Introducing the new GRASS module g.infer for data-driven rule-based applications

Peter Löwe

Helmholtz Centre Potsdam
GFZ German Research Centre for Geosciences

ploewe@gfz-potsdam.de

Abstract

This paper introduces the new GRASS GIS add-on module g.infer. The module enables rule-based analysis and workflow management in GRASS GIS, via data-driven inference processes based on the expert system shell CLIPS. The paper discusses the theoretical and developmental background that will help prepare the reader to use the module for Knowledge Engineering applications. In addition, potential application scenarios are sketched out, ranging from the rule-driven formulation of nontrivial GIS-classification tasks and GIS workflows to ontology management and intelligent software agents.

Keywords: module g.infer, GRASS, data-driven rule-based application

1. Introduction

Maps are used to represent the world surrounding us. They are put into use as tools to categorize, classify and judge our environments, to make decisions and act accordingly. In more general terms, the science of mapmaking, cartography, is to provide usable and understandable spatial information for a section of space for decision support. The motivation for computer driven cartography, mostly shouldered by Geographic Information Systems (GIS) such as GRASS GIS, is to perform the overall tasks of mapmaking as a workflow, including means to apply the human expertise and know-how required to infer decision-support for human actions. In GIS, the development of such „map-making“ workflows is usually handled by stepwise execution of the consecutive processing steps by a human operator, to create and document the unfolding workflow, by interacting with the actual spatial data. Once a mapping workflow has been laid out, the next step is its automatisation, turning it into software. This can involve scripting, i.e. the definition of an execution-chain of available GIS modules, or programming, which includes the development of new GIS modules. Free and Open Source GIS like GRASS GIS allow rapid development of both solutions as the overall codebase can be exploited, so there is no need to reinvent previously developed functionalities because of copyright infringement issues. However, if a mapping workflow can be formulated by the human GIS operator, but can not be implemented as script or GIS module, there's a problem. In this case, the task at hand is basically solveable, but the available software environment lacks the flexibility to accommodate the workflow within acceptable time and effort constraints. This situation occurs frequently for classification tasks (remote sensing data or similar fields), resulting in the use of suboptimal classification algorithms: The implemented solution is not

oriented on the original task solving strategy, but is limited by available software tools and programming skills.

A similar field is the flexible set up of GIS workflows which needs to adapt the processing chain according to changing constraints such as the availability and quality of data input, again within acceptable time and effort constraints. What is needed in these two scenarios, both for classification and GIS-workflow execution, is a way to encode understood, yet hard to formulate, „rules of thumb“, acquired from human domain experts. A tool based on such rules will excel in scenarios where the following tell-tale indicators exist [28]

- Classification tasks which may not appear demanding, but no robust way for building a solution can be defined in acceptable time and effort.
- Simple workflows, where the processing rules keep changing depending on the available input and other parameters.
- Problems which have not fully understood or are very complex to solve.

In such cases, a rule-based approach, as provided by the new GRASS module g.infer becomes advantageous:

- Rule-based modelling allows to focus on "What to do" instead of "How to do it".
- Rules allow to express solutions to complex problems and to verify them consequently by logging the decision steps leading to a particular solution.
- Separation of logic (know-how) and data allows to keep the know-how to be stored in centralized rule-bases, providing a central point of access for further editing and improvement.
- Rules can also be human-readable, doubling as their own documentation and to be reviewable by domain experts.

While GRASS GIS provides a wide range of classification tools for raster, vector and volume data, a flexible yet generic approach tailored to conveniently express such rules, applicable to all GIS data types, is currently lacking. GRASS 4.x and GRASS 5.x featured the r.infer module, which provided basic rule-based analysis capabilities for raster data [20][23]. The module remains to be ported into GRASS6 and GRASS7. The same holds true for the r.binfer module , which uses an inference engine based on Bayesian statistics (making decisions based on past experience) to assist human experts in a field develop computerized expert systems for land use planning and management, basing bases the probable impacts of a future land use action on the conditional probabilities about the impact of similar past actions [1][2].

2. Artificial Intelligence

Artificial Intelligence (A.I.) is the field of computer science which focuses on the processes of human thinking an their implementation in software. A.I. is divided in various disciplines such as Artificial Life, Software Agents, Neural Networks, Genetic Algorithms, Decision Trees, Frame Systems and Expert Systems [6][25]. Several GRASS GIS modules, including the add-on modules r.fuzzy.* [13][14], r.agent[22], magical [16][17][18], ann.*[24], have been developed to solve tasks related to A.I. disciplines.

Knowledge representation and Classification is the A.I. Discipline focusing on how human knowledge for problem solving can be represented, manipulated and preserved. So called Knowledge-based Systems, also known as Expert Systems, facilitate the encoding of knowledge for automated reasoning or inference, i.e., the processing of data to infer conclusions, which can be mapped out in a GIS. The overall process of making human expertise available through an Expert System is called Knowledge Engineering. A Rule Engine implementing an Expert System instance for a specific knowledge-domain is called a Production Rule System. The term "Production Rule" stems from formal grammars where it is described as an abstract structure that describes a formal language precisely [15]. Such a Production Rule System is the core of g.infer.

3. The C Language Integrated Production System

The C Language Integrated Production System (CLIPS) is a Production Rule System toolkit for building Expert Systems. The project was started by NASA (Johnson Space Center) in 1985, where it was maintained until the 1990s. It is currently hosted at Sourceforge and is provided under a public domain licence. The name is an acronym for "C Language Integrated Production System“, succeeding the original name „NASA's AI Language (NAIL)“[26].

CLIPS is written in C and provides a rule-based data-driven programming language. It resembles in syntax and user interface closely the language LISP [10]. CLIPS traces its origins to Inference's ART which in turn stems from OPS5 [27]. The CLIPS language continued to evolve and includes today paradigms for rule-based, procedural, functional and object-oriented programming. Since 1991, CLIPS includes the CLIPS Object Oriented Language (COOL) for object-oriented development. In the recent CLIPS Version 6.3, rules can be triggered via Objects enabling Object Oriented Modelling to drive the inference process.

Rules in CLIPS can be loose- or close-coupled: In the latter case, the activation („firing“) of a rule explicitly invokes the firing of other rules. This is also referred to as a categoric problem solving approach [26]. On the other hand, loose coupling is achieved by a rule-base manipulating sets of variables or non-ordered facts, with independent rules pattern-matching on these, firing only if certain value ranges are met. This considered a heuristic classification approach [26].

CLIPS provides several approaches to deal with situations when multiple rules will be activated simultaneously and a prioritisation is needed. All rules can be provided with a integer salience value, allowing the rule with the highest salience value to fire first. Alternatively, a CLIPS knowledge-base can be partitioned into thematic modules. The modules are put in a sequence on an execution stack, allowing all rules from the top-most module to fire. Once all rules from this module have been evaluated, processing moves on to the following module [7][8][9].

The rise of the Java programming language led to implementations of languages similar to CLIPS in projects such as JESS, DROOLS and JRules, adopting a similar syntax [2][4]. This family of rule-based and data-driven tools still shares the same basic syntax for the definition of rules to encode human knowledge. While the individual features and capabilities have diverged, it is still possible to port an application if a restricted subset of features is used to write portable programs. As a side effect, a wealth of documentation can be used from these CLIPS-related projects such as g.infer [2][4][6][25].

4. Rete

The core of the Production Rule System CLIPS is an Inference Engine that is able to handle a large number of rules and facts using the Rete algorithm for forward-chaining inference. The word „Rete“ is Latin, meaning „net“ or „comb“. The Rete algorithm was designed by Dr Charles L. Forgie [3]. Rete has become the basis for many popular rule engines and expert system shells apart from CLIPS. It provides a generalized logical description of an implementation of functionality, which is responsible for matching data (facts) against rules (production) in a pattern-matching production system. A production (rule) consists of one or more conditions and a set of actions which may be undertaken for each complete set of facts that match the conditions.

4.1. Pattern-Matching Performance

The efficiency of the Rete pattern-matching algorithm is based on the assumption that data changes slowly over time. This assumption will fail for applications where rapid data change can occur as it is often the case in GIS. Because of this, g.infer should not be perceived as a replacement of optimized GRASS tools such as r.mapcalc, which manipulate each cell of a raster layer [8]. However, in many cases data pre-processing will allow to comply with the Rete assumption. Such approaches include the grouping of data into larger sets, limiting rule-based intervals to elements which transition into another set, or to convert numeric value ranges into symbolic values like “unavailable, nominal, critical”, retracting the current fact and assert a new one only if the symbolic value has changed. For g.infer, such pre-processing can be achieved by GRASS modules such as r.clump, r.mask, r.reclass or r.recode.

5. Embedding CLIPS in GRASS

CLIPS and GRASS are based on the C language. Until now, no close-coupling below the API based on the C API has been published. Of the numerous software projects which embed CLIPS in other programming languages, two are currently known to have been used to connect CLIPS and GRASS GIS.

The **CLIPS And Perl with Extensions (CAPE)** project was developed in the late 1990s [11]. CAPE closely integrates CLIPS and the procedural programming language Perl, and provides extensions to facilitate building systems with an intimate mixture of the two [12].

The GRASSCAPE libraries were a merger of CAPE and GRASS5 to provide rule based programming in a GRASS environment [19]. This was used to assess the validity of radar meteorological data products, to generate warning messages for the general public for storm events and to create rainfall intensity maps for soil erosion studies. The development of GRASSCAPE was stopped in 2003.

PyCLIPS is an extension module for the Python language, interfacing it with CLIPS [5]. Python has become increasingly popular for scripting in GRASS GIS, and was selected as the reference language for GRASS 7.0 extensions: A reimplementation and extension of GRASS-CAPE based on PyCLIPS was started in 2011. This development eventually resulted in the GRASS module g.infer.

5.1. The GRASS GIS Module g.infer

g.infer is a GRASS add-on module for GRASS6.4.x and GRASS 7.0. It allows one to define and execute rule-based data-driven processing based on GRASS data layers. Currently, raster-, volume- and point vector-layers can be processed with g.infer. In addition, GRASS environment variables, including the GRASS location region settings can be queried and manipulated within the g.infer Production System.

Access to various parameters of the embedded CLIPS instance is provided by options and flags accessible both from the Command Line Interface or a GRASS Graphical User Interface (GUI). In addition to automated inference runs, an interactive mode allows to interact with the rule-base environment on the fly. The g.infer module further allows access to GRASS modules and their output. This makes the set-up of rule-driven GIS workflows possible. The development of g.infer is currently supported by GISIX.com to develop sample applications and create performance metrics. The module will be released as a GRASS add-on in late 2012.

GIS-based Inference Workflow This section describes the tasks executed by g.infer to allow the processing of GIS-layers by rule-based inference in the CLIPS environment. The involved tasks are best described when using both GRASS- and CLIPS- centered perspectives.

GRASS GIS-centered Workflow From the perspective of a GRASS GIS user, g.infer provides access to a Production Rule System to set-up and maintain specific rule-based data-driven tasks in GRASS as an Expert System. The following steps are required to implement such an Expert System and to conduct a rule-based analysis of spatial data layers:

In the preparation phase, the goal of the analysis must be defined. The knowledge and expertise needed to reach this goal will be written down as plain language rules (Knowledge Engineering). This will likely require the involvement of human domain experts. The plain language rules are in a next step translated into CLIPS rule syntax to be stored in a rule-base file. In this step, the names of the GRASS layers to be queried must be used in the CLIPS rules. The standard naming convention for how to address GRASS layers in CLIPS rules is provided in the g.infer html documentation [21].

In the application phase, g.infer is invoked with the required parameters: The provided GRASS layers and the rulebase-file are ingested. The user can opt to have elements of the rule- or fact-base printed out, or to interact with the inference environment via a command line prompt. Now the inference run will commence unless the early abort flag has not been set. A successful inference run leads to the update of selected GIS data layers, and optionally the creation of new vector layers and log files.

Production System-centered Workflow From the perspective of the CLIPS Production System, it is operating in g.infer in an embedded environment, which provides inference-related parameters and data. The overall CLIPS-based process begins by the setting of Inference Engine-related parameters and the definition of fact-templates for each GIS layer to be used. In turn, the content of the GIS layers is copied into facts for the inference process, and the rule-base is set up from the provided rule-file. At this stage interactive access via a CLIPS prompt can allow one to list and manipulate the current content. If no abort signal is given by the user from the GRASS layer, the inference process starts.

The existing rules are pattern-matched against the existing facts to start the firing of the first rules, resulting in modifications and extensions of the fact-base. This can lead to renewed firing of rules, starting an iterative process. Once the firing of rules ceases, the inference process ends. For the rule-based system, the final updating of GIS layers from parts of the fact-base is transparent.

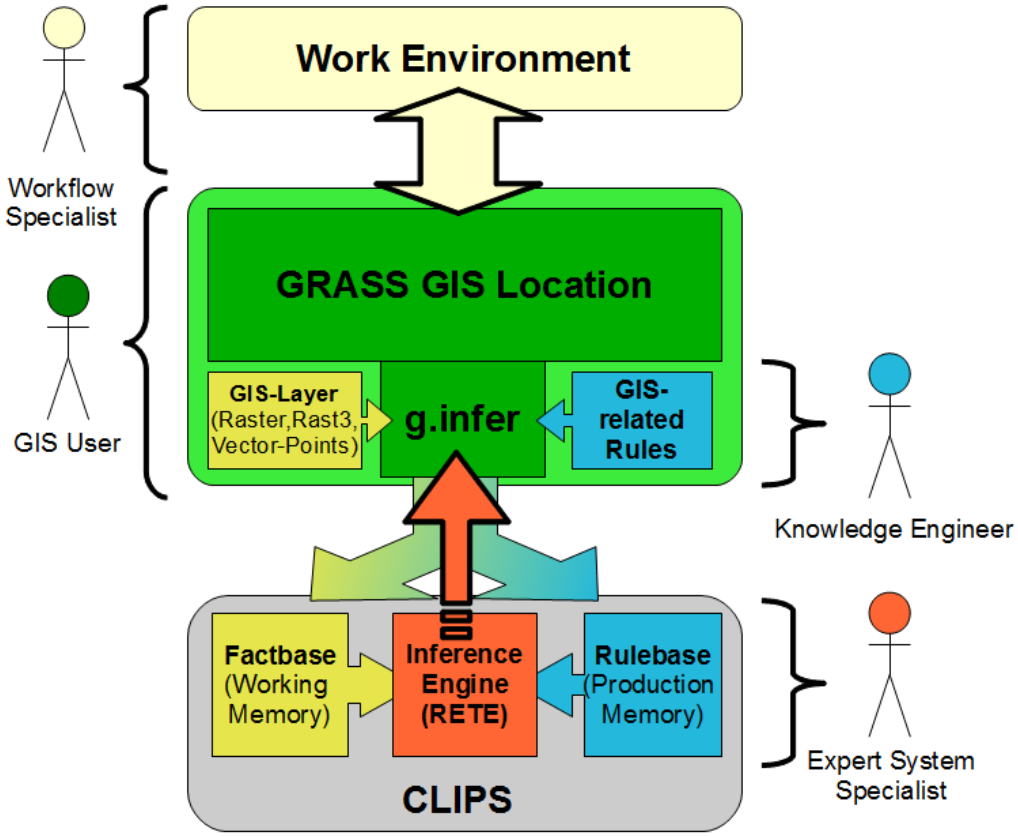

Figure 1: Interaction of the g.infer module with related GRASS GIS components and the CLIPS Production Rule System. Human experts (shown as figurines) can interact with this work environment independently on multiple levels.

Knowledge Engineering and Management g.infer provides multiple levels of knowledge modelling and rule-programming approaches, allowing one to use increasingly complex techniques when required. The options range from closely-coupled to loosely-coupled inference rules, extensions of the CLIPS language by functions, rulebase stratification by salience values or its partitioning into modules, up to ontologies and object oriented modelling using the COOL language. Depending on the techniques used for knowledge engineering, the evaluation and testing of a rule base can become a complex task. Multiple factors are to be considered, ranging from simple typos to content issues of input data layers and nontrivial rule precedence issues. This requires options for stepwise interactive execution and checking the inference process, the capability to log specific processes for later analysis and to save all or parts of the rule base or the object instances. Such features exist in the CLIPS environment and can be accessed by GIS users as GRASS module flags and -options in g.infer.

6. Business Processes in GRASS GIS

Software projects such a Jess and Drools have been used in the past years to apply the rule-based data-driven approach to new tasks, beyond the classical field of classification topics [2][4].

They are used to set up business-themed systems, in a vertical stack of tasks, including knowledge engineering, management and deployment of rules, collaboration between rule-based systems, analysis and end user tools. Software systems to handle such task stacks are called Business Rule Management Systems.

The emerging methodology of describing the application of rule-based systems in enterprise environments for structured, product-generating activities has been named the Business Rules Approach [2][15]. This is also of interest for GIS application and the related workflow perspective: GRASS GIS modules can be used in a similar fashion to set-up data-processing workflows, while on a higher level, GRASS GIS-based workflows can be fully integrated into greater workflow-chains.

Inference processes in g.infer can trigger the modifications of its fact-base, but they can also be used to execute further GRASS modules or scripted GRASS-workflows. By doing this, it is possible for an inference process to have new data imported into GRASS, have it processed and have the outcome to extend its fact-base. Further, direct queries to the user requesting interactive input can be triggered by rules. This allows one to effectively have a data-processing workflow in GRASS GIS controlled by an inference process within g.infer. The overall process is illustrated in Figure 2. The topic of interaction between COOL object instances and the rule-base will be covered in a later publication.

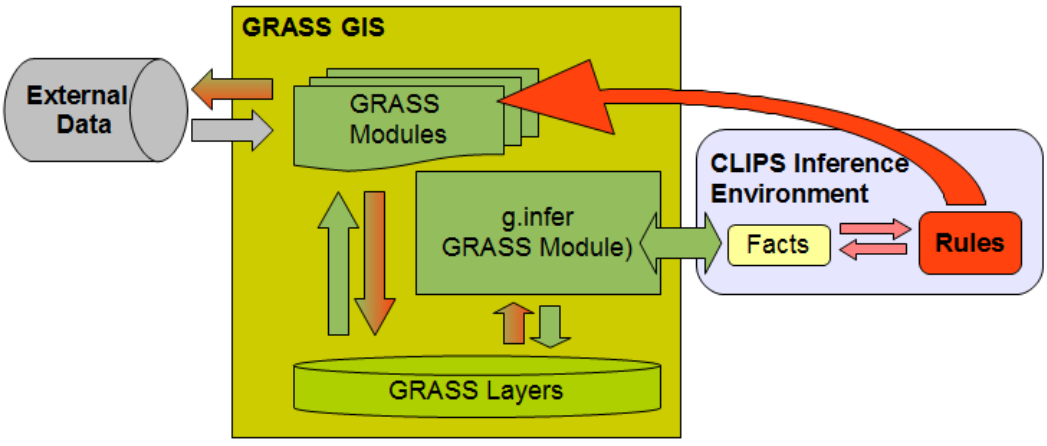

Figure 2: Overview of the interactions between the GRASS GIS environment (green), the CLIPS-based inference environment (blue) and external data sources (grey), highlighting (red) the potential for workflow control to be exerted by the inference process.

7. Application Scenarios

The forerunner modules of g.infer, r.[b]infer and GRASSCAPE have already enabled rule-based inference for GRASS GIS, yet did not succeed to attract a large audience of GIS application developers in the long term. As a consequence, their porting to the current versions of GRASS 6.x and GRASS 7 did not occur for lack of serious need. The same challenge applies for g.infer to give today's community of GRASS application developers significant added value in their work. For this reason, a cookbook-type publication will follow up this introductory article, providing hands on application examples with respective rule bases to lower the learning curve, and to show inference-based applications on varying levels of complexity.

As g.infer connects the domains of GIS, workflows, classification, and rule-based systems, the module can be applied for at least three different application scenarios in general:

1. **GIS-based classification tasks**, where g.infer can be used to quickly set up and apply rule-based data-driven classification of varying levels of complexity. This includes the combination of queries on information from both raster- and vector-layers and derived facts (e.g.: „IF a location is both an archeological site [vector information] and an abandoned mine [raster information] THEN assert geoarcheological monument“). Classification rules can be flexibly chained within the rulebase: „IF a location is an abandoned mine THEN assert no-trespassing“; „IF the location is a monument AND today-is-national monument day THEN assert guided-tours-available“

2. **Workflows and business processes**, where GIS-based data processing chains need to adapt to changing data quality, human user input or time constraints. This allows to use rules similar to the exception-construct known from other programming languages, such as Python: IF “the current GRASS region is smaller than threshold X” THEN “assert no-use-satellite-images; use-aerial-photography”.

3. **Extensions for the underlying production system**: In this case, the perception is reversed. From the standpoint of the embedded Production System, the g.infer-provided interface to GRASS GIS is merely an extension for advanced A.I.-focused tasks such as ontology modelling or intelligent software agents: While g.infer rules define knowledge about cause-effect actions among certain entities (facts), an ontology is a structural framework defining the object classes of the entities for the current knowledge domain and their properties and relations (taxonomy). Within g.infer, a simple implicit ontology for the GRASS GIS domain is used, to translate GRASS layers in corresponding domain objects which are defined via CLIPS templates. So the CLIPS templates and their interrelation define the ontology. Any g.infer application can set-up additional ontologies for their specific knowledge domain. Another field for exploration are intelligent rule-based software agents implemented in g.infer. Such autonomous entities are capable to observe and manipulate their surroundings while trying to achieve goals, effectively interacting with the GIS layers by manipulating the corresponding spatially-enabled facts in g.infer. They can be distributed in GIS-geographical space if they posses spatial locations (e.g. virtual sensor networks) and can become mobile if they have also the means to change their current position within the GRASS location. Another option are multi-agent systems (MAS), which are able to communicate to achieve a common goal.

8. Conclusion

The new add-on module g.infer re-introduces generic rule-based data-driven modelling to GRASS GIS for the current versions of GRASS 6.4.x and GRASS 7. It provides a new flexible interface between GRASS GIS-based geoscientific modelling and Artificial Intelligence (A.I.) research based on the C-Language Integrated Production System (CLIPS) toolkit. This allows to develop rule-based data-driven processing of GRASS GIS data sources by Expert Systems encoded as CLIPS knowledge bases.

The description of the theoretical and developmental background of the g.infer module already brings up possible application scenarios, ranging from the rule-driven formulation of hard to describe GIS-classification tasks, the flexible set-up and management of GIS workflows to Artificial Intelligence-focused topics such as the ontology management, defining taxonomies of knowledge domains, and the exploration of intelligent software agents encoded as g.infer rulebases.

Detailed examples of the practical application of g.infer for a range of real world problems will be provided in a follow up publication.

References

[1] Buehler K. (1990). A GIS providing grounds for water resources research

http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1188&context=watertech

[2] Buehler K. (1999). r.binfer Documentation

https://svn.osgeo.org/grass/grass/branches/releasebranch_5_5/html/html/r.binfer.html

[3] Browne P. (2009) JBOSS Drools Business Rules. Packt Publishing. ISBN 1-847-19606-3

[4] Forgy C., (1982) Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, 19

[5] Friedman-Hill E. (2003). Jess in Action. Manning Publications. ISBN 1-930-11089-8

[6] Garosi F. (2008). PyCLIPS Manual Release 1.0.

http://sourceforge.net/projects/pyclips/files/pyclips/pyclips-1.0/pyclips-1.0.7.348.pdf/download

[7] Giarratano J., Gary R. (2004). Expert Systems: Principles and Programming. Course Technology. ISBN 0-534-38447-1

[8] Giarratano, J.C. (2007). CLIPS User's Guide.

http://clipsrules.sourceforge.net/documentation/v630/ug.pdf

[9] Giarratano, J.C. (2007). CLIPS Reference Manual: Basic Programming Guide.

http://clipsrules.sourceforge.net/documentation/v630/bpg.pdf

[10] Giarratano, J.C. (2008). CLIPS Reference Manual: Advanced Programming Guide.

http://clipsrules.sourceforge.net/documentation/v630/apg.pdf

[11] Graham P. (1995). ANSI Common Lisp. Prentice Hall, ISBN 0-133-79875-6

[12] Inder R. (1998) CAPE Users Manual, ETLTechnical Report ETL-TR98-3, Electrotechnical Laboratory, Tsukuba, Japan.

[13] Inder R. (2000) CAPE: Extending CLIPS for the internet, Knowledge-Based Systems 13 (2000), Elsevier

[14] Jasiewicz J. (2011): r.fuzzy GRASS Addons Repository

http://trac.osgeo.org/grass/browser/grass-addons/grass6/raster/r.fuzzy

[15] Jasiewicz J., Di Leo M. (2012): Application of GRASS fuzzy inference system in flood prone areas prediction

http://geoinformatics.fsv.cvut.cz/gwiki/Application_of_GRASS_fuzzy_inference_system_in_flood_prone_areas_prediction

[16] JBOSS Community Documentation(2008) The Rule Engine

http://docs.jboss.org/drools/release/5.4.0.CR1/drools-expert-docs/html/ch01.html

[17] Lake M. W. (2000). MAGICAL computer simulation of Mesolithic foraging. In Kohler, T. A. and Gumerman, G. J., editors, *Dynamics in Human and Primate Societies: Agent-Based Modelling of Social and Spatial Processes*. Oxford University Press, New York.

[18] Lake M. W. (2000) MAGICAL computer simulation of Mesolithic foraging on Islay. In Mithen, S. J., editor, *Hunter-Gatherer Landscape Archaeology: The Southern Hebrides Mesolithic Project, 1988-98, volume 2: Archaeological Fieldwork on Colonsay, Computer Modelling, Experimental Archaeology, and Final Interpretations*. The McDonald Institute for Archaeological Research, Cambridge.

[19] Lake M. W. (2002) Magical for GRASS4.x

http://www.ucl.ac.uk/~tcrnmar/simulation/magical/manual/index.html

[20] Löwe P. (2004). Technical Note - A Spatial Decision Support System for Radar-Metereology in South Africa. Transactions in GIS. 8(2), Blackwell Publishing Ltd, Oxford.

[21] Löwe P. (2005). Knowledge Management and GRASS GIS: r.infer, GRASS-Newsletter 01/2005, ISSN 1614-8746

[22] Löwe P. (2012) g.infer Documentation (2012):

http://grasslab.gisix.com/scripts/g.infer/g.infer.html

[23] Lustenberger M (2012) r.agent GRASS Addons Repository

http://trac.osgeo.org/grass/browser/grass-addons/grass7/raster/r.agent

[24] Martin M., Westervelt J. (1991).GRASS4.0 Inference Engine: r.infer

http://grass.osgeo.org/gdp/raster/infer.ps.gz

[25] Netzel P (2011) Implementation of ANN in GRASS – an example of using ANN for spatial interpolation

http://www.wgug.org/images/stories/materialy/20110519praga-ann.pdf

[26] Jackson P. (1998). Introduction to Expert Systems. Addison Wesley. ISBN 0-201-87686-8

[27] Puppe F. (1993). Systematic Introduction to Expert Systems. Springer. ISBN 3-540-56255-9

[28] Riley G., (2008). The History of CLIPS.

http://clipsrules.sourceforge.net/WhatIsCLIPS.html#History

[29] Rudolph G. (2008). Some Guidelines For Deciding Whether To Use a Rule Engine.

http://www.jessrules.com/guidelines.shtml