Graphical Representation of Defeasible Logic Rules Using Digraphs

Efstratios Kontopoulos and Nick Bassiliades

Department of Informatics, Aristotle University of Thessaloniki,
GR-54124 Thessaloniki, Greece
{skontopo, nbassili}@csd.auth.gr

Abstract. Defeasible reasoning is a rule-based approach for efficient reasoning with incomplete and conflicting information. Nevertheless, it is based on solid mathematical formulations and is not fully comprehensible by end users, who often need graphical trace and explanation mechanisms for the derived conclusions. Directed graphs (or digraphs) can assist in this affair, but their applicability is balanced by the fact that it is difficult to associate data of a variety of types with the nodes and the connections in the graph. In this paper we try to utilize digraphs in the graphical representation of defeasible rules, by exploiting their expressiveness, but also trying to counter their major disadvantage, by defining multiple node and connection types.

1 Introduction

Defeasible reasoning [2] constitutes a simple rule-based approach to reasoning with incomplete and conflicting information. It can represent facts, rules, as well as priorities and conflicts among rules. Such conflicts arise, among others, from rules with exceptions (e.g. policies and business rules) and priority information is often available to resolve conflicts among rules. However, although defeasible reasoning features a significant degree of expressiveness and intuitiveness, it is still based on solid mathematical formulations, which, in many cases, may seem too complicated. So, end users might often consider the conclusion of a defeasible logic theory incomprehensible. A graphical trace and an explanation mechanism would certainly be very beneficial.

Directed graphs (or digraphs) are a special case of graphs that constitute a powerful and convenient way of representing relationships between entities [4]. In a digraph, entities are represented as nodes and relationships as directed lines or arrows that connect the nodes. The orientation of the arrows follows the flow of information in the digraph [5]. Digraphs offer a number of advantages to information visualization, with the most important of them being: (a) comprehensibility - the information that a digraph contains can be easily and accurately understood by humans [8] and (b) expressiveness - digraph topology bears non-trivial information [4]. Furthermore, in the case of graphical representation of logic rules, digraphs seem to be extremely appropriate. They can offer explanation of derived conclusions, since the series of inference steps in the graph can be easily detected and retraced [1]. Also, by going backwards from the conclusion to the triggering conditions, one can validate the truth of the inference result, gaining a means of proof visualization and validation. Finally,

G. Antoniou et al. (Eds.): SETN 2006, LNAI 3955, pp. 529 – 533, 2006.

especially in the case of defeasible logic rules, the notion of direction can also assist in graphical representations of rule attacks, superiorities etc.

There is, however, one major disadvantage, not only of digraphs but of graphs in general. More specifically, it is difficult to associate data of a variety of types with the nodes and with the connections between the nodes in the graph [4].

In this paper we attempt to exploit the expressiveness and comprehensibility of directed graphs, as well as their suitability for rule representation, but also try to leverage their disadvantages, by adopting an "enhanced" digraph approach.

There exist systems that implement rule representation/visualization with graphs, such as Graphviz [6], although we haven't come across a system that represents defeasible logic rules yet. Certain knowledge-based system development tools also feature rule and execution graph-drawing. Finally, there have been attempts of creating rule graphs for certain rule types, like association rules [3] or production rules [7], but they remained at an elementary stage of development.

2 Representing Rules with Digraphs

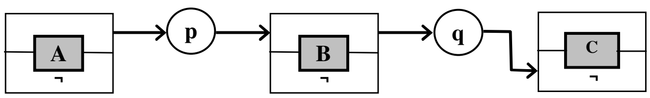
In an attempt to leverage the inability of directed graphs to use a variety of distinct entity types, the digraphs in our approach will contain two kinds of nodes, similarly to the methodology followed by [7]. The two node types will be:

- literals, represented by rectangles, which we call "*literal boxes*"
- rules, represented by circles

Thus, according to this principle, the following rule base:

```
p: if A then B          q: if B then ¬C
```

can be represented by the directed graph:



Each literal box consists of two adjacent "*atomic formula boxes*", with the upper one of them representing a positive atomic formula and the lower one representing a negated atomic formula. This way, the atomic formulas are depicted together clearly and separately, maintaining their independence.

```
r: if ¬A and B then C
```

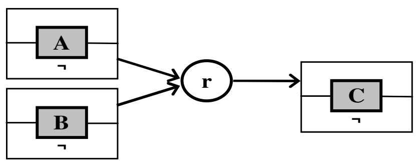


Fig. 1. Digraph featuring a conjunction

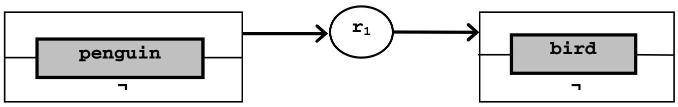
If the rule body consists of a conjunction of literals the representation is not profoundly affected, as illustrated in Fig. 1. As can be observed, digraphs, "enhanced" with the addition of distinct node types, offer a significant level of expressiveness in representing rules. The next step is to use directed graphs in the representation of defeasible logic rules, which are more demanding in representational capabilities.

3 Defeasible Logics and Digraphs

A *defeasible theory D* (i.e. a knowledge base or a program in defeasible logic) consists of three basic ingredients: a set of facts (F), a set of rules (R) and a superiority relationship (>). Therefore, D can be represented by the triple (F, R, >).

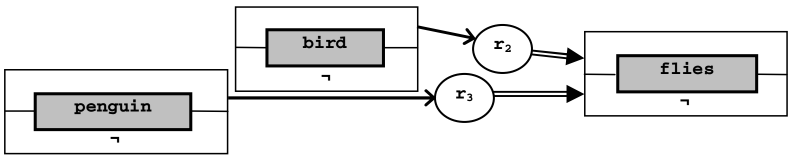
In defeasible logic, there are three distinct types of rules: strict rules, defeasible rules and defeaters. In our approach, each one of the three rule types will be mapped to one of three distinct connection types (i.e. arrows), so that rules of different types can be represented clearly and distinctively.

The first rule type in defeasible reasoning is *strict rules*, which are denoted by $A \rightarrow p$ and are interpreted in the typical sense: whenever the premises are indisputable, then so is the conclusion. An example is: "*Penguins are birds*", which would become: `r1: penguin(X) → bird(X)`, and is represented by digraphs, as follows:

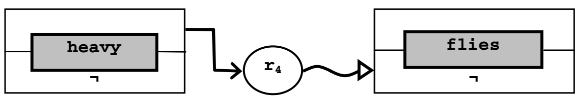


Notice that in the rule graph we only represent the predicate and not the literal (i.e. predicate plus all the arguments) because we are mainly interested in emphasizing the interrelationships between the concepts (through the rules) and not the complete details of the defeasible theory.

Contrary to strict rules, *defeasible rules* can be defeated by contrary evidence and are denoted by $A \Rightarrow p$. Examples of defeasible rules are `r2: bird(X) ⇒ flies(X)`, which reads as: "*Birds typically fly*" and `r3: penguin(X) ⇒ ¬flies(X)`, namely: "*Penguins typically do not fly*". Rules `r2` and `r3` would be mapped to the following directed graphs, respectively:

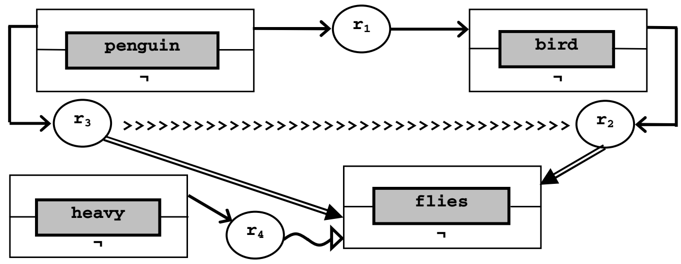


Defeaters, denoted by *A ~> p*, are rules that do not actively support conclusions, but can only prevent them, namely they can defeat some defeasible conclusions by producing evidence to the contrary. An example is: `r4: heavy(X) ~> ¬flies(X)`, which reads as: "*Heavy things cannot fly*". This defeater can defeat the (defeasible) rule `r2` mentioned above and it can be represented as:



Finally, the *superiority relationship* is an acyclic relation > that is used to resolve conflicts among rules. For example, given the defeasible rules r_2 and r_3, no conclusive decision can be made about whether a penguin can fly or not, because rules r_2 and r_3 contradict each other. But if the superiority relationship $r_3 > r_2$ is introduced, then r_3 overrides r_2 and we can indeed conclude that the penguin cannot fly. Rule r_3 is called *superior* to r_2. Thus, a fourth connection type is introduced and the aforementioned superiority relationship is represented as follows:

The set of rules (r_1 - r_4) mentioned in this section form a bigger, compact directed rule graph that can indeed raise the level of comprehensibility on behalf of the user.



4 Conclusions and Future Work

In this paper we argued that graphs can be a powerful tool in the field of information visualization. Especially in the case of rules, directed graphs can be particularly useful, since by definition they embrace the idea of information flow, a notion that is also encountered in rules and inference. Directed graphs present, however, a major disadvantage, which is their inability to associate data of a variety of types with the nodes and with the connections between the nodes. In this paper we propose an approach that aims at leveraging this disadvantage, by allowing different node and connection types. Digraphs, "enhanced" with these extra features, can greatly assist in representing defeasible logic rules.

In the future we plan to delve deeper into the proof layer of the Semantic Web architecture, by enhancing further the rule representation with rule execution tracing, explanation, proof exchange in an XML/RDF format, proof visualization and validation, etc. These facilities would be useful for increasing the user trust for the Semantic Web and for automating proof exchange and trust among agents in the Semantic Web.

References

[1] Antoniou G., Harmelen F. van, *A Semantic Web Primer*, MIT Press, 2004.
[2] Antoniou G., *Nonmonotonic Reasoning*, MIT Press, 1997.
[3] Chakravarthy S., Zhang H., "Visualization of association rules over relational DBMSs", *Proc. 2003 ACM Symp. on Applied Computing*, ACM Press, pp. 922-926, 2003.

[4] Clarke D., "An Augmented Directed Graph Base for Application Development", *Proc. 20th annual Southeast regional Conf.*, ACM Press, pp. 155-159, 1982.
[5] Diestel R., *Graph Theory (Graduate Texts in Mathematics)*, 2nd ed., Springer, 2000.
[6] Graphviz - Graph Visualization Software, http://www.graphviz.org.
[7] Jantzen J., "Inference Planning Using Digraphs and Boolean Arrays", Proc. Int. Conf. on APL, ACM Press, pp. 200-204, New York, USA, 1989.
[8] Nascimento H.A.D. do, "A Framework for Human-Computer Interaction in Directed Graph Drawing", *Proc. Australian Symp. on Information Visualization*, pp. 63-69, 2001.