Combining Rules and Ontologies with Carneades

Thomas F. Gordon[1]

Fraunhofer FOKUS, Berlin
thomas.gordon@fokus.fraunhofer.de

Abstract. The Carneades software system provides support for constructing, evaluating and visualizing arguments, using formal representations of facts, concepts, defeasible rules and argumentation schemes. This paper illustrates how rules and ontologies can be combined in Carneades with a prototype legal application for analyzing open source software license compatibility issues in particular cases.

Introduction

This paper illustrates how rules and ontologies can be used in combination in the Carneades argumentation system [14] with examples from a prototype legal application for analyzing open source license compatibility issues [15]. As Bing [3], Fiedler [7], McCarty [17] and many others have noted, legal argumentation is not primarily deductive, but rather a modeling process of shaping an understanding of the facts, based on evidence, and an interpretation of the legal sources, to construct a theory for some legal conclusion [3]. The parties in a legal dispute construct competing theories and argue about their relative merits. Carneades is designed to support all the steps in this process of theory construction, argumentation and evaluation.[1]

The Carneades software system is based on a well-founded formal model of structured argumentation with support for proof burdens and standards [9,10], now called Carneades Argument Evaluation Structures (CAES). It has been formally proven that the Carneades model of argument is a specialization of both Prakken's ASPIC+ model of structured argumentation [8] and Brewka's Abstract Dialectical Frameworks [5] and thus an instantiation of Dung's Abstract Argumentation Framework [4,8]. Carneades has also been shown by Governatori to be closely related to Defeasible Logic [16]. A formal model of abduction in Carneades argument evaluation structures has been developed [2], which is useful for identifying relevant issues and computing minimal sets of statements, called positions, which, if proven, would make some goal statement acceptable (in) or not acceptable (out) in a stage of a dialogue.

Building on this formal foundation, the Carneades software provides a number of tools for interactively constructing, evaluating and visualizing arguments, as well as computing positions. Arguments are constructed using formalizations of facts, concepts, defeasible rules and argumentation schemes [11,12]. Facts and concepts are represented using the Web Ontology Language (OWL), an XML schema for description logic [1], a subset of first-order logic and thus with a monotonic (strict) entailment relation. Legal rules and argumentation schemes [18] are both modeled as defeasible inference rules, represented in the Legal Knowledge Interchange Format (LKIF) [6]. The rules of alternative, competing theories of the law can be included in a single model.

A combination of forwards and backwards reasoning is used to construct arguments: a description logic reasoner constructs the deductive closure of the concepts and facts in a forwards manner; the Carneades rules engine uses backwards reasoning to apply the defeasible inference rules in a goal-directed and stratified way to the deductive closure of the description logic theory of facts and concepts. The LKIF rule language has been extended to provide a way to declare the domain of variables using predicates defined in OWL, similar to the way variables are typed in programming languages. These domain declarations provide important control information that enables the rule engine to iterate over instances of the domains to more efficiently instantiate the rules.

[1] http://carneades.github.com

In addition to the arguments constructed automatically from a knowledge-base of facts, concepts and rules, arguments can manually entered into the system by the user. These arguments can be completely ad hoc or instantiations of argumentation schemes. The Carneades system currently includes a library of about 20 of Walton's most important argumentation schemes along with a software assistant which steps the user through the process of selecting and instantiating schemes.

As the arguments are constructed and edited, they are visualized in an argument map [13]. The graphical user interface, called the Carneades Editor, supports argument evaluation by providing tools to accept and reject statements, assign proof standards and weigh arguments. After every modification, the underlying computational model of argument is used to update and visualize the acceptability status of statements in the map. The differential legal effects of competing theories can be analyzed by assuming their rules to be valid and then checking how this effects the acceptability of issues of interest in the argument map. Moreover Carneades provides a *find positions* assistant which can be used to abduce theories with desired legal effects.

The rest of this paper show how ontologies and rules can be used in combination in Carneades with examples from the prototype legal application for analyzing open source license compatibility issues. We start with examples from a simple OWL ontology for describing software licenses and use and derivation relationships between works of software. Next we show how to use the ontology to model the facts of a case. We then show how to model some rules of copyright law in LKIF, focusing on the issue of whether linking to a software library produces a derivative work.

Concepts and Facts

Carneades uses the Web Ontology Language (OWL), a World Wide Web standard XML schema for representing and interchanging description logic knowledge bases. These knowledge bases have two parts, for concepts (TBox) and facts (ABox). The top-level concepts, called *classes* in OWL, for our application are CopyrightLicense, CopyrightLicenseTemplate, LegalEntity, LicenseTerm and Work. The Work class is for all works protectable by copyright. There is a SoftwareEntity subclass of Work, intended to cover all kinds of software artifacts.

The main property of software entities of interest for license compatibility issues is the isDerivedFrom property, expressing that one entity has been derived from another. The ontology includes properties for representing various ways that software can use other software, such compiledBy and linksTo.

The software ontology was used to model an example software project, roughly based on the current version of the Carneades system.

Rules

Description logic (DL) is semantically a decidable subset of first-order logic. This means that the inferences of description logic reasoners are *strict*: if the axioms of a DL knowledge base are true in some domain, then all of the inferences made by a (correctly implemented) DL reasoner are necessarily also true, without exception. While DL is very powerful and useful, monotonic logics are not sufficient for modeling legal rules, such as the rules of copyright law, in a maintainable and verifiable way, isomorphic with the structure of legislation and regulations. Legislation is typically organized as general rules subject to exceptions. Arguments made by applying legal rules are *defeasible*. Their conclusions can be defeated with better counterarguments. Various legal rules may conflict with each other. Theses conflicts are resolved using legal principals about priority relationships between rules, such as the principal of *lex superior*, which gives rules from a higher authority, such as federal law, priority over rules from a lower authority, such as state law. These properties of legal rules are well known in AI and Law and have been studied extensively. References are omitted for lack of space.

Thus we model legal rules using a defeasible rule language which has been developed especially for this purpose, as part of the Legal Knowledge Interchange Format (LKIF), and use description logic (OWL more specifically) for more limited purposes: 1) to declare the language of unary and binary predicate symbols and 2) to make assertions about these predicates, using DL axioms, which are judged to be universally true and beyond dispute in the domain.

Here we illustrate the LKIF rule language by modeling two interpretations of the concept of a derivative work in copyright law. We begin with the general rule that the copyright owner of software may license the software using any license template he chooses.

```
<rule id="DefaultLicenseRule">
    <head>
        <s pred="&oss;mayUseLicenseTemplate">
            <v>SE</v> may be licensed using
            the <v>T</v> template
        </s>
    </head>
</rule>
```

Since LKIF is an XML schema, rules are represented in XML. This particular rule has a head (conclusion) but no body (conditions). Even though the rule has no conditions, inferences made using this rule are not necessarily or universally true, but remain defeasible. We will make use of this feature to express exceptions to this general rule below.

The rule has been assigned an identifier, DefaultLicenseRule, which may be used to formulate statements about the rule. That is, rules are reified and may be reasoned about just like other objects.

The predicate symbol of the statement (proposition) in the head of the rule is specified using the *pred* attribute. Its value can be the name of a class or property in a OWL ontology, as in this example. The ? entity reference refers to the ontology, using its URI.

Declaring predicate symbols in ontologies makes it possible to divide up the model of a complex domain theory into modules, with a separate LKIF file for each module. OWL provides a way to import the classes and properties of other OWL files, recursively. Similarly, LKIF provides a way to import both LKIF and OWL files. OWL makes it easy to manage predicate symbols across the boundaries of modules and to make sure that symbols in different modules refer to the same class or property when this is desired.

The XML syntax for rules in LKIF is rather verbose and not especially readable. Fortunately, it is easy to write programs for converting XML into more readable formats. Moreover, XML structure editors exist which use style sheets to enable authors to edit XML documents directly in a more readable form. Using this feature, the above rule can be displayed in the editor as follows:

```
rule DefaultLicenseRule
    head SE may be licensed using the T template
```

We will use this more readable format for displaying LKIF rules in the remainder of this article. Next let us formulate an exception to the general rule that any license template may be used for reciprocal licenses:

```
rule ReciprocityRule
    head
        not: SE1 may be licensed using the T1 template
    domains
        SE1 uses SE2
        SE2 has license L
    body
        L is reciprocal
        SE1 is derived from SE2
        unless exists T2 : L is an instance of template T2
            such that T1 is compatible with T2
```

This reciprocity rule states that a software entity, SE1, may not be licensed using a license template, T1, if the software is derived from another software entity, SE2, licensed using a reciprocal license, L, unless L is an instance of a template license, T2, which is compatible with T1. The use of domains in this rule provides control information to make use of forward chaining in the description logic reasoner, as discussed in the introduction. Notice that the conclusion of the rule is negated and that the last condition of the rule expresses a further exception, using an *unless* operator.

These two rules illustrate two kinds of exceptions. In argumentation terms, arguments constructed using the ReciprocityRule *rebut* arguments constructed using the DefaultRule and arguments which make use of the explicit exception of the ReciprocityRule, by showing that the licenses are compatible, *undercut* the reciprocity argument.

Let us end this brief overview with rules modeling two conflicting views about whether or not linking creates a derivative work. According to the lawyers of the Free Software Foundation, linking does create a derivate work. Lawrence Rosen, a legal expert on open source licensing issues, takes the opposing point of view and argues that linking per se is not sufficient to create derivate works.

```
rule FSFTheoryOfLinking
    head
        SE1 is derived from SE2
    body
        SE2 is a software library
        SE1 is linked to SE2
        The FSF theory of linking is valid

rule RosenTheoryOfLinking
    head
        not: SE1 is derived from SE2
    body
        SE2 is a software library
        SE1 is linked to SE2
        The Rosen theory of linking is valid
```

The last condition of each of these rules requires that the interpretation of copyright law represented by the rule is legally valid. Making this condition explicit enables us to argue about which theory of linking is correct, to compare the effects of these two theories on particular cases, and to use abduction to derive positions about which theory to prefer.

Conclusion

We have illustrated how ontologies and rules can be used together in the Carneades argumentation system with a prototype legal application for analyzing software licensing issues. To our knowledge, no other argumentation or rule-based system currently provides the combination of tools required for this application: 1) automatic argument construction from a knowledge base of strict and defeasible rules; 2) argument mapping; 3) argument evaluation; 4) interactive construction of arguments using argumentation schemes; 5) exploration of effects of alternative legal theories; and 6) computation of positions, using abduction. The source code of the application is freely available, as open source software.

The current user interface is a desktop application, written in Java using the Swing user interface library. Work is in progress on a web version of Carneades. The user interface of this web version is a Rich Internet Application (RIA) implemented using only World-Wide-Web Concortiums standards, in particular XML and Javascript (AJAX). Argument graphs in the web version are rendered using Scalable Vector Graphics (SVG), another W3C standard.

Acknowledgements

This work was partially funded by the European projects Qualipso (IST-FP6–034763) and IMPACT (IST-FP7–247228). Earlier version of this paper were presented at the Jurix 2010 workshop on Modeling Legal Cases and Legal Rules and the Thirteenth International Conference on Artificial Intelligence and Law (ICAIL 2011).

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., eds. *The Description Logic Handbook — Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. Ballnat, S. and Gordon, T.F. Goal Selection in Argumentation Processes — A Formal Model of Abduction in Argument Evaluation Structures. *Proceedings of the Third International Conference on Computational Models of Argument (COMMA)*, IOS Press (2010), 51–62.
3. Bing, J. Uncertainty, Decisions and Information Systems. In C. Ciampi, ed., *Artificial Intelligence and Legal Information Systems*. North-Holland, 1982.
4. Brewka, G., Dunne, P.E., and Woltran, S. Relating the Semantics of Abstract Dialectical Frameworks and Standard AFs. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI–2011)*, (2011), in press.
5. Brewka, G. and Gordon, T.F. Carneades and Abstract Dialectical Frameworks: A Reconstruction. *Proceedings of the Third International Conference on Computational Models of Argument (COMMA)*, IOS Press (2010), 3–12.
6. ESTRELLA Project. The Legal Knowledge Interchange Format (LKIF). 2008.
7. Fiedler, H. Expert Systems as a Tool for Drafting Legal Decisions. In A.A. Martino and F.S. Natali, eds., *Logica, Informatica, Diritto*. Consiglio Nazionale delle Richere, Florence, 1985, 265–274.
8. Gijzel, B.V. and Prakken, H. Relating Carneades with abstract argumentation. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI–2011), (2011), in press.
9. Gordon, T.F., Prakken, H., and Walton, D. The Carneades Model of Argument and Burden of Proof. *Artificial Intelligence 171*, 10–11 (2007), 875–896.
10. Gordon, T.F. and Walton, D. Proof Burdens and Standards. In I. Rahwan and G. Simari, eds., *Argumentation in Artificial Intelligence*. Springer-Verlag, Berlin, Germany, 2009, 239–260.
11. Gordon, T.F. and Walton, D. Legal Reasoning with Argumentation Schemes. *12th International Conference on Artificial Intelligence and Law (ICAIL 2009)*, ACM Press (2009), 137–146.
12. Gordon, T.F. Constructing Arguments with a Computational Model of an Argumentation Scheme for Legal Rules – Interpreting Legal Rules as Reasoning Policies. *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, (2007), 117–121.

13. Gordon, T.F. Visualizing Carneades Argument Graphs. *Law, Probability and Risk 6*, 2007, 109–117.
14. Gordon, T.F. An Overview of the Carneades Argumentation Support System. In C.W. Tindale and C. Reed, eds., *Dialectics, Dialogue and Argumentation. An Examination of Douglas Walton[2BC?]s Theories of Reasoning.* College Publications, 2010, 145–156.
15. Gordon, T.F. Report on a Prototype Decision Support System for OSS License Compatibility Issues. 2010.
16. Governatori, G. On the Relationship between Carneades and Defeasible Logic. *Proceedings of the International Conference on Artificial Intelligence and Law (ICAIL–2011)*, (2011).
17. McCarty, L.T. Some Arguments About Legal Arguments. *International Conference on Artificial Intelligence and Law*, (1997), 215–224.
18. Walton, D., Reed, C., and Macagno, F. *Argumentation Schemes.* Cambridge University Press, 2008.