DAMN: Defeasible Reasoning Tool for Multi-Agent Reasoning

Abdelraouf Hecham,[1] **Madalina Croitoru,**[1] **Pierre Bisquert**[2]
[1]LIRMM, University Of Montpellier, France, {hecham, croitoru}@lirmm.fr
[2]IATE, INRA, Montpellier, France, pierre.bisquert@inra.fr

Abstract

This demonstration paper introduces DAMN: a defeasible reasoning platform available on the web. It is geared towards decision making where each agent has its own knowledge base that can be combined with other agents to detect and visualize conflicts and potentially solve them using a semantics. It allows the use of different defeasible reasoning semantics (ambiguity blocking/propagating with or without team defeat) and integrates agent collaboration and visualization features.

Relevance and Related Work

One of the aims of knowledge representation and conflict-tolerant reasoning is to help with the decision making process where the knowledge of different stakeholder agents can be put together in order to help reach a decision or at least detect conflicts. In order to reason with combined knowledge, a conflict-tolerant reasoning mechanism is needed. Defeasible reasoning (Nute 1988) allows to reason with incomplete knowledge where conclusions can be challenged by additional information. It provides different semantics given its systematic reliance on a set of intuitions and rules of thumb (ambiguity propagating. blocking, team defeat), which have been long debated between logicians.

Statement Graphs (SG) (Hecham, Bisquert, and Croitoru 2018) have been defined in order to represent different semantics of Defeasible Logics (Antoniou et al. 2000) in a single formal framework.

This demonstration paper presents a platform called DAMN[1] that implements SG and provides a data engineer with the following features:

- Multi-agent integration: allowing agents to have their own knowledge bases (having, potentially, nothing in common).
- Real-time agent collaboration with access control.
- Defeasible reasoning on the combined knowledge featuring ambiguity propagating/blocking, with or without team defeat, used to solve contradictions between agents.
- Visualization of the resulting reasoning graph.
- Importing knowledge bases using DLGP formats (Baget et al. 2015), or a JSON file.

A presentation video explaining all of the features is available online at http://tiny.cc/mx2hcz.

[1]https://hamhec.github.io/damn/home

Technical Details

Let us consider the following knowledge: $kowalski$ is undeniably a penguin with potentially broken wings, penguins are bird who do not fly, generally birds fly (a less trustworthy rule), and one cannot say that birds with broken wings fly. The associated SG, with the queries $fly(kowalski)$ and $notFly(kowalski)$ is shown in Figure 1.

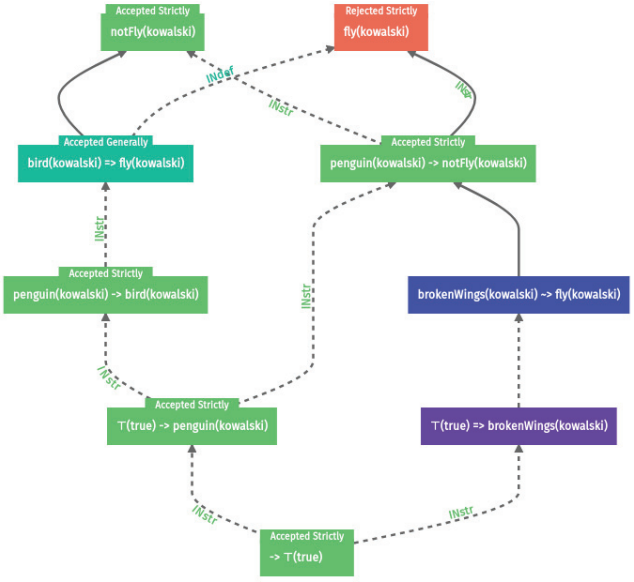

Figure 1: Statement Graph example.

DAMN Workflow. The typical workflow of the DAMN platform is: (1) Each user creates a project that will contain the different knowledge bases; (2) The user can add or remove agents along with their knowledge bases (that can be imported from DLGP files); (3) The user can invite other users to update the $\mathcal{KB}$ of some or all agents in real-time; (4) The user can build the SG using the $\mathcal{KB}$ of some or all

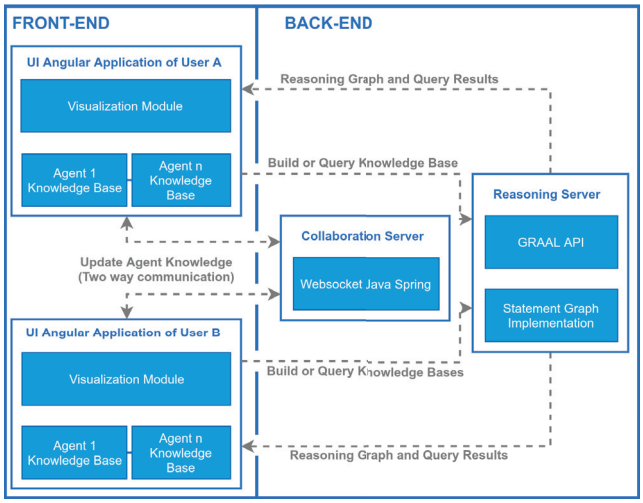

Figure 2: DAMN architecture.

the agents and run queries with a certain semantics; (5) Last, the user can interact with the resulting graph (see Figure 1 for an example of a graph).

DAMN Architecture. The DAMN architecture (cf. Figure 2) is composed of:

- A client side application (UI built with Angular) for user interaction that displays the agents and their knowledge bases along with graph visualization features.
- A collaboration server that handles the interaction between different users on the same project (a Spring Websocket server).
- A reasoning server that contains an implementation of SG (Java REST server) based on GRAAL (Baget et al. 2015) extended with features specific to defeasible reasoning (semantics, preferences between rules, etc.).
- A persistence server that stores the data (a MongoDB server).

Demonstration Application Scenario. In the demonstration we will consider the real world scenario issued from the EU H2020 NoAW project, where 21 experts from different domains put together their knowledge in order to reason about how to manage wastes from wine production. We will show how DAMN allowed the detection of conflicts. For instance, due to obsolete knowledge one wine maker argued that the wine pomace should be sent to distilleries because it was mandatory by law. Another agent argued that it was no longer the case since 2016. This use-case is available along with other examples (such as a group of friends trying to choose a restaurant to go to) on the website of the tool (not accessible here for privacy reasons).

Significance and Discussion

Different tools for defeasible reasoning have been proposed in the literature, most notably, *ASPIC+*[2] (Prakken 2010), *DeLP*[3] (García and Simari 2004), *DEFT* (Hecham, Croitoru, and Bisquert 2017), *ELDR* (Hecham, Bisquert, and Croitoru 2018), *Flora-2* (Wan, Kifer, and Grosof 2015), and *SPINdle* (Lam 2012). However, each tool allows for a different set of defeasible reasoning features and none of them provides support for multi-agent collaboration or visualization.

Given the recurrent need to reason with knowledge extracted from the Semantic Web, we opted for SG and their ability to represent most defeasible reasoning semantics and their support for existential rules. We have implemented SG and provided several features making it usable in a distributed decision making context. We evaluate the usefulness of our tool by considering a practical decision making scenario issued from the H2020 NoAW aiming at finding novel wine waste management methods.

Acknowledgments

The authors acknowledge the support of the CASDAR Docamex Programme from the French Ministry of Agriculture (2016–2020) and the H2020 NoAW European project.

References

Antoniou, G.; Billington, D.; Governatori, G.; Maher, M. J.; and Rock, A. 2000. A Family of Defeasible Reasoning Logics and its Implementation. In *Proceedings of the 14th European Conference on Artificial Intelligence*, 459–463.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; Rocher, S.; and Sipieter, C. 2015. Graal: A toolkit for query answering with existential rules. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, 328–344. Springer.

García, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming* 4(1+ 2):95–138.

Hecham, A.; Bisquert, P.; and Croitoru, M. 2018. On a Flexible Representation of Defeasible Reasoning Variants. In *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*.

Hecham, A.; Croitoru, M.; and Bisquert, P. 2017. Argumentation-based defeasible reasoning for existential rules. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, volume 3, 1568—-1569.

Lam, H. P. 2012. On the Derivability of Defeasible Logic.

Nute, D. 1988. *Defeasible reasoning: a philosophical analysis in prolog*. Springer.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. *Argument and Computation* 1(2):93–124.

Wan, H.; Kifer, M.; and Grosof, B. N. 2015. Defeasibility in answer set programs with defaults and argumentation rules. *Semantic Web* 6(1):81–98.

[2]http://aspic.cossac.org

[3]http://lidia.cs.uns.edu.ar/delp_client