A prototype system for argumentation-based reasoning about trust

Yuqing Tang[1], Kai Cai[1], Elizabeth Sklar[1,2], and Simon Parsons[1,2]

[1] Department of Computer Science, Graduate Center
City University of New York, 365 Fifth Avenue, New York, NY 10016, USA
{ytang,kcai}@gc.cuny.edu
[2] Department of Computer & Information Science, Brooklyn College,
City University of New York, 2900 Bedford Avenue, Brooklyn, NY 11210 USA
{sklar,parsons}@sci.brooklyn.cuny.edu

Abstract. This paper describes a prototype system for reasoning about trust. The system takes as input a set of statements in logic, each attributed to an agent, and a social network that indicates what trust relations hold between the agents. Given a query in the form of a proposition, and an agent that is interested in that query, the system returns a graph containing all arguments that have the proposition as its conclusion and all the arguments that bear on the acceptability of the conclusion.

1 Introduction

Trust is a mechanism for managing the uncertainty about autonomous entities and the information they deal with. As a result, trust can play an important role in any decentralized system, and particularly in multiagent systems, where agents are often engage in competitive interactions. As a result, there has been much work on the topic of trust in multiagent systems [7].

The system we describe here is a partial implementation of the formal system from [9], where a full description of the system can be found. That work combines work on propagating trust through a social network with argumentation, showing how the results of this propagation can be linked to Dung-style [2] argumentation — where the arguments are structured as in [3,6]. The result is a system in which one agent can reason using information from other agents that it knows through the social network, assigning belief to that information depending on how much the agents that are the source of the information are trusted (as in [5]).

Following Castelfranchi and Falcone [1], we believe that trust is based on reasons. We interpret this to mean that there is an advantage in clearly identifying the sources of information and relating these to the conclusions drawn from them, the sources and their connections to the conclusion being the reasons. We track these relationships using argumentation, and we summarize the resulting connections as a graph, which is presented to the users of our software. We see these users as being individuals who have to make decisions based on information that comes from acquaintances of varying trustworthiness.

2 Trust and argumentation

Here we briefly and informally sketch the model our prototype implements. An agent Ag_i has knowledge base $\Sigma_i = F_i \cup \Delta_i$. F_i is a set of facts, each of which is a logical statement in a language $\mathcal{L}$. Δ_i is a set of inference rules δ each of which links some set of premises $p_i \in \mathcal{L}$ to a conclusion $c \in \mathcal{L}$.

Inference rules can be taken to be nodes in a graph with connections to premises and conclusions, and arguments are then graphs constructed of such components. Each premise and rule in an argument comes from some Σ_i, and the agent that owns the relevant Σ_i is connected to those facts and rules. This makes it simple to identify the source of each piece of information, and hence the basis for computing the level of belief that should be accorded to it by the agent constructing the argument.

As in all work on argumentation, we allow for arguments to defeat one another, recognizing four forms of defeat where the conclusion of one argument disagrees with (1) the conclusion of another (called "rebut"), (2) an initial premise of another ("premise-undercut"), (3) the conclusion of a rule in another that is not the final conclusion ("intermediate-undercut"), or (4) an inference rule ("inference-undercut"). In [9] we show how to interpret argumentation semantics from Dung [2] to compute which subset of arguments is acceptable. The current version of the software implements the grounded semantics to do this, and we are working on the implementation of additional semantics.

3 The system

This section describes which aspects of the formal system from [9] are currently implemented. The software is written in Java and is available on request under an open-source license.

3.1 The language

The system allows a subset of first-order logic to be used to represent premises, conclusions and intermediate conclusions. (This is the implementation of $\mathcal{L}$.) The grammar for these sentences is:

$$
\begin{aligned}
\mathit{Sentence} &\rightarrow \mathit{AtomicSentence} \mid \mathit{CompoundSentence} \\
\mathit{AtomicSentence} &\rightarrow \mathit{Predicate}(\mathit{Term}_1, \ldots, \mathit{Term}_n) \\
\mathit{CompoundSentence} &\rightarrow (\mathit{Sentence}) \mid \mathit{NOT}\ \mathit{Sentence} \mid \mathit{Sentence}\ \mathit{AND}\ \mathit{Sentence} \\
&\quad \mid \mathit{Sentence}\ \mathit{OR}\ \mathit{Sentence} \mid \mathit{Sentence}\ => \ \mathit{Sentence} \\
&\quad \mid \mathit{Sentence}\ <=> \ \mathit{Sentence} \\
\mathit{Term} &\rightarrow \mathit{Constant} \mid \mathit{Variable} \\
\mathit{Constant} &\rightarrow A \mid X_1 \mid \mathit{john} \ \text{(symbols explicitly specified as constants)} \\
\mathit{Variable} &\rightarrow x \mid s \qquad \text{(non-constant symbols that start with a lower case letter)}
\end{aligned}
$$

The symbol => denotes implication; <=> denotes the bi-conditional. As described above, the system operates on knowledge facts (*Fact*) and inference rules (*InferenceRule*) and the syntax of these is defined as below:

$$\begin{array}{rlrl} Fact & \rightarrow Sentence & PremiseList & \rightarrow Sentence_1, ...Sentence_m \\ InferenceRule & \rightarrow \frac{PremiseList}{Conclusion} & Conclusion & \rightarrow Sentence \end{array}$$

While the systems allows facts, premises and conclusions to be specified using the subset of logic above, the current implementation is sound and complete only if facts, premises and conclusions are all literals (that is, positive atomic sentences or negated atomic sentences).

3.2 Input

The system takes as input an XML file in a format which we only have room to sketch here (full details on request). First, we have a specification of the trust graph, for example:

```
<trustnet>
     <agent> john </agent>
     ...
      <trust>
          <truster> john </truster>
          <trustee> mary </trustee>
          <level> 0.9 </level>
      </trust>
     ...
</trustnet>
```

which specifies the agents involved and the trust relationships between them, including the level of trust (specified as a number between 0 and 1). We also have the specification of each agent's knowledge and degree of belief for each component of its knowledge base:

```
<beliefbase>
    <belief>
        <agent> john </agent>
        <fact> IndieFilm(hce) </fact>
        <level> 0.8 </level>
    </belief>
    ...
    <belief>
        <agent> dave </agent>
        <rule>
            <premise> IndieFilm(x) </premise>
            <premise> DirectedBy(x, almodovar) </premise>
            <conclusion> Watch(x) </conclusion>
        </rule>
        <level> 0.89 </level>
    </belief>
    ...
</beliefbase>
```

The current implementation computes the trust that one agent places on another, given the network specified in the `trustnet` construct, using TidalTrust [4].

3.3 Output and interface

Given the XML input file, the system can answer queries about whether a given conclusion can be derived by a given agent. The system is invoked from the Unix command line, and generates output in the form of an annotated `dot`[3] description of a graph. This can be converted to PDF — an example of a PDF file generated by the system is given in Figure 1. Since this output rapidly becomes hard to understand, we are working on approaches to providing a zoomable interface. The current prototype version of the software can generate a simple zoomable HTML interface which allows the user to look at the relationship between the arguments (represented as a simple graph with one node for each argument, as in Dungine [8]) and to expand any argument to look at its internal structure. We are currently porting this interface to run on mobile devices under Android.

Acknowledgements Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. C. Castelfranchi and R. Falcone. Trust is much more than subjective probability: Mental components and sources of trust. In *Proceedings of the 33rd Hawaii International Conference on System Science*, Maui, Hawai'i, January 2000.
2. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.
3. A. J. García and G. Simari. Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.
4. J. Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland, College Park, 2005.
5. C-J. Liau. Belief, information acquisition, and trust in multi-agent systems — a modal logic formulation. *Artificial Intelligence*, 149:31–60, 2003.
6. H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1:93–124, 2010.
7. J. Sabater and C. Sierra. Review on computational trust and reputation models. *AI Review*, 23(1):33–60, September 2005.
8. M. South, G. Vreeswijk, and J. Fox. Dungine: A Java argumentation engine. In *Proceedings of 1st International Conference on Computational Models of Argument*, Toulouse, France, September 2008.
9. Y. Tang, K. Cai, P. McBurney, E. Sklar, and S. Parsons. Using argumentation to reason about trust and belief. *Journal of Logic and Computation*, (to appear), 2011.

[3] `http://www.graphviz.org/`

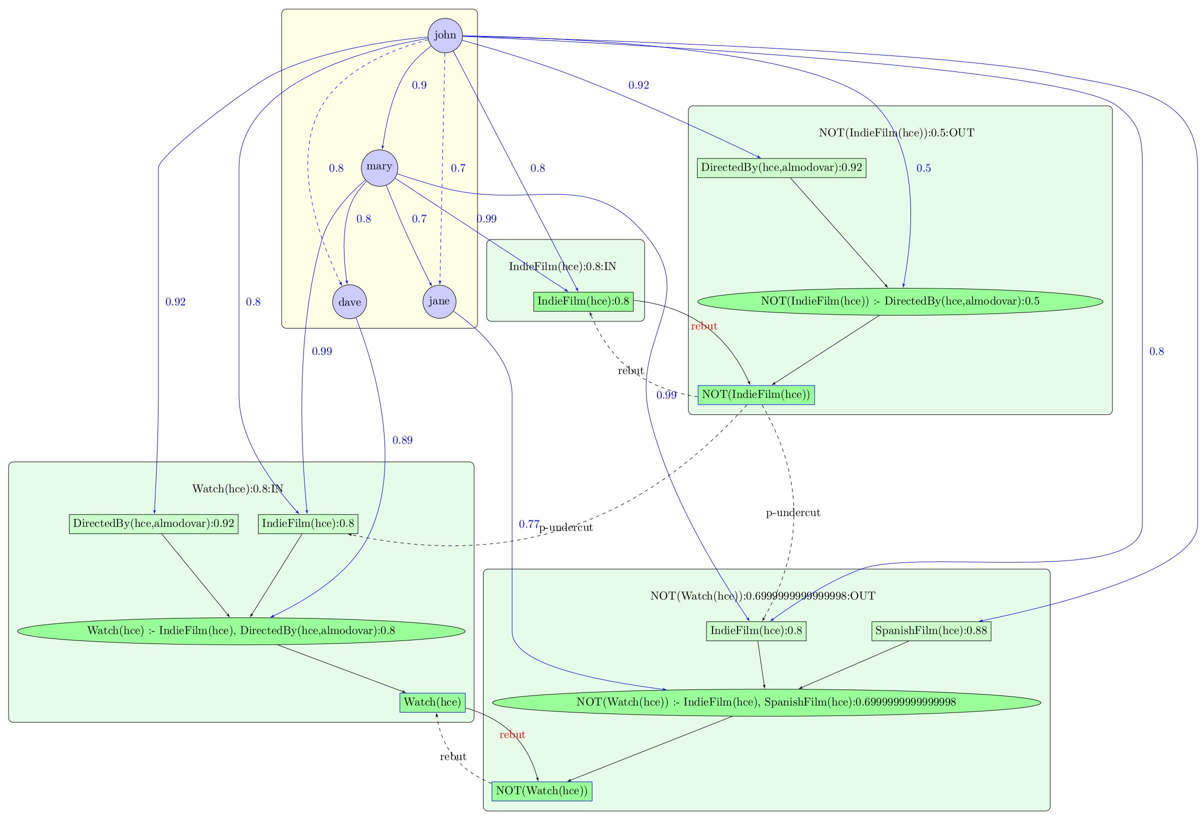

Fig. 1. A trust and argumentation graph generated by the system. Note (a) the trust graph in the top left corner; (b) the links between agents in the trust graph and the information provided by those agents; (c) the arguments (the green nested boxes); (d) the rebut and undercut relationships between the argument, and the labelling of the arguments as IN or OUT.