the logical concept of universal computing machine. In the cellular environment proposed by von Neumann both computing and constructive universality can be achieved. Von Neumann proved that in his cellular lattice both a Turing machine and a machine capable of producing any other cell assembly, when fed with a suitable program, can be embedded. He called the latter machine a "universal constructor" and showed that, when provided with a program containing its own description, this is capable of self−reproducing.

MAML

- Web site: www.syslab.ceu.hu/maml/

The current version of MAML is basically an extension to Objective−C (using the Swarm libraries). It consists of a couple of 'macro−keywords' that define the general structure of a simulation. The remaining must be filled with pure swarm−code. A MAML−to−Swarm (named xmc) compiler is also being developed which compiles the source code into a swarm application.

Swarm

- Web site: www.swarm.org
- FTP site: ftp.swarm.org/pub/swarm/

The swarm Alife simulation kit. Swarm is a simulation environment which facilitates development and experimentation with simulations involving a large number of agents behaving and interacting within a dynamic environment. It consists of a collection of classes and libraries written in Objective−C and allows great flexibility in creating simulations and analyzing their results. It comes with three demos and good documentation.

5.2 Alife & CS software kits, applications, etc.

These are various applications, software kits, etc. meant for research in the field of artificial life. Their ease of use will vary, as they were designed to meet some particular research interest more than as an easy to use commercial package.

Achilles

- Web site: achilles.sourceforge.net

Achilles is an evolution simulation based on Larry Yaeger's PolyWorld. It uses Hebbian neural networks, and an extremely simplified physical model that allows virtual organisms to interact freely in a simulated environment.

Avida

- Web site: http://www.krl.caltech.edu/avida/home/software.html
- Web site: www.krl.caltech.edu/avida/pubs/nature99/

The computer program avida is an auto−adaptive genetic system designed primarily for use as a platform in Artificial Life research. The avida system is based on concepts similar to those employed by the tierra program, that is to say it is a population of self−reproducing strings with a Turing−complete genetic basis subjected to Poisson−random mutations. The population adapts to the combination of an intrinsic fitness landscape (self−reproduction) and an externally imposed (extrinsic) fitness function provided by the researcher. By studying this system, one can examine evolutionary adaptation, general traits of living systems (such as self−organization), and other issues pertaining to theoretical or evolutionary biology and dynamic systems.

BugsX

- FTP site: ftp.de.uu.net/pub/research/ci/Alife/packages/bugsx/

Display and evolve biomorphs. It is a program which draws the biomorphs based on parametric plots of Fourier sine and cosine series and let's you play with them using the genetic algorithm.

The Cellular Automata Simulation System

- Web site: www.cs.runet.edu/~dana/ca/cellular.html

The system consists of a compiler for the Cellang cellular automata programming language, along with the corresponding documentation, viewer, and various tools. Cellang has been undergoing refinement for the last several years (1991−1995), with corresponding upgrades to the compiler. Postscript versions of the tutorial and language reference manual are available for those wanting more detailed information. The most important distinguishing features of Cellang, include support for:

- any number of dimensions;
- compile time specification of each dimensions size; cell neighborhoods of any size (though bounded at compile time) and shape;
- positional and time dependent neighborhoods;
- associating multiple values (fields), including arrays, with each cell;
- associating a potentially unbounded number of mobile agents [Agents are mobile entities based on a mechanism of the same name in the Creatures system, developed by Ian Stephenson (ian@ohm.york.ac.uk).] with each cell; and
- local interactions only, since it is impossible to construct automata that contain any global control or references to global variables.

Cyphesis

- Web site: www.worldforge.org/website/servers/cyphesis/

Cyphesis will be the AI Engine, or more plainly, the intelligence behind Worldforge (WF). Cyphesis will aims to achieve 'live' virtual worlds. Animals will have young, prey on each other and eventually die. Plants grow, flower, bear fruit and even die just as they do in real life. When completed, NPCs in Cyphesis will do all sorts of interesting things like attempt to acomplish ever−changing goals that NPCs set for themselves, gossip to PCs and other NPCs, live, die and raise children. Cyphesis aims to make NPCs act just like you and me.

dblife & dblifelib

- FTP site: ftp.cc.gatech.edu/ac121/linux/games/amusements/life/

dblife: Sources for a fancy Game of Life program for X11 (and curses). It is not meant to be incredibly fast (use xlife for that:−). But it IS meant to allow the easy editing and viewing of Life objects and has some powerful features. The related dblifelib package is a library of Life objects to use with the program.

dblifelib: This is a library of interesting Life objects, including oscillators, spaceships, puffers, and other weird things. The related dblife package contains a Life program which can read the objects in the Library.

Drone

- Web site: pscs.physics.lsa.umich.edu/Software/Drone/

Drone is a tool for automatically running batch jobs of a simulation program. It allows sweeps over arbitrary sets of parameters, as well as multiple runs for each parameter set, with a separate random seed for each run. The runs may be executed either on a single computer or over the Internet on a set of remote hosts. Drone is written in Expect (an extension to the Tcl scripting language) and runs under Unix. It was originally designed for use with the Swarm agent−based simulation framework, but Drone can be used with any simulation program that reads parameters from the command line or from an input file.

EBISS

- Web site: www.ebiss.org/english/

EBISS is a multi−disciplinary, open, collaborative project aimed at investigating social problems by means of computational modeling and social simulations. During the past four years we have been developing SARA, a multi−agent gaming simulation platform providing for easy construction of simulations and gamings.

We believe that in order to have a break−through in the difficult task of understanding real−world complex social problems, we need to gather researchers and experts with different backgrounds not only in discussion forums, but in a tighter cooperative task of building and sharing common experimental platforms.

EcoLab

- Web site: parallel.acsu.unsw.edu.au/rks/ecolab.html

EcoLab is a system that implements an abstract ecology model. It is written as a set of Tcl/Tk commands so that the model parameters can easily be changed on the fly by means of editing a script. The model itself is written in C++.

Game Of Life (GOL)

- Web site: www.arrakeen.demon.co.uk/downloads.html
- FTP site: metalab.unc.edu/pub/Linux/science/ai/life

GOL is a simulator for conway's game of life (a simple cellular automata), and other simple rule sets. The emphasis here is on speed and scale, in other words you can setup large and fast simulations.

gant

- Web site: gant.sourceforge.net

This project is an ANSI C++ implementation of the Generalized Langton Ant, which lives on a torus.

gLife

- Web site: glife.sourceforge.net
- SourceForge site: sourceforge.net/project/glife/

This program is similiar to "Conway's Game of Life" but yet it is very different. It takes "Conway's Game of Life" and applies it to a society (human society). This means there is a very different (and much larger) ruleset than in the original game. Things need to be taken into account such as the terrain, age, sex, culture, movement, etc

Grany−3

- Web site: altern.org/gcottenc/html/grany.html

Grany−3 is a full−featured cellular automaton simulator, made in C++ with Gtk−−, flex++/bison++, doxygen and gettext, useful to granular media physicists.

Langton's Ant

- Web site: theory.org/software/ant/

Langton's Ant is an example of a finite−state cellular automata. The ant (or ants) start out on a grid. Each cell is either black or white. If the ant is on a black square, it turns right 90 and moves forward one unit. If the ant is on a white square, it turns left 90 and moves forward one unit. And when the ant leaves a square, it inverts the color. The neat thing about Langton's Ant is that no matter what pattern field you start it out on, it eventually builds a "road," which is a series of 117 steps that repeat indefinitely, each time leaving the ant displaced one pixel vertically and horizontally.

LEE

- Web site: dollar.biz.uiowa.edu/~fil/LEE/
- FTP site: dollar.biz.uiowa.edu/pub/fil/LEE/

LEE (Latent Energy Environments) is both an Alife model and a software tool to be used for simulations within the framework of that model. We hope that LEE will help understand a broad range of issues in theoretical, behavioral, and evolutionary biology. The LEE tool described here consists of approximately 7,000 lines of C code and runs in both Unix and Macintosh platforms.

Net−Life & ZooLife

- FTP site: ftp.coe.uga.edu/users/jae/alife/

*(netlife−2.0.tar.gz contains both Net−Life and ZooLife)

Net−Life is a simulation of artificial−life, with neural "brains" generated via slightly random techniques. Net−Life uses artificial neural nets and evolutionary algorithms to breed artificial organisms that are similar to single cell organisms. Net−life uses asexual reproduction of its fittest individuals with a chance of mutation after each round to eventually evolve successful life−forms.

ZooLife is a simulation of artificial−life. ZooLife uses probabilistic methods and evolutionary algorithms to breed artificial organisms that are similar to plant/animal zoo organisms. ZooLife uses asexual reproduction with a chance of mutation.

POSES++

- Web site: www.tu−chemnitz.de/ftp−home/pub/Local/simulation/poses++/www/index.html

The POSES++ software tool supports the development and simulation of models. Regarding the simulation technique models are suitable reproductions of real or planned systems for their simulative investigation.

In all industrial sectors or branches POSES++ can model and simulate any arbitrary system which is based on a discrete and discontinuous behaviour. Also continuous systems can mostly be handled like discrete systems e.g., by quantity discretion and batch processing.

Primordial Soup

- Web site: alife.santafe.edu/alife/software/psoup.html

Primordial Soup is an artificial life program. Organisms in the form of computer software loops live in a shared memory space (the "soup") and self−reproduce. The organisms mutate and evolve, behaving in accordance with the principles of Darwinian evolution.

The program may be started with one or more organisms seeding the soup. Alternatively, the system may be started "sterile", with no organisms present. Spontaneous generation of self−reproducing organisms has been observed after runs as short as 15 minutes.

Tierra

- Web site: www.hip.atr.co.jp/~ray/tierra/tierra.html
- FTP site: alife.santafe.edu/pub/SOFTWARE/Tierra/
- Alternate
- FTP site: ftp.cc.gatech.edu/ac121/linux/science/biology/

Tierra's written in the C programming language. This source code creates a virtual computer and its operating system, whose architecture has been designed in such a way that the executable machine codes are evolvable. This means that the machine code can be mutated (by flipping bits at random) or recombined (by swapping segments of code between algorithms), and the resulting code remains functional enough of the time for natural (or presumably artificial) selection to be able to improve the code over time.

TIN

- FTP site: ftp.coe.uga.edu/users/jae/alife/

This program simulates primitive life−forms, equipped with some basic instincts and abilities, in a 2D environment consisting of cells. By mutation new generations can prove their success, and thus passing on "good family values".

The brain of a TIN can be seen as a collection of processes, each representing drives or impulses to behave a certain way, depending on the state/perception of the environment (e.g. presence of food, walls, neighbors, scent traces) These behavior process currently are : eating, moving, mating, relaxing, tracing others, gathering food and killing. The process with the highest impulse value takes control, or in other words: the tin will act according to its most urgent need.

XLIFE

- FTP site: ftp.cc.gatech.edu/ac121/linux/games/amusements/life/

This program will evolve patterns for John Horton Conway's game of Life. It will also handle general cellular automata with the orthogonal neighborhood and up to 8 states (it's possible to recompile for more states, but very expensive in memory). Transition rules and sample patterns are provided for the 8−state automaton of E. F. Codd, the Wireworld automaton, and a whole class of `Prisoner's Dilemma' games.

Xtoys

- Web site: penguin.phy.bnl.gov/www/xtoys.html

xtoys contains a set of cellular automata simulators for X windows. Programs included are:

- xising −−− a two dimensional Ising model simulator,
- xpotts −−− the two dimensional Potts model,
- xautomalab −−− a totalistic cellular automaton simulator,
- xsand −−− for the Bak, Tang, Wiesenfeld sandpile model,
- xwaves −−− demonstrates three different wave equations,
- schrodinger −−− play with the Scrodinger equation in an adjustable potential.

---

6. Agents

Also known as intelligent software agents or just agents, this area of AI research deals with simple applications of small programs that aid the user in his/her work. They can be mobile (able to stop their execution on one machine and resume it on another) or static (live in one machine). They are usually specific to the task (and therefore fairly simple) and meant to help the user much as an assistant would. The most popular (ie. widely known) use of this type of application to date are the web robots that many of the indexing engines (eg. webcrawler) use.

AgentK

- FTP site: ftp.csd.abdn.ac.uk/pub/wdavies/agentk

This package synthesizes two well−known agent paradigms: Agent−Oriented Programming, Shoham (1990), and the Knowledge Query & Manipulation Language, Finin (1993). The initial implementation of AOP, Agent−0, is a simple language for specifying agent behaviour. KQML provides a standard language for inter−agent communication. Our integration (which we have called Agent−K) demonstrates that Agent−0 and KQML are highly compatible. Agent−K provides the possibility of inter−operable (or open) software agents, that can communicate via KQML and which are programmed using the AOP approach.

Agent

- FTP site: www.cpan.org/modules/by−category/23_Miscellaneous_Modules/Agent/

The Agent is a prototype for an Information Agent system. It is both platform and language independent, as it stores contained information in simple packed strings. It can be packed and shipped across any network with any format, as it freezes itself in its current state.

agentTool

- Web site: en.afit.af.mil/ai/agentool.htm
- Download site: en.afit.af.mil/ai/_vti_bin/shtml.dll/registration.htm

Another Java based agent development framework. Fairly unique in that it emphasizes the use of a GUI for designing the system which will "semi−automatically synthesize multiagent systems to meet those requirements". You need a java enabled browser to download. :P

Aglets Workbench

- Web site: www.trl.ibm.co.jp/aglets/

An aglet is a Java object that can move from one host on the Internet to another. That is, an aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the aglet moves, it takes along its program code as well as its state (data). A built−in security mechanism makes it safe for a computer to host untrusted aglets. The Java Aglet API (J−AAPI) is a proposed public standard for interfacing aglets and their environment. J−AAPI contains methods for initializing an aglet, message handling, and dispatching, retracting, deactivating/activating, cloning, and disposing of the aglet. J−AAPI is simple, flexible, and stable. Application developers can write platform−independent aglets and expect them to run on any host that supports J−AAPI.

A.L.I.C.E.

- Web site: www.alicebot.org

The ALICE software implements AIML (Artificial Intelligence Markup Language), a non−standard evolving markup language for creating chat robots. The primary design feature of AIML is minimalism. Compared with other chat robot languages, AIML is perhaps the simplest. The pattern matching language is very simple, for example permitting only one wild−card ('*') match character per pattern. AIML is an XML language, implying that it obeys certain grammatical meta−rules. The choice of XML syntax permits integration with other tools such as XML editors. Another motivation for XML is its familiar look and feel, especially to people with HTML experience.

Ara

- Web site: www.uni−kl.de/AG−Nehmer/Projekte/Ara/index_e.html

Ara is a platform for the portable and secure execution of mobile agents in heterogeneous networks. Mobile agents in this sense are programs with the ability to change their host machine during execution while preserving their internal state. This enables them to handle interactions locally which otherwise had to be performed remotely. Ara's specific aim in comparison to similar platforms is to provide full mobile agent functionality while retaining as much as possible of established programming models and languages.

Bee−gent

- Web site: www2.toshiba.co.jp/beegent/index.htm

Bee−gent is a new type of development framework in that it is a 100% pure agent system. As opposed to other systems which make only some use of agents, Bee−gent completely "Agentifies" the communication that takes place between software applications. The applications become agents, and all messages are carried by agents. Thus, Bee−gent allows developers to build flexible open distributed systems that make optimal use of existing applications.

Bond

- Web site: bond.cs.purdue.edu/

Yet another java agent system...

Bond is a Java based distributed object system and agent framework. It implements a message based middleware and associated services like directory, persistence, monitoring and security. Bond allows to easily build multi agent, distributed applications. Another application of Bond will be a Virtual Laboratory supporting data annotation and metacomputing.

D'Agent (was AGENT TCL)

- Web site: agent.cs.dartmouth.edu/software/agent2.0/
- FTP site: ftp.cs.dartmouth.edu/pub/agents/

A transportable agent is a program that can migrate from machine to machine in a heterogeneous network. The program chooses when and where to migrate. It can suspend its execution at an arbitrary point, transport to another machine and resume execution on the new machine. For example, an agent carrying a mail message migrates first to a router and then to the recipient's mailbox. The agent can perform arbitrarily complex processing at each machine in order to ensure that the message reaches the intended recipient.

FIPA−OS

- Web site: fipa−os.sourceforge.net
- Secondary Web site: www.nortelnetworks.com/products/announcements/fipa/

FIPA−OS is an open source implementation of the mandatory elements contained within the FIPA specification for agent interoperability. In addition to supporting the FIPA interoperability concepts, FIPA−OS also provides a component based architecture to enable the development of domain specific agents which can utilise the services of the FIPA Platform agents. It is implemented in Java.

Bots

- Web site: utenti.tripod.it/Claudio1977/bots.html

Another AI−robot battle simulation. Utilizing probablistic logic as a machine learning technique. Written in C++ (with C++ bots).

Cadaver

- Web site: www.erikyyy.de/cadaver/

Cadaver is a simulated world of cyborgs and nature in realtime. The battlefield consists of forests, grain, water, grass, carcass (of course) and lots of other things. The game server manages the game and the rules. You start a server and connect some clients. The clients communicate with the server using a very primitive protocol. They can order cyborgs to harvest grain, attack enemies or cut forest. The game is not intended to be played by humans! There is too much to control. Only for die−hards: Just telnet to the server and you can enter commands by hand. Instead the idea is that you write artificial intelligence clients to beat the other artificial intelligences. You can choose a language (and operating system) of your choice to do that task. It is enough to write a program that communicates on standard input and standard output channels. Then you can use programs like "socket" to connect your clients to the server. It is NOT needed to write TCP/IP code, although i did so :) The battle shall not be boring, and so there is the so called spyboss client that displays the action graphically on screen.

Dunce

- Web site: www.boswa.com/boswabits/

Dunce is a simple chatterbot (conversational AI) and a language for programming such chatterbots. It uses a basic regex pattern matching and a semi−neural rule/response firing mechanism (with excitement/decay cycles).

Dunce is listed about halfway down the page.

FishMarket

- Web site: www.iiia.csic.es/Projects/fishmarket/

FM − The FishMarket project conducted at the Artificial Intelligence Research Institute (IIIA−CSIC) attempts to contribute in that direction by developing FM, an agent−mediated electronic auction house which has been evolved into a test−bed for electronic auction markets. The framework, conceived and implemented as an extension of FM96.5 (a Java−based version of the Fishmarket auction house), allows to define trading scenarios based on fish market auctions (Dutch auctions). FM provides the framework wherein agent designers can perform controlled experimentation in such a way that a multitude of experimental market scenarios−−that we regard as tournament scenarios due to the competitive nature of the domain−− of varying degrees of realism and complexity can be specified, activated, and recorded; and trading (buyer and seller) heterogeneous (human and software) agents compared, tuned and evaluated.

Grasshopper

- Web site: www.grasshopper.de/

Another Java agent system. Full featured and actively developed. Commercial, but free. Historically targeted at embedded systems.

Hive

- Web site: hive.sourceforge.net
- Web site: www.hivecell.net

Hive is a Java software platform for creating distributed applications. Using Hive, programmers can easily create systems that connect and use data from all over the Internet. At its heart, Hive is an environment for distributed agents to live, communicating and moving to fulfill applications. We are trying to make the Internet alive.

ICM

- Web site: www.nar.fujitsulabs.com/icm/
- SourceForge site: sourceforge.net/projects/networkagent/

The Inter−Agent Communication Model (ICM) is a communication mechanism that can be used for sending messages between agents in an asynchronous fashion. Its intended application area is as a transportation mechanism for agent communication languages (ACLs), such as KQML and FIPA's ACL.

Jacomma

- Web site: jacomma.sourceforge.net
- SourceForge site: sourceforge.net/projects/jacomma/

Jacomma is an agent development platform/framework for developing distributed, mobile, and reactive information agents with heterogeneous communication capabilities, in Java and JPython.

Jacomma provides a development framework and an execution environment, which sits on top of the Inter−Agent Communication Model infrastructure. The ICM defines a communication protocol, a store and forward messaging architecture, and low level communication infrastructure for message exchange. Communication is truly asynchronous, based on TCP sockets.

ICM has an entry in this howto, or you can find it via a link off the site.

Jade

- Web site: sharon.cselt.it/projects/jade/

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi−agent systems through a middle−ware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run−time by moving agents from one machine to another one, as and when required.

JAFMAS

- Web site: www.ececs.uc.edu/~abaker/JAFMAS

JAFMAS provides a framework to guide the coherent development of multiagent systems along with a set of classes for agent deployment in Java. The framework is intended to help beginning and expert developers structure their ideas into concrete agent applications. It directs development from a speech−act perspective and supports multicast and directed communication, KQML or other speech−act performatives and analysis of multiagent system coherency and consistency.

Only four of the provided Java classes must be extended for any application. Provided examples of the N−Queens and Supply Chain Integration use only 567 and 1276 lines of additional code respectively for implementation.

JAM Agent

- Web site: members.home.net/marcush/IRS/

JAM supports both top−down, goal−based reasoning and bottom−up data−driven reasoning. JAM selects goals and plans based on maximal priority if metalevel reasoning is not used, or user−developed metalevel reasoning plans if they exist. JAM's conceptualization of goals and goal achievement is more classically defined (UMPRS is more behavioral performance−based than truly goal−based) and makes the distinction between plans to achieve goals and plans that simply encode behaviors. Goal−types implemented include achievement (attain a specified world state), maintenance (re−attain a specified world state), and performance. Execution of multiple simultaneous goals are supported, with suspension and resumption capabilities for each goal (i.e., intention) thread. JAM plans have explicit precondition and runtime attributes that restrict their applicability, a postcondition attribute, and a plan attributes section for specifying plan/domain−specific plan features. Available plan constructs include: sequencing, iteration, subgoaling, atomic (i.e., non−interruptable) plan segments, n−branch deterministic and non−deterministic conditional execution, parallel execution of multiple plan segments, goal−based or world state−based synchronization, an explicit failure−handling section, and Java primitive function definition through building it into JAM as well as the invocation of predefined (i.e., legacy) class members via Java's reflection capabilities without having to build it into JAM.

JATLite

- Web site: java.stanford.edu/java_agent/html/

JATLite is providing a set of java packages which makes easy to build multi−agent systems using Java. JATLite provides only light−weight, small set of packages so that the developers can handle all the packages with little efforts. For flexibility JATLite provides four different layers from abstract to Router implementation. A user can access any layer we are providing. Each layer has a different set of assumptions. The user can choose an appropriate layer according to the assumptions on the layer and user's application. The introduction page contains JATLite features and the set of assumptions for each layer.

JATLiteBeans

- Web site: waitaki.otago.ac.nz/JATLiteBean/
- Improved, easier−to−use interface to JATLite features including KQML message parsing, receiving, and sending.

- Extensible architecture for message handling and agent "thread of control" management
- Useful functions for parsing of simple KQML message content
- JATLiteBean supports automatic advertising of agent capabilities to facilitator agents
- Automatic, optional, handling of the "forward" performative
- Generic configuration file parser
- KQML syntax checker

Java(tm) Agent Template

- Web site: cdr.stanford.edu/ABE/JavaAgent.html

The JAT provides a fully functional template, written entirely in the Java language, for constructing software agents which communicate peer−to−peer with a community of other agents distributed over the Internet. Although portions of the code which define each agent are portable, JAT agents are not migratory but rather have a static existence on a single host. This behavior is in contrast to many other "agent" technologies. (However, using the Java RMI, JAT agents could dynamically migrate to a foreign host via an agent resident on that host). Currently, all agent messages use KQML as a top−level protocol or message wrapper. The JAT includes functionality for dynamically exchanging "Resources", which can include Java classes (e.g. new languages and interpreters, remote services, etc.), data files and information inlined into the KQML messages.

Java−To−Go

- Web site: ptolemy.eecs.berkeley.edu/dgm/javatools/java−to−go/

Java−To−Go is an experimental infrastructure that assists in the development and experimentation of mobile agents and agent−based applications for itinerative computing (itinerative computing: the set of applications that requires site−to−site computations. The main emphasis here is on a easy−to−setup environment that promotes quick experimentation on mobile agents.

Kafka

- Web site: www.fujitsu.co.jp/hypertext/free/kafka/

Kafka is yet another agent library designed for constructing multi−agent based distributed applications. Kafka is a flexible, extendable, and easy−to−use java class library for programmers who are familiar with distributed programming. It is based on Java's RMI and has the following added features:

- Runtime Reflection: Agents can modify their behaviour (program codes) at runtime. The behaviour of the agent is represented by an abstract class Action. It is useful for remote maintenance or installation services.
- Remote Evaluation: Agents can receive and evaluate program codes (classes) with or without the serialized object. Remote evaluation is a fundamental function of a mobile agent and is thought to be a push model of service delivery.
- Distributed Name Service: Agents have any number of logical names that don't contain the host name. These names can be managed by the distributed directories.
- Customizable security policy: a very flexible, customizable, 3−layered security model is

implemented in Kafka.
- 100% Java and RMI compatible: Kafka is written completely in Java. Agent is a Java RMI server object itself. So, agents can directly communicate with other RMI objects.

Khepera Simulator

- Web site: diwww.epfl.ch/lami/team/michel/khep−sim/

Khepera Simulator is a public domain software package written by Olivier MICHEL during the preparation of his Ph.D. thesis, at the Laboratoire I3S, URA 1376 of CNRS and University of Nice−Sophia Antipolis, France. It allows to write your own controller for the mobile robot Khepera using C or C++ languages, to test them in a simulated environment and features a nice colorful X11 graphical interface. Moreover, if you own a Khepera robot, it can drive the real robot using the same control algorithm. It is mainly oriented toward to researchers studying autonomous agents.

lyntin

- Web site: lyntin.sourceforge.net/

Lyntin is an extensible Mud client and framework for the creation of autonomous agents, or bots, as well as mudding in general. Lyntin is centered around Python, a dynamic, object−oriented, and fun programming language and based on TinTin++ a lovely mud client.

Mole

- Web site: mole.informatik.uni−stuttgart.de/

Mole is an agent system supporting mobile agents programmed in Java. Mole's agents consist of a cluster of objects, which have no references to the outside, and as a whole work on tasks given by the user or another agent. They have the ability to roam a network of "locations" autonomously. These "locations" are an abstraction of real, existing nodes in the underlying network. They can use location−specific resources by communicating with dedicated agents representing these services. Agents are able to use services provided by other agents and to provide services as well.

Narval

- Web site: www.logilab.org

Narval is the acronym of "Network Assistant Reasoning with a Validating Agent Language". It is a personal network assistant based on artificial intelligence and agent technologies. It executes recipes (sequences of actions) to perform tasks. It is easy to specify a new action using XML and to implement it using Python. Recipes can be built and debugged using a graphical interface.

NeL

- Web site: www.nevrax.org

NeL is actually a game development library (for massive multi−player games), but I'm including it here as it (will) include a fairly sizable AI library. Here's a blurb from the whitepaper:

The purpose of the AI library is to provide a pragmatic approach to creating a distributed agents platform. Its focus is agents; individual entities that communicate regardless of location, using an action−reaction model.

Penguin!

- FTP site: www.perl.org/CPAN/modules/by−category/23_Miscellaneous_Modules/Penguin/FSG/

Penguin is a Perl 5 module. It provides you with a set of functions which allow you to:

- send encrypted, digitally signed Perl code to a remote machine to be executed.
- receive code and, depending on who signed it, execute it in an arbitrarily secure, limited compartment.

The combination of these functions enable direct Perl coding of algorithms to handle safe internet commerce, mobile information−gathering agents, "live content" web browser helper apps, distributed load−balanced computation, remote software update, distance machine administration, content−based information propagation, Internet−wide shared−data applications, network application builders, and so on.

RealTimeBattle

- Web site: www.lysator.liu.se/realtimebattle/

RealTimeBattle is a programming game, in which robots controlled by programs are fighting each other. The goal is to destroy the enemies, using the radar to examine the environment and the cannon to shoot.

- Game progresses in real time, with the robot programs running as child processes to RealTimeBattle.
- The robots communicate with the main program using the standard input and output.
- Robots can be constructed in almost any programming language.
- Maximum number of robots can compete simultaneously.
- A simple messaging language is used for communication, which makes it easy to start constructing robots.
- Robots behave like real physical object.
- You can create your own arenas.
- Highly configurable.

Remembrance Agents

- Web site: rhodes.www.media.mit.edu/people/rhodes/RA/

Remembrance Agents are a set of applications that watch over a user's shoulder and suggest information relevant to the current situation. While query−based memory aids help with direct recall, remembrance agents are an augmented associative memory. For example, the word−processor version of the RA continuously updates a list of documents relevant to what's being typed or read in an emacs buffer. These suggested documents can be any text files that might be relevant to what you are currently writing or reading. They might be old emails related to the mail you are currently reading, or abstracts from papers and newspaper articles that discuss the topic of your writing.

SimRobot

- Web site: www.informatik.uni−bremen.de/~simrobot/
- FTP site: ftp.uni−bremen.de/pub/ZKW/INFORM/simrobot/

SimRobot is a program for simulation of sensor based robots in a 3D environment. It is written in C++, runs under UNIX and X11 and needs the graphics toolkit XView.

- Simulation of robot kinematics
- Hierarchically built scene definition via a simple definition language
- Various sensors built in: camera, facette eye, distance measurement, light sensor, etc.
- Objects defined as polyeders
- Emitter abstractly defined; can be interpreted e.g. as light or sound
- Camera images computed according to the raytracing or Z−buffer algorithms known from computer graphics
- Specific sensor/motor software interface for communicating with the simulation
- Texture mapping onto the object surfaces: bitmaps in various formats
- Comprehensive visualization of the scene: wire frame w/o hidden lines, sensor and actor values
- Interactive as well as batch driven control of the agents and operation in the environment
- Collision detection
- Extendability with user defined object types
- Possible socket communication to e.g. the Khoros image processing software

Sulawesi

- Web site: wearables.essex.ac.uk/sulawesi/

A framework called Sulawesi has been designed and implemented to tackle what has been considered to be important challenges in a wearable user interface. The ability to accept input from any number of modalities, and perform if necessary a translation to any number of modal outputs. It does this primarily through a set of proactive agents to act on the input.

TclRobots

- FTP site: ftp.neosoft.com/pub/tcl/sorted/games/tclrobots−2.0/
- Redhat Patch: ftp.coe.uga.edu/users/jae/ai/tclrobots−redhat.patch
- RPMs (search at): http://rufus.w3.org/

TclRobots is a programming game, similar to 'Core War'. To play TclRobots, you must write a Tcl program that controls a robot. The robot's mission is to survive a battle with other robots. Two, three, or four robots compete during a battle, each running different programs (or possibly the same program in different robots.) Each robot is equipped with a scanner, cannon, drive mechanism. A single match continues until one robot is left running. Robots may compete individually, or combine in a team oriented battle. A tournament can be run with any number of robot programs, each robot playing every other in a round−robin fashion, one−on−one. A battle simulator is available to help debug robot programs.

The TclRobots program provides a physical environment, imposing certain game parameters to which all robots must adhere. TclRobots also provides a view on a battle, and a controlling user interface. TclRobots requirements: a wish interpreter built from Tcl 7.4 and Tk 4.0.

TKQML

- Web site: www.csee.umbc.edu/tkqml/

TKQML is a KQML application/addition to Tcl/Tk, which allows Tcl based systems to communicate easily with a powerful agent communication language.

The Tocoma Project

- Web site: www.tacoma.cs.uit.no/

An agent is a process that may migrate through a computer network in order to satisfy requests made by clients. Agents are an attractive way to describe network−wide computations.

The TACOMA project focuses on operating system support for agents and how agents can be used to solve problems traditionally addressed by operating systems. We have implemented a series of prototype systems to support agents.

TACOMA Version 1.2 is based on UNIX and TCP. The system supports agents written in C, Tcl/Tk, Perl, Python, and Scheme (Elk). It is implemented in C. This TACOMA version has been in public domain since April 1996.

We are currently focusing on heterogeneity, fault−tolerance, security and management issues. Also, several TACOMA applications are under construction. We implemented StormCast 4.0, a wide−area network weather monitoring system accessible over the internet, using TACOMA and Java. We are now in the process of evaluating this application, and plan to build a new StormCast version to be completed by June 1997.

Ummon

- Web site: www.spacetide.com/projects/ummon/

Ummon is an advanced Open Source chatterbot. The main principle of the bot is that it has no initial knowledge of either words or grammar; it learns everything "on the fly." Numerous AI techniques will be explored in the development of Ummon to achieve realistic "human" communication with support for different, customizable personalities.

UMPRS Agent

- Web site: members.home.net/marcush/IRS/

UMPRS supports top−down, goal−based reasoning and selects goals and plans based on maximal priority. Execution of multiple simultaneous goals are supported, with suspension and resumption capabilities for each goal (i.e., intention) thread. UMPRS plans have an integrated precondition/runtime attribute that constrain their applicability. Available plan constructs include: sequencing, iteration, subgoaling, atomic (i.e., non−interruptable) blocks, n−branch deterministic conditional execution, explicit failure−handling section, and C++ primitive function definition.

Virtual Secretary Project (ViSe)

(Tcl/Tk)

- Web site: www.cs.uit.no/DOS/Virt_Sec

The motivation of the Virtual Secretary project is to construct user−model−based intelligent software agents, which could in most cases replace human for secretarial tasks, based on modern mobile computing and computer network. The project includes two different phases: the first phase (ViSe1) focuses on information filtering and process migration, its goal is to create a secure environment for software agents using the concept of user models; the second phase (ViSe2) concentrates on agents' intelligent and efficient cooperation in a distributed environment, its goal is to construct cooperative agents for achieving high intelligence. (Implemented in Tcl/TclX/Tix/Tk)

VWORLD

- Web site: zhar.net/gnu−linux/projects/vworld/

Vworld is a simulated environment for research with autonomous agents written in prolog. It is currently in something of an beta stage. It works well with SWI−prolog, but should work with Quitnus−prolog with only a few changes. It is being designed to serve as an educational tool for class projects dealing with prolog and autonomous agents. It comes with three demo worlds or environments, along with sample agents for them. There are two versions now. One written for SWI−prolog and one written for LPA−prolog. Documentation is roughly done (with a student/professor framework in mind), and a graphical interface is planned.

WebMate

- Web site: www.cs.cmu.edu/~softagents/webmate/

WebMate is a personal agent for World−Wide Web browsing and searching. It accompanies you when you travel on the internet and provides you what you want.

Features include:

- Searching enhancement, including parallel search, searching keywords refinement using our relevant keywords extraction technology, relevant feedback, etc.

- Browsing assistant, including learning your current interesting, recommending you new URLs according to your profile and selected resources, monitoring bookmarks of Netscape or IE, sending the current browsing page to your friends, etc.
- Offline browsing, including downloading the following pages from the current page for offline browsing.
- Filtering HTTP header, including recording http header and all the transactions between your browser and WWW servers, etc.
- Checking the HTML page to find the errors or dead links, etc.
- Programming in Java, independent of operating system, runing in multi−thread.

Zeus

- Web site: www.labs.bt.com/projects/agents/zeus/

The construction of multi−agent systems involves long development times and requires solutions to some considerable technical difficulties. This has motivated the development of the ZEUS toolkit, which provides a library of software components and tools that facilitate the rapid design, development and deployment of agent system

---

7. Programming languages

While any programming language can be used for artificial intelligence/life research, these are programming languages which are used extensively for, if not specifically made for, artificial intelligence programming.

Allegro CL

- Web site: www.franz.com

Franz Inc's free linux version of their lisp development environment. You can download it or they will mail you a CD free (you don't even have to pay for shipping). It is generally considered to be one of the better lisp platforms.

APRIL

- Web site: sourceforge.net/project/networkagent/

APRIL is a symbolic programming language that is designed for writing mobile, distributed and agent−based systems especially in an Internet environment. It has advanced features such as a macro sub−language, asynchronous message sending and receiving, code mobility, pattern matching, higher−order functions and strong typing. The language is compiled to byte−code which is then interpreted by the APRIL runtime−engine. APRIL now requires the InterAgent Communications Model (ICM) to be installed before it can be installed. [Ed. ICM can be found at the same web site]

B−Prolog

- Web site: www.sci.brooklyn.cuny.edu/~zhou/bprolog.html
- Web site: www.cad.mse.kyutech.ac.jp/people/zhou/bprolog.html

B−Prolog is a compact and complete CLP system that runs Prolog and CLP(FD) programs. An emulator−based system, B−Prolog has a performance comparable with SICStus−Prolog.

- In addition to Edinburgh−style programs, B−Prolog accepts canonical−form programs that can be compiled into more compact and faster code than standard Prolog programs.
- B−Prolog includes an interpreter and provides an interactive interface through which users can consult, list, compile, load, debug and run programs. The command editor facilitates reuse old commands.
- B−Prolog provides a bi−directional interface with C and Java.> resources in C and Java such as Graphics and sockets, and also makes it possible for a Prolog program to be embadded in a C and Java applications.
- B−Prolog supports most of the built−ins in ISO Prolog.
- B−Prolog supports the delaying (co−routining) mechanism, which can be used to implement concurrency, test−and−generate search algorithms, and most importantly constraint propagation algorithms.
- B−Prolog has an efficient constraint compiler for constraints> over finite−domains and Booleans.
- B−Prolog supports the tabling mechanism, which has proven effective for applications including parsing, problem solving, theorem proving, and deductive databases.

Ciao Prolog

- Web site: www.clip.dia.fi.upm.es/Software/Ciao/

Ciao is a complete Prolog system subsuming ISO−Prolog with a novel modular design which allows both restricting and extending the language. Ciao extensions currently include feature terms (records), higher−order, functions, constraints, objects, persistent predicates, a good base for distributed execution (agents), and concurrency. Libraries also support WWW programming, sockets, and external interfaces (C, Java, TCL/Tk, relational databases, etc.). An Emacs−based environment, a stand−alone compiler, and a toplevel shell are also provided.

DHARMI

- Web site: http://megazone.bigpanda.com/~wolf/DHARMI/

DHARMI is a high level spatial, tinker−toy like language who's components are transparently administered by a background process called the Habitat. As the name suggests, the language was designed to make modelling prototypes and handle living data. Programs can be modified while running. This is accomplished by blurring the distinction between source code, program, and data.

ECoLisp

- Web site (???): www.di.unipi.it/~attardi/software.html

ECoLisp (Embeddable Common Lisp) is an implementation of Common Lisp designed for being embeddable into C based applications. ECL uses standard C calling conventions for Lisp compiled functions, which allows C programs to easily call Lisp functions and viceversa. No foreign function interface is required: data can be exchanged between C and Lisp with no need for conversion. ECL is based on a Common Runtime Support (CRS) which provides basic facilities for memory managment, dynamic loading and dumping of binary images, support for multiple threads of execution. The CRS is built into a library that can be linked with the code of the application. ECL is modular: main modules are the program development tools (top level, debugger, trace, stepper), the compiler, and CLOS. A native implementation of CLOS is available in ECL: one can configure ECL with or without CLOS. A runtime version of ECL can be built with just the modules which are required by the application. The ECL compiler compiles from Lisp to C, and then invokes the GCC compiler to produce binaries.

ESTEREL

♦ Web site: www−sop.inria.fr/meije/esterel/

Esterel is both a programming language, dedicated to programming reactive systems, and a compiler which translates Esterel programs into finite−state machines. It is particularly well−suited to programming reactive systems, including real−time systems and control automata.

Only the binary is available for the language compiler. :P

Gödel

♦ Web page: www.cs.bris.ac.uk/~bowers/goedel.html

Gödel is a declarative, general−purpose programming language in the family of logic programming languages. It is a strongly typed language, the type system being based on many−sorted logic with parametric polymorphism. It has a module system. Gödel supports infinite precision integers, infinite precision rationals, and also floating−point numbers. It can solve constraints over finite domains of integers and also linear rational constraints. It supports processing of finite sets. It also has a flexible computation rule and a pruning operator which generalizes the commit of the concurrent logic programming languages. Considerable emphasis is placed on Gödel's meta− logical facilities which provide significant support for meta−programs that do analysis, transformation, compilation, verification, debugging, and so on.

LIFE

♦ Web page: www.isg.sfu.ca/life

LIFE (Logic, Inheritance, Functions, and Equations) is an experimental programming language proposing to integrate three orthogonal programming paradigms proven useful for symbolic computation. From the programmer's standpoint, it may be perceived as a language taking after logic programming, functional programming, and object−oriented programming. From a formal perspective, it may be seen as an instance (or rather, a composition of three instances) of a Constraint Logic Programming scheme due to Hoehfeld and Smolka refining that of Jaffar and Lassez.

CLisp (Lisp)

- Web page: clisp.sourceforge.net
- FTP site: clisp.cons.org/pub/lisp/clisp/source

CLISP is a Common Lisp implementation by Bruno Haible and Michael Stoll. It mostly supports the Lisp described by Common LISP: The Language (2nd edition) and the ANSI Common Lisp standard. CLISP includes an interpreter, a byte−compiler, a large subset of CLOS (Object−Oriented Lisp) , a foreign language interface and, for some machines, a screen editor.

The user interface language (English, German, French) is chosen at run time. Major packages that run in CLISP include CLX & Garnet. CLISP needs only 2 MB of memory.

CMU Common Lisp

- Web page: www.cons.org/cmucl/
- Old Web page: www.mv.com/users/pw/lisp/index.html
- FTP site: ftp2.cons.org/pub/languages/lisp/cmucl/release/
- Linux Installation: www.telent.net/lisp/howto.html

CMU Common Lisp is a public domain "industrial strength" Common Lisp programming environment. Many of the X3j13 changes have been incorporated into CMU CL. Wherever possible, this has been done so as to transparently allow the use of either CLtL1 or proposed ANSI CL. Probably the new features most interesting to users are SETF functions, LOOP and the WITH−COMPILATION−UNIT macro.

GCL (Lisp)

- FTP site: ftp.ma.utexas.edu/pub/gcl/

GNU Common Lisp (GCL) has a compiler and interpreter for Common Lisp. It used to be known as Kyoto Common Lisp. It is very portable and extremely efficient on a wide class of applications. It compares favorably in performance with commercial Lisps on several large theorem−prover and symbolic algebra systems. It supports the CLtL1 specification but is moving towards the proposed ANSI definition. GCL compiles to C and then uses the native optimizing C compilers (e.g., GCC). A function with a fixed number of args and one value turns into a C function of the same number of args, returning one value, so GCL is maximally efficient on such calls. It has a conservative garbage collector which allows great freedom for the C compiler to put Lisp values in arbitrary registers.

It has a source level Lisp debugger for interpreted code, with display of source code in an Emacs window. Its profiling tools (based on the C profiling tools) count function calls and the time spent in each function.

GNU Prolog

- Web site: pauillac.inria.fr/~diaz/gnu−prolog/
- Web site: www.gnu.org/software/prolog/prolog.html

GNU Prolog is a free Prolog compiler with constraint solving over finite domains developed by Daniel Diaz.

GNU Prolog accepts Prolog+constraint programs and produces native binaries (like gcc does from a C source). The obtained executable is then stand−alone. The size of this executable can be quite small since GNU Prolog can avoid to link the code of most unused built−in predicates. The performances of GNU Prolog are very encouraging (comparable to commercial systems).

Beside the native−code compilation, GNU Prolog offers a classical interactive interpreter (top−level) with a debugger.

The Prolog part conforms to the ISO standard for Prolog with many extensions very useful in practice (global variables, OS interface, sockets,...).

GNU Prolog also includes an efficient constraint solver over Finite Domains (FD). This opens contraint logic pogramming to the user combining the power of constraint programming to the declarativity of logic programming.

Mercury

- Web page: www.cs.mu.oz.au/research/mercury/

Mercury is a new, purely declarative logic programming language. Like Prolog and other existing logic programming languages, it is a very high−level language that allows programmers to concentrate on the problem rather than the low−level details such as memory management. Unlike Prolog, which is oriented towards exploratory programming, Mercury is designed for the construction of large, reliable, efficient software systems by teams of programmers. As a consequence, programming in Mercury has a different flavor than programming in Prolog.

Mozart

- Web page: www.mozart−oz.org/

The Mozart system provides state−of−the−art support in two areas: open distributed computing and constraint−based inference. Mozart implements Oz, a concurrent object−oriented language with dataflow synchronization. Oz combines concurrent and distributed programming with logical constraint−based inference, making it a unique choice for developing multi−agent systems. Mozart is an ideal platform for both general−purpose distributed applications as well as for hard problems requiring sophisticated optimization and inferencing abilities. We have developed applications in scheduling and time−tabling, in placement and configuration, in natural language and knowledge representation, multi−agent systems and sophisticated collaborative tools.

SWI Prolog

- Web page: www.swi.psy.uva.nl/projects/SWI−Prolog/
- FTP site: swi.psy.uva.nl/pub/SWI−Prolog/

SWI is a free version of prolog in the Edinburgh Prolog family (thus making it very similar to

Quintus and many other versions). With: a large library of built in predicates, a module system, garbage collection, a two−way interface with the C language, plus many other features. It is meant as a educational language, so it's compiled code isn't the fastest. Although it similarity to Quintus allows for easy porting.

XPCE is freely available in binary form for the Linux version of SWI−prolog. XPCE is an object oriented X−windows GUI development package/environment.

Kali Scheme

- Web site: www.neci.nj.nec.com/PLS/Kali.html

Kali Scheme is a distributed implementation of Scheme that permits efficient transmission of higher−order objects such as closures and continuations. The integration of distributed communication facilities within a higher−order programming language engenders a number of new abstractions and paradigms for distributed computing. Among these are user−specified load−balancing and migration policies for threads, incrementally−linked distributed computations, agents, and parameterized client−server applications. Kali Scheme supports concurrency and communication using first−class procedures and continuations. It integrates procedures and continuations into a message−based distributed framework that allows any Scheme object (including code vectors) to be sent and received in a message.

RScheme

- Web site: www.rscheme.org
- FTP site: ftp.rscheme.org/pub/rscheme/

RScheme is an object−oriented, extended version of the Scheme dialect of Lisp. RScheme is freely redistributable, and offers reasonable performance despite being extraordinarily portable. RScheme can be compiled to C, and the C can then compiled with a normal C compiler to generate machine code. By default, however, RScheme compiles to bytecodes which are interpreted by a (runtime) virtual machine. This ensures that compilation is fast and keeps code size down. In general, we recommend using the (default) bytecode code generation system, and only compiling your time−critical code to machine code. This allows a nice adjustment of space/time tradeoffs. (see web site for details)

Scheme 48

- Web site: www.neci.nj.nec.com/homepages/kelsey/

Scheme 48 is a Scheme implementation based on a virtual machine architecture. Scheme 48 is designed to be straightforward, flexible, reliable, and fast. It should be easily portable to 32−bit byte−addressed machines that have POSIX and ANSI C support. In addition to the usual Scheme built−in procedures and a development environment, library software includes support for hygienic macros (as described in the Revised^4 Scheme report), multitasking, records, exception handling, hash tables, arrays, weak pointers, and FORMAT. Scheme 48 implements and exploits an experimental module system loosely derived from Standard ML and Scheme Xerox. The development environment supports interactive changes to modules and interfaces.

SCM (Scheme)

- Web site: www−swiss.ai.mit.edu/~jaffer/SCM.html
- FTP site: swiss−ftp.ai.mit.edu:/archive/scm/

SCM conforms to the Revised^4 Report on the Algorithmic Language Scheme and the IEEE P1178 specification. Scm is written in C. It uses the following utilities (all available at the ftp site).

- SLIB (Standard Scheme Library) is a portable Scheme library which is intended to provide compatibility and utility functions for all standard Scheme implementations, including SCM, Chez, Elk, Gambit, MacScheme, MITScheme, scheme−>C, Scheme48, T3.1, and VSCM, and is available as the file slib2c0.tar.gz. Written by Aubrey Jaffer.
- JACAL is a symbolic math system written in Scheme, and is available as the file jacal1a7.tar.gz.
- Interfaces to standard libraries including REGEX string regular expression matching and the CURSES screen management package.
- Available add−on packages including an interactive debugger, database, X−window graphics, BGI graphics, Motif, and Open−Windows packages.
- A compiler (HOBBIT, available separately) and dynamic linking of compiled modules.

Shift

- Web site: www.path.berkeley.edu/shift/

Shift is a programming language for describing dynamic networks of hybrid automata. Such systems consist of components which can be created, interconnected and destroyed as the system evolves. Components exhibit hybrid behavior, consisting of continuous−time phases separated by discrete−event transitions. Components may evolve independently, or they may interact through their inputs, outputs and exported events. The interaction network itself may evolve.

YAP Prolog

- Web site: www.ncc.up.pt/~vsc/Yap/

YAP is a high−performance Prolog compiler developed at LIACC/Universidade do Porto. Its Prolog engine is based in the WAM (Warren Abstract Machine), with several optimizations for better performance. YAP follows the Edinburgh tradition, and is largely compatible with DEC−10 Prolog, Quintus Prolog, and especially with C−Prolog. Work on the more recent version of YAP strives at several goals:

- Portability: The whole system is now written in C. YAP compiles in popular 32 bit machines, such as Suns and Linux PCs, and in a 64 bit machines, the Alphas running OSF Unix and Linux.
- Performance: We have optimised the emulator to obtain performance comparable to or better than well−known Prolog systems. In fact, the current version of YAP performs better than the original one, written in assembly language.
- Robustness: We have tested the system with a large array of Prolog applications.
- Extensibility: YAP was designed internally from the beginning to encapsulate manipulation of terms. These principles were used, for example, to implement a simple and powerful

C−interface. The new version of YAP extends these principles to accomodate extensions to the unification algorithm, that we believe will be useful to implement extensions such as constraint programming.

- Completeness: YAP has for a long time provided most builtins expected from a Edinburgh Prolog implementation. These include I/O functionality, data−base operations, and modules. Work on YAP aims now at being compatible with the Prolog standard.
- Openess: We would like to make new development of YAP open to the user community.
- Research: YAP has been a vehicle for research within and outside our group. Currently research is going on on parallelisation and tabulation, and we have started work to support constraint handling.