Automated Reasoning for Collective Information Management

F.Toni
Imperial College of Science, Technology and Medicine
Department of Computing
180 Queen's Gate, London SW7 2BZ, UK.
ft@doc.ic.ac.uk

Abstract

This paper describes an automated reasoning approach for managing information generated by ordinary people interacting in a community. Information is organised in networks of interconnected nodes that may be heterogeneous and distributed over a computer network.
The highly dynamic and distributed nature of these information networks poses problems of effective generation and maintenance, so that given properties (or meta-data) are satisfied. The paper investigates the use of logic-based software agents that act on behalf of the individuals in the community for the generation and maintenance of such networks. The agents rely upon a logic-based representation of the meta-data and make use of the automated resoning engine the chosen representation is equipped with for checking as well as enforcing the properties the meta-data represent.

1 Introduction

This paper addresses the problem of managing information generated by ordinary people interacting in a community. Information is organised in networks of interconnected nodes to capture the complexity of social interactions. Nodes in such networks may be heterogeneous and distributed over a computer network. The links between nodes of the information networks represent symbolic connections between the information held at the nodes. The networks can grow dynamically, as nodes proliferate and links are added. Thus, these networks support the complexity of collectively generated information. Examples of these networks include the WWW and networks for collective memory management [7].
The highly dynamic and distributed nature of these networks poses problems of effective generation and maintenance, so that given properties are satisfied. These properties can be thought of as meta-data on the (object-level) data held at the nodes of the network as well as on the structure of the network. For example,

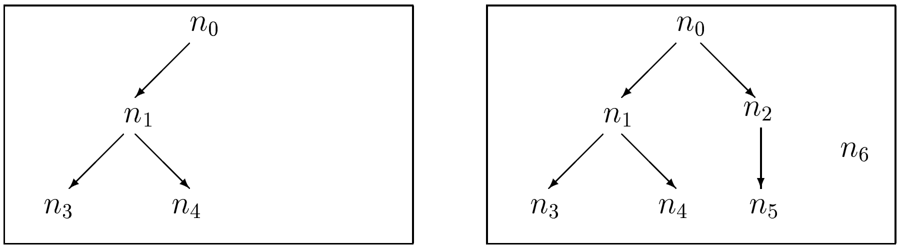

Figure 1: (a) A simple information network, and (b) Unreferenced nodes in information networks

one might require that every node holding title and author of a book should link to two nodes, one holding a brief description of the content of the book, the other holding a list of addresses of libraries where copies of the book can be borrowed from. In addition to domain-specific meta-data of the kind above, one might require that any given network satisfies "structural" properties, e.g. that it contains no dangling links and no unreferenced nodes.
The paper investigates the use of logic-based software agents that act on behalf of the individuals in the community for the generation and maintenance of such networks. These agents are general-purpose and can be employed for a number of other applications [5, 8].
The agents rely upon a logic-based representation of the meta-data and make use of the automated resoning engine the chosen representation is equipped with for *checking* as well as *enforcing* the properties the meta-data represent. Indeed, the proposed techniques allow not only to identify unwanted features of the networks, but also suggest possible *repair* strategies, in the form of actions that can be taken to put the given network into a format satisfying the properties. For example, the addition of new nodes and/or links might be reccommended. Moreover, the agents might be instructed to actually perform the changes.
The paper is organised as follows. Section 2 provides motivations and describes the examples we will concentrate on in the paper. Section 3 describes the formalism for knowledge representation underlying the proposed automated reasoning technique, summarises the technique itself, and decribes the behaviour of the approach for some simple examples. Section 4 describes the software agent architecture embedding the automated reasoning technique. Section 5 concludes and discusses directions for future research.

2 Motivations

In this section we illustrate some of the problems we want to address in this paper. Figure 1(a) exemplifies a simple information network with nodes n_0, n_1, n_3, n_4 and links between nodes given by the directed edges. The links represent symbolic connections between the information held at the nodes connected by the links. For example, the network might hold information about books reviewed by people in a given community, to be accessed by other people. The node n_0 could

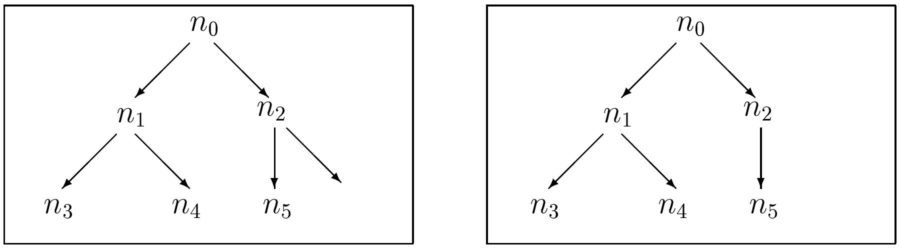

Figure 2: (a) Dangling links and (b) violation of meta-data in information networks

be the entry node to the network, node n_1 could be about some concrete book1, node n_3 could contain a brief description of the content of book1, and node n_4 could hold a list of addresses of libraries where copies of book1 can be borrowed. The network evolves over time, as people add and delete nodes and links, and modify information held at nodes. Figure 1(b) shows a possible extension of the simple network in figure 1(a) by the addition of nodes n_2, n_5, n_6 and links from n_0 to n_2 and from n_2 to n_5. The extension violates the *structural property* that networks contain *no unreferenced nodes*. The extensions can be modified so that the property is satisfied, by adding a link from some node amongst $n_0, \ldots, n_5$ to n_6. Figure 2(a) shows an alternative extension of the simple network in figure 1(a) by the addition of nodes n_2, n_5 and links from n_0 to n_2, from n_2 to n_5 and from n_2 to some unspecified node. The extension violates the *structural property* that networks contain *no dangling links*. The extensions can be modified so that the property is satisfied, by adding a node to which the dangling link points.

Note that, even if the network is structurally correct, as in figure 2(b), it might violate some properties the information held by the network must satisfy. These properties can be seen as *meta-data*, because they refer to the data (or information) held by the network. For example, one such property could be that for every book reviewed in the network both a brief description of its content and a list of libraries holding copies of the book must be provided. Then, if nodes n_1, n_2 hold information about book1 and book2, respectively, nodes n_3, n_5 describe the content of book1 and book2, respectively, and node n_4 holds a list of libraries from which book1 can be borrowed, then the network in figure 2(b) violates the above meta-data. The network can be repaired by adding a node n_6 holding the information about the required list of libraries and adding a link from n_2 to n_6.

In the remainder of the paper we will illustrate the use of existing logic-based, automated reasoning techniques [3, 6, 5, 8, 1] to detect violation of structural properties and meta-data in information networks and to suggest repairs automatically. These techniques rely upon a specific logic-based representation of networks, structural properties and meta-data, described in the next section.

3 Knowledge representation and automated reasoning

Information networks, structural properties and meta-data can be represented via **abductive logic programs** [1] [4], namely triples $\langle P, A, I\rangle$ such that

- P is a **logic program**, i.e. a set of if-implications of the form
  $Head \leftarrow Body$
  where $Head$ is an atom, $Body$ is a (possibly empty) conjunction of atoms and negations of atoms and all variables in $Head$ and $Body$ are implicitly universally quantified from the outside.
  P is understood as an incomplete set of data or beliefs.

- A is a set of atoms referred to as **abducibles**.
  The atoms in A are understood as hypotheses that can be used to expand the theory.

- I is a set of **integrity constraints**, i.e. if-then implications of the form
  $Cond \Rightarrow Conc$
  where $Cond$ is a (possibly empty) conjunction of atoms and negations of atoms, $Conc$ is an atom and all variables in $Cond$ and $Conc$ are implicitly universally quantified from the outside.
  Sentences in I are understood as sentences that must be "satisfied" by any sets of abducibles expanding the logic program, in the same way integrity constraints in databases are understood as sentences that must be "satisfied" in every database state.

Given an **observation** (conjunction of literals) O, a set of abducibles $\mathcal{D}$, and a variable substitution θ for the variables in O, the pair $(\mathcal{D}, \theta)$ is an **(abductive) explanation** for O, given an abductive logic program as above, if

1. $P \cup \mathcal{D}$ **entails** $O\theta$, and

2. $P \cup \mathcal{D}$ **satisfies** I.

Various notions of entailment and satisfaction can be adopted [4], but their provision is beyond the scope of this paper.
An **abductive proof procedure** is an engine for the automatic computation of explanations for observations. Examples of abductive proof procedures can be found in [3, 6, 8]. Typically, abductive proof procedures interleave backward reasoning with if-definitions in the logic program (reduction of $Head$ to $Body$) and forward reasoning with if-then implications in the integrity constraints (addition of $Conc$ to $Cond$).

[1] **Abduction** is the reasoning mechanism allowing to infer premises of implications from their conclusions. It can be seen as the "opposite" of *deduction*, which allows to infer conclusions of implications from their premises. However, whereas deduction is a sound form of reasoning, abduction is not.

Abductive logic programming can be used to represent (software and hardware) agents, and abductive proof procedures can serve as the engine underlying the reasoning process of these agents [5, 8]. In particular, abducibles can be thought of as (physical and communicative) actions that agents may perform, logic programs as beliefs describing how to construct plans to achieve given objectives of agents, integrity constraints as condition-action and committment rules that agents should obey to in order to construct their plans dynamically, as well as obligations, prohibitions and communciation protocols [8]. In this setting, observations can be thought of as agents' objectives as well as incoming inputs agents need to react to, and explanations can be thought of as plans of action. This understanding of abductive logic programming is at the hearth of the system architecture described in the next section 4.

3.1 Checking integrity of information networks

Information networks can be represented via sets of atoms in the predicates *link* and *node*. For example, the simple network in figure 1(a) can be represented by $P_{1(a)}$:

$$\begin{aligned}&\{link(n_0, n_1),\\ &\ link(n_1, n_3),\\ &\ link(n_1, n_4),\\ &\ node(n_0),\\ &\ node(n_1),\\ &\ node(n_3),\\ &\ node(n_4)\}.\end{aligned}$$

$P_{1(a)}$ is a logic program consisting solely of if-implications with an empty body.
The structural property that there are no dangling links can be represented by means of the set of integrity constraints I_{dang}:

$$\{link(X, Y) \Rightarrow node(X),\ link(X, Y) \Rightarrow node(Y)\}$$

expressing that every link in a network connects two existing nodes.
The additional set of if-implications $P_{1(a),type}$:

$$\begin{aligned}&\{book(n_1),\\ &\ content(n_3),\\ &\ libraries(n_4)\}\end{aligned}$$

represents the "type" of the nodes, namely the sort of information held at the nodes.
The meta-data that every "book" node should be linked to a "content" node and a "libraries" node can be expressed by I_{meta}:

$$\{book(X) \Rightarrow twolinks(X)\}$$

where the predicate *twolinks* is defined by the logic program P_{meta}:

$$\{twolinks(X) \leftarrow \quad link(X,Y)\,\&\,content(Y)\,\& \\ link(X,Z)\,\&\,libraries(Z)\}$$

Then, given the abductive logic program
$\langle P_{1(a)} \cup P_{1(a),type}, \{\}, I_{dang} \cup I_{meta}\rangle$,
any correct abductive proof procedure returns $true$ as an explanation for the empty observation. Indeed, the given network satisfies the structural property as well as the meta-data on the information it contains.
However, let
$P_{2(b)} = \{node(n_2), node(n_5), link(n_0, n_2), link(n_2, n_5)\}$
$P_{2(b),type} = \{book(n_2), content(n_5)\}$.
Note that $P_{1(a)} \cup P_{2(b)}$ represents the network in figure 2(b). Moreover, $P_{2(b),type}$ represent the "type" of the new nodes in the network in figure 2(b). Then, given the abductive logic program
$\langle P_{1(a)} \cup P_{2(b)} \cup P_{1(a),type} \cup P_{2(b),type}, \{\}, I_{dang} \cup I_{meta}\rangle$,
any correct abductive proof procedure returns $false$ as an explanation for the empty observation. Indeed, the given network violates the meta-data.
So far we have illustrated how abductive proof procedures can be used to detect compliance of given information networks with required structural properties and meta-data, based upon the representation of networks, properties and meta-data via abductive logic programs. In the remainder of this section we will illustrate how the same techniques can be used to suggest repairs to non-compliant networks, by modifying the knowledge representation (namely the abductive logic program).

3.2 Repairing information networks

We introduce predicates *virtual-node*, *add-node*, *virtual-link*, *add-link*, and a new logic program P^{new}:

$$\{virtual\text{-}node(X) \leftarrow node(X), \\ virtual\text{-}node(X) \leftarrow add\text{-}node(X), \\ virtual\text{-}link(X,Y) \leftarrow link(X,Y), \\ virtual\text{-}link(X,Y) \leftarrow add\text{-}link(X,Y)\}.$$

Intuitively, predicates *node* and *links* are used to represent given networks as above, and *add-node* and *add-link* can be seen as repair actions. Indeed, all atoms in the predicates *add-node* and *add-link* are *abducibles*, namely they can be added to the given logic program if needed, e.g. to avoid the integrity constraints to be violated.
The additional integrity constraints I^{false}:

$$\{add\text{-}link(X,Y)\&link(X,Y) \Rightarrow false, \\ add\text{-}node(X)\&node(X) \Rightarrow false\}$$

guarantee that no repair action is taken unless absolutely necessary.
The predicates *virtual-node* and *virtual-link* are used to describe "intentionally" repaired networks, before the networks are physically created. The system architecture described in the next section can be used as a support to revise networks physically according to reccommended repairs.
Similarly, we introduce predicates *virtual-book*, *add-book*, *virtual-content*, *add-content*, *virtual-libraries*, *add-libraries*, and add to P^{new} analogous definitions as the ones above.
The integrity constraints I_{dang} and I_{meta} are rewritten as I^{new}_{dang} and I^{new}_{meta}, respectively, where:

$$
\begin{aligned}
I^{new}_{dang} = \{ & \textit{virtual-link}(X,Y) \Rightarrow \textit{virtual-node}(X), \\
& \textit{virtual-link}(X,Y) \Rightarrow \textit{virtual-node}(Y)\}
\end{aligned}
$$

$$I^{new}_{meta} = \{\textit{virtual-book}(X) \Rightarrow \textit{twolinks}(X)\}$$

and the predicate *twolinks* is defined by P^{new}_{meta}:

$$
\begin{aligned}
\{\textit{twolinks}(X) \leftarrow \quad & \textit{virtual-link}(X,Y) \,\&\, \textit{virtual-content}(Y) \,\&\, \\
& \textit{virtual-link}(X,Z) \,\&\, \textit{virtual-libraries}(Z)\}
\end{aligned}
$$

Let A be the following set of abducibles:

$$
\begin{aligned}
\{ & \textit{add-node}(t), \textit{add-link}(t,t'), \textit{add-book}(t), \textit{add-content}(t), \textit{add-libraries}(t) \mid \\
& \qquad t, t' \text{ are variable-free terms in the vocabulary of the network}\}.
\end{aligned}
$$

Then, given the abductive logic program

$$\langle P_{1(a)} \cup P_{1(a),type} \cup P^{new} \cup P^{new}_{meta}, A, I^{new}_{dang} \cup I^{new}_{meta}\rangle,$$

and an observation O (set of input actions)

$\textit{add-link}(n_0, n_2)\&$
$\textit{add-link}(n_2, n_5)\&$
$\textit{add-node}(n_2)\&$
$\textit{add-node}(n_5)\&$
$\textit{add-book}(n_2)\&$
$\textit{add-content}(n_5)$

representing the extension of the network in figure 1(a) to that in figure2(b), any correct abductive proof procedure computes the explanation

$$\{\textit{add-link}(n_2, X), \textit{add-libraries}(X)\} \cup O$$

where X is existentially quantified. The subset of this explanation not in the original O can be seen as a reccommended repair to the network.

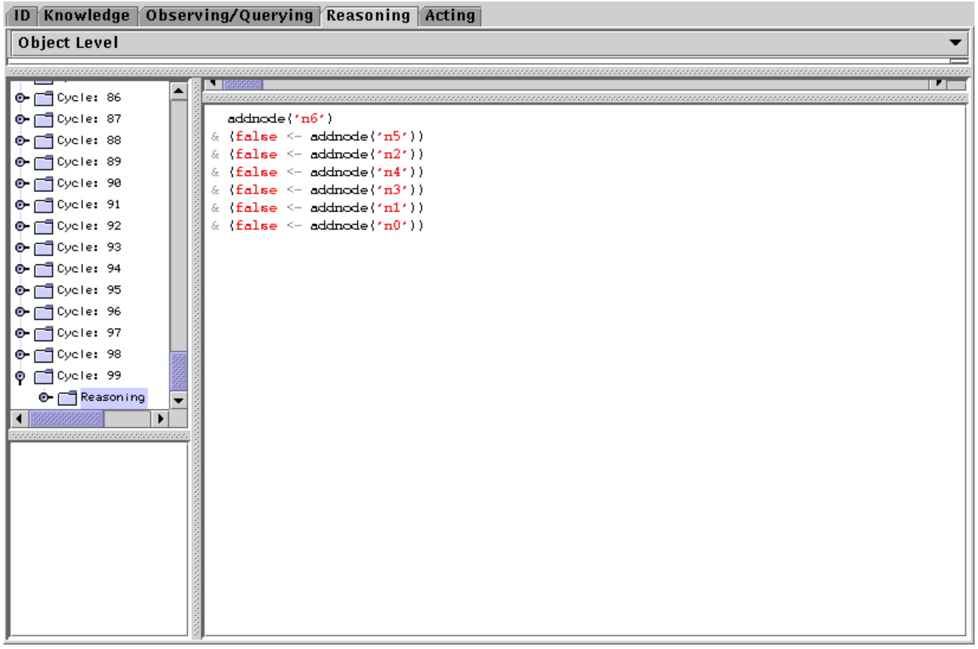

Figure 3: Monitoring the reasoning of the agent

4 System architecture

Abductive proof procedures can serve as the reasoning engine underlying logic-based agents whose beliefs and reactive rules are represented via abductive logic programs [5, 8].

A prototype implementation of logic-based agents of this kind is described in [1]. In the implementation, the Java programming language is used for implementing the agents and their interactions with the environment, and Voyager (see `http://www.objectspace.com/`) for implementing the inter-agent features (such as communication). In the implementation, the execution of the abductive proof procedure presented in [3] is interleaved with the analysis of incoming observations and the execution of selected actions, via a resource-bounded sense-reason-act cycle. The level of rationality of agents depends on the amount of resources assigned to the reasoning part of the cycle (the more the resouces, the more rational, but less reactive, the agent).

Figure 3 illustrates the component of the system implementation responsible for viewing and monitoring the Reasoning activity of the agent. The identifier (ID) of an agent is the name other agents use to interact (via communication) with the given agent. The Knowledge of the agent is given by an abductive logic program. New observations (and queries) to the agent can be inputed via the Observing/Querying component of the implementation. Actions are abducibles in selected explanations and are dynamically selected, as they are generated, by the Acting component. The Reasoning component repeatedtly applies, a pre-defined number of times, the operations of the underlying abductive proof procedure.

In the concrete case shown in the figure, the abductive logic program represents the network in figure 2(a) and the structural property that there are no dangling

links in information network, in the spirit of section 3.2. The right-most box on the screen shows the explanation computed for the empty observation. Note that the suggested repair (correctly) amounts to adding an additional node, n_6. The remaining conjuncts in the explanation are "residues" of the integrity constraint that no node should be added if already existent in the network.
The use of the techniques described in the previous section 3 within the agent architecture above is beneficial, because it can serve as a basis for a completely autonomous tool for network handling. This tool would check for properties to be satisfied, would suggest repairs when needed, and would either implement repairs directly, whenever possible, or guide the people creating the network through the process of repair. Moreover, the agent presentation paves the way to a distributed, multi-agent modeling of the problem, where different agents are responsible for checking and enforcing different properties. This would allow higher modularity and reliability.

5 Conclusions

We have argued that abductive logic programming, an existing general-purpose tool for knowledge representation in artificial intelligence, can be used to specify desired properties of networks of information so that existing automated reasoning tools abductive logic programming is equipped with can be used to check and enforce these properties. This is especially beneficial when the networks are generated by ordinary people, since in this setting the integrity of the network is likely to be violated.
We have argued that an agent architectures relying upon the above mentioned logic-based techniques would pave the way to autonomous and distributed tools for network handling.
Our proposal is somewhat related, in its aims, to the proposal of [2]. However, we propose to adapt general-purpose techniques rather than define ad-hoc tools for the problem at hand.

Acknowledgements

This reasearch was supported by the EPSRC project "Logic-based multi-agent systems". The author is grateful to Kostas Stathis and anonymous referees for many useful suggestions.

References

[1] Y. Carer, *Distributed communicating agents.* Individual Project, Department of Computing, Imperial College, 1998/1999.

[2] M. Fernández, D. Florescu, A. Levy, D. Suciu, Verifying integrity constraints on web sites. *Proc. IJCAI'99, 16th International Joint Conference on Artificial Intelligence*, pages 36–41, T. Dean (ed.), Morgan Kaufmann, 1999.

[3] T.H. Fung, R.A. Kowalski, The iff procedure for abductive logic programming. *Journal of Logic Programming* 33(2):151–165, Elsevier.

[4] A.C. Kakas, R.A. Kowalski, F. Toni, The role of abduction in logic programming. *Handbook of Logic in AI and Logic Programming* 5:235–324, OUP, 1998.

[5] R.A. Kowalski, F. Sadri, From logic programming towards multi-agent systems. *Annals of Mathemathics and Artificial Intelligence* (Forthcoming).

[6] R.A. Kowalski, F. Toni, G. Wetzel, Executing suspended logic programs. *Fundamenta Informaticae* 34(3):203–224, ISO Press, 1998.

[7] A. Mamdani, J. Pitt, K. Stathis, Connected communities from the standpoint of multi-agent systems. *New Generation Computing Journal*, Special issue on New Challenges in Intelligent Systems, 17(4), T. Nishida (ed.), 1999.

[8] F. Sadri, F. Toni, Abduction with negation as failure for active databases and agents. *Proc. AI*IA 99, 6th Congress of the Italian Association for Artificial Intelligence*, pages 353–362, Pitagora Editrice Bologna, 1999.