A System for Modal and Deontic Defeasible Reasoning

Grigoris Antoniou[1] and Nikos Dimaresis[1] and Guido Governatori[2]

[1] Institute of Computer Science, FO.R.T.H., Vassilika Voutwn
P.O. Box 1385, GR 71110, Heraklion, Greece
[2] School of ITEE, The University of Queensland, Australia

1 Introduction

The first source of motivation for our work is the modelling of multi-agent systems. In particular, we follow the approach of [1] that combines two perspectives: (a) a cognitive account of agents that specifies motivational attitudes, using the BDI architecture [2], and (b) modelling of agent societies by means of normative concepts [3].

Commonly, both motivational attitudes and normative aspects are logically captured through the use of modal logics which are, by definition, monotonic. Therefore, they cannot deal properly with inconsistencies, that may easily arise in multi-agent and web environments. As argued in [1], reasoning about intentions and other mental attitudes has *defeasible nature*, and defeasibility is a key aspect for normative reasoning.

In our work, we adopt the well-known defeasible logic [4]. It is a simple, rule-based and computationally cheap approach. The main objective of this paper is to extend defeasible logic with modal and deontic operators, and to report on an implementation.

The second important source for motivation for our work is the *modelling of policies*. Policies play crucial roles in enhancing security, privacy and usability of distributed services and extensive research has been done in this area, including the Semantic Web community. It encompasses the notions of security policies, trust management, action languages and business rules. As for modelling multi-agent systems, our formalism of choice is defeasible reasoning.

The expressive power of formal specification languages required by the business rules community is high and includes deontic notions like obligation, permission and prohibition. Thus, this task is compatible with the first aim to model multi-agent systems. Again, we will rely on deontic extensions of defeasible logic.

The two scenarios outlined above can be combined with the *semantic web* initiative [5], which aims at enhancing the current state of the web through the use of semantic information. Semantic web languages and technologies support the issue of *semantic interoperability*, which is important both for multi-agent systems and for policies. Our language of choice, defeasible logic, is compatible with developments in this area. Now that the layers of metadata (RDF) and ontology (OWL) are stable, an important focus is on rule languages for the semantic web. While initially the focus has been on monotonic rule systems [6,7,8], nonmonotonic rule systems are increasingly gaining attention [9,10,11]. In particular, there are implementations of defeasible logic that interoperate with semantic web standards [10,11].

As already stated, the aim of this paper is to propose modal and deontic extensions of defeasible logic, and to describe the basic characteristics of an implemented system.

M.A. Orgun and J. Thornton (eds)
20th Australian Joint Conference on Artificial Intelligence, AI 2007, pp. 1–5, 2007.
© Springer 2007.
The original publication is available at www.springerlink.com.

We base our implementation on the system DR-Prolog [11], which uses XSB [12] as the underlying logical engine.

2 Defeasible Logic

A *defeasible theory* is a triple (*F, R,* $>$), where F is a set of literals (called *facts*), R a finite set of rules, and $>$ a superiority relation on R.

There are three kinds of rules: *Strict rules* are denoted by $A \rightarrow p$, where A is a finite set of literals and p is a literal, and are interpreted in the classical sense: whenever the premises are indisputable (e.g. facts) then so is the conclusion. Defeasible rules are denoted by $A \Rightarrow p$, and can be defeated by contrary evidence. *Defeaters* are denoted by $A \rightsquigarrow p$ and are used to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary.

A superiority relation is an acyclic relation $>$ on R (that is, the transitive closure of $>$ is irreflexive).

A formal definition of the proof theory is found in [4].

3 Extension of Defeasible Logic with Modalities

Recent work [1] shows that Defeasible Logic is a nonmonotonic approach that can be extended with modal and deontic operators. This paper presents a computationally oriented nonmonotonic logical framework that deals with modalities. It combines two independent perspectives about cognitive agents, belief-desire-intention(BDI) agent architecture and agent models that are based on social and normative concepts. This approach has many similarities with the Beliefs-Obligations-Intentions-Desires architecture (BOID) [3]. In BOID conflicts may arise among informational and motivational attitudes. The way these conflicts are resolved determines the type of the agent.

The logical framework deals with the following modalities:

1. *knowledge* - the agent's theory about the world
2. *intention* - policy-based intentions, that is the agent's general policies
3. *agency* - agent's intentional actions
4. *obligation* - obligations from the agent's normative system

We consider an additional deontic modality, permission. This component represents what an agent is permitted to do, according to his normative system. We extend defeasible logic with the permission operator, in order to represent and reason with business rules and policies properly, in Semantic Web applications.

Defeasible Logic is the suitable formalism that can deal with these components. The reason being ease of implementation, flexibility and it is efficient. A rule-based nonmonotonic formalism was developed that extends defeasible logic and represents and reasons with these modal operators. It has as main feature the introduction of the mode for every rule, which determines the modality of rule's conclusion. It also supports modalised literals that can be defined in defeasible theories as facts or as part of the antecedents of rules.

4 Translation Into Logic Programs

We use the approach proposed in [13], [14], to perform reasoning over a defeasible theory. According to this, we translate a defeasible theory D into a logic program P(D), and we use a logic metaprogram that simulates the proof theory of the formalism that extents defeasible logic, to reason over the defeasible theory. In the following we describe a sample of the metaclauses used.

Cclauses for definite provability: a literal is strictly (or definitely) provable in knowledge, if it is a fact and strictly provable in other modalities, if the corresponding modal literal is a fact. A modalised literal is represented as prefixed with a modal operator (agency, intention, obligation or permission). An unmodalised literal belongs to the knowledge of the environment. A literal is also strictly provable in a modality, if it is supported by a strict rule, with the same mode and the premises of which are strictly provable. A definite provable literal in intention is defined as

```
strictly(P,intention):-fact(intention(P)).
```

The next clauses define defeasible provability: a literal is defeasibly provable in a modality, either if it is strictly provable in the same modality, or if the literal, for this modality, is consistent, it is supported by a rule, and there is not an undefeated applicable conflicting rule.

```
defeasibly(P,Operator):-strictly(P,Operator).
defeasibly(P,Operator):-consistent(P,Operator),supported(R,Operator,P),
negation(P,P1),not(undefeated_applicable(S,Operator,P1)).
```

A literal is consistent in a modality, if its complementary literal is not strictly provable in the same modality and in any of the attacking modalities. For example, in a strongly independent agent, a literal is consistent in intention, if its complementary is not strictly provable in intention, knowledge and agency:

```
consistent(P,intention):-negation(P,P1),not(strictly(P1,knowledge)),
not(strictly(P1,intention)),not(strictly(P1,agency)).
```

A literal is supported in a modality, if it is supported by a supportive rule with the same mode, the premises of which are defeasibly provable.

```
supported(R,Operator,P):-supportive_rule(R,Operator,P,A),defeasibly(A).
```

A rule is undefeated applicable in a modality, if it is a supportive rule or a defeater in a mode that attacks the modality, the premises of the rule are defeasibly provable, and it is not defeated neither by a supported literal in the modality nor by an applicable rule for the rule 's mode. For example, in a strongly independent type, a rule is undefeated applicable in agency, if it is a rule in knowledge which is not defeated neither by a supported literal in agency nor by an applicable rule in knowledge.

```
undefeated_applicable(S,agency,P):-rule(S,knowledge,P,A),defeasibly(A),
not(defeated_by_supported(S,agency,P)),not(defeated_by_applicable(S,knowledge,P)).
```

A literal, supported by a rule R, is defeated by a supported literal in a modality, if the latter is complementary, it is supported in the same modality by a rule S and S is superior to R.

```
defeated_by_supported(R,X,P):-negation(P,P1),supported(S,X,P1),
superior(S,R).
```

A literal, supported by a rule R, is defeated by an applicable rule in a modality, if this rule is conflicting to R, it is applicable in the same modality and superior to R.

```
defeated_by_applicable(R,X,P):-negation(P,P1),applicable(S,X,P1),
superior(S,R).
```

A rule is applicable in a modality, if it is a supportive rule or a defeater for an attacking modality and its premises are defeasibly provable; it is also applicable even it has a mode that can be converted to an attacking modality with the feature of rule conversion. For example, a rule is applicable in knowledge, if it is a rule in agency, the premises of which are defeasibly provable:

```
applicable(R,knowledge,P):-rule(R,agency,P,A),defeasibly(A).
```

5 Implementation

Our nonmonotonic rule-based system supports reasoning in defeasible logic, extended with the modalities. It provides automated decision support, when running a specific case with the given logic programs and ontological knowledge to get a correct answer.

An additional functionality that the system supports, is that it has the ability to treat RDF data as facts of the user's defeasible theories, in order to be processed by the organization's rules. The RDF/S documents are retrieved from the Web, and validated by the Semantic & Syntactic Validator, before being loaded to the system. This validator is an embedded version of VRP parser [15]. The system employs the SWI RDF parser to load the valid RDF/S data and translate them into RDF triples. The triples are then translated into Prolog facts and rules, which are passed to the Reasoning Engine.

The Reasoning Engine compiles the metaprogram, corresponding to the agent type we use, and the logic programs, representing the rules and contain the ontological knowledge. Then it evaluates the answers to user's queries expressed in Prolog syntax and applied to the compiled programs. The Reasoning Engine was implemented by using the Java programming library of InterProlog [16]. This is an open-source Java front-end that supports the Prolog engines of YAP [17], SWI and XSB. It provides access to Prolog engines over TCP/IP sockets and launches Prolog processes in the background, outside the Java Virtual Machine. Thus InterProlog provides the interface to pass logic programs and process queries to YAP and XSB, the engines that make the reasoning, and RDF/S data to SWI, which translates them in logic programs.

Finally the system provides a Graphical User Interface based on Java Foundation Classes (Swing). By interacting with the GUI, the user can import logic programs, load RDF/S ontologies and query the system.

6 Conclusion

We argued that defeasible reasoning is a computationally efficient way of dealing with issues related to the modelling of policies and multi-agent systems. We have described

how to enhance standard defeasible logic with agency, intention, permission and obligation operators, and briefly outlined an implemented system that is also compatible with semantic web technologies.

In future work, we intend to provide an experimental evaluation, experiment with other logical underlying engine, and develop realistic applications.

References

1. Governatori, G., Rotolo, A.: Defeasible Logic: Agency, Intention and Obligation. In: DEON. (2004) 114–128
2. Rao, A.S., Georgeff, M.P.: Modeling rational agents within a BDI-architecture. In Allen, J., Fikes, R., Sandewall, E., eds.: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91), Morgan Kaufmann publishers Inc.: San Mateo, CA, USA (1991) 473–484
3. Broersen, J., Dastani, M., Hulstijn, J., Huang, Z., van der Torre, L.: The BOID architecture: conflicts between beliefs, obligations, intentions and desires. In: AGENTS '01: Proceedings of the fifth international conference on Autonomous agents, New York, NY, USA, ACM Press (2001) 9–16
4. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM Trans. Comput. Logic **2**(2) (2001) 255–287
5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284**(5) (2001) 34–44
6. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: WWW. (2003) 48–57
7. Horrocks, I., Patel-Schneider, P.F.: A proposal for an OWL Rules Language. In: WWW '04: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, ACM Press (2004) 723–731
8. Rosati, R.: On the decidability and complexity of integrating ontologies and rules. WSJ **3**(1) (2005) 41–60
9. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: dlvhex: A System for Integrating Multiple Semantics in an Answer-Set Programming Framework. In: WLP. (2006) 206–210
10. Bassiliades, N., Antoniou, G., Vlahavas, I.P.: DR-DEVICE: A Defeasible Logic System for the Semantic Web. In: PPSWR. (2004) 134–148
11. Antoniou, G., Bikakis, A.: DR-Prolog: A System for Defeasible Reasoning with Rules and Ontologies on the Semantic Web. IEEE Transactions on Knowledge and Data Engineering **19**(2) (2007) 233–245
12. XSB - Logic Programming and Deductive Database System for Unix and Windows. http://xsb.sourceforge.net (2007)
13. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Embedding defeasible logic into logic programming. Theory Pract. Log. Program. **6**(6) (2006) 703–735
14. Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T.: Efficient defeasible reasoning systems. International Journal on Artificial Intelligence Tools **10**(4) (2001) 483–501
15. VRP - The ICS-FORTH Validating Rdf Parser. http://139.91.183.30:9090/RDF/VRP (2007)
16. InterProlog - a Prolog-Java interface. http://www.declarativa.com/interprolog (2007)
17. YAP Prolog. http://www.ncc.up.pt/ vsc/Yap (2007)