Defeasible Logic Graphs for Decision Support

Donald Nute

Artificial Intelligence Center
Department of Philosophy
The University of Georgia
Athens, GA 30602, U.S.A.

Katrin Erk

Department of Computer Science
University of Koblenz
Koblenz, Germany

Abstract

Knowledge based systems provide decision support by applying a previously developed representation of knowledge for a particular domain. We describe a method for representing knowledge about any domain using defeasible logic graphs. Because these graphs are based on a defeasible logic of the sort described in [9], they can represent uncertain or incomplete knowledge.

We reason about the represented domain by propagating markers in the graph to show which propositions are true, false, or unestablished. We propose to construct an argumentation based software system incorporating defeasible logic graphs. We establish the formal foundations for such a system by showing that the inference mechanism for defeasible logic graphs is sound and complete with respect to defeasible logic.

1 Introduction

Decision support systems include a wide variety of software systems designed to aid the user in making complex decisions.\(^1\) Knowledge based systems (KBS) that model inference about specific domains incorporate representations of the knowledge necessary to solve problems in their domains. We believe another kind of decision support tool is needed that allows users to model knowledge not already represented in the system. Such an argumentation based system (ABS) would provide tools to help the user represent knowledge about any domain in a graphic manner.

It would incorporate an inference mechanism to help the user derive conclusions from the knowledge that has been modeled. The system would make the inference process visible to the user and allow the user to construct a variety of “what-if” scenarios easily and quickly. While a KBS applies preselected argument structures to the information provided by the user, an ABS would allow the user to construct and evaluate competing arguments on any subject before making a decision. Systems of this sort have been presented in [1, 2].

An ABS inference mechanism should support reasoning in uncertain domains and reasoning with incomplete information. Many knowledge based systems use certainty factors, probabilities, fuzzy logic, or other essentially quantitative methods for this kind of reasoning. Finding useful numbers for such systems is a major part of the knowledge acquisition process. An ABS should incorporate a qualitative approach to the representation of uncertain or incomplete information, one that does not require the user to assign numbers to pieces of knowledge. The inference scheme must be reasonably simple and intuitive. Fortunately, recent AI research provides formalisms for defeasible reasoning in which a line of argument can be defeated by another line of argument ([3, 4, 5, 6, 7, 8, 9, 10, 11]). Defeasible formalisms directly model the ways in which arguments can rebut or undercut each other. These systems represent the pieces from which arguments are constructed as rules having no numerical component.

We will describe a visually oriented knowledge representation system and a defeasible inference mechanism that lends itself to implementation as an ABS. Knowledge is represented in a defeasible logic graph. A node in a graph represents the premise or conclusion of a rule. The arcs in the graph are arrows, and different kinds of arrows indicate different roles a rule can play in an argument. We reason with a graph by propagating the markers +, −, and ? through the graph to mark which of the premises or conclusions are true, false, or impossible to establish from available information. The system of defeasible logic graphs is based on the family of defeasible logics developed in [9]. We

\(^1\)The second author gratefully acknowledges support from the Deutscher Akademischer Austauschdienst during her year at the University of Georgia.
will state results that explain how the formal proof theory serves as a semantics for the defeasible logic graph system. Proofs for these results, which could not be included due to space limitations, are available in [12].

The method of defeasible logic graphs has not been implemented as a software program. In the final section of the paper, we describe the kind of ABS that could (and, we hope, will) incorporate the defeasible logic graph method.

2 The language

We define atomic formulas in the usual way. A literal is any atomic formula or its negation. Where p is an atomic formula, we say p and ~p are the complements of each other. ~p denotes the complement of any literal p, positive or negative.

Rules are a class of expressions distinct from formulas. Rules are constructed using three primitive symbols: →, ⇒, and ~. Where A U {p} is a set of formulas, A→p is a strict rule, A⇒p is a defeasible rule, and A ~p is a defeater. In each case, we call A the antecedent of the rule and call p the consequent of the rule. Where A = {q}, we denote A→p as q→p, and similarly for defeasible rules and defeaters. Antecedents for strict rules and defeaters must be non-empty; antecedents for defeasible rules may be empty. We will call a rule of the form 0⇒p a presumption and represent it more simply as ⇒p. All rules are read as "if-then" statements. We read A→p as 'If A, then p', A⇒p as 'If A, then evidently (normally, typically, presumptuously) p', and A ~p as 'If A, then it might be that p'. The role of a defeater is only to interfere with the process of drawing an inference from a defeasible rule. Defeaters never support inferences directly. If free variables occur in a rule, we interpret the rule as though all variables were bound by universal quantifiers that have the entire rule within their scope. So, for example, we read

Fx⇒Gx as 'F's are typically G's'.

Rules themselves are conceived as policies for forming and revising beliefs. A typical example of a defeasible rule in English is 'Birds fly'. We might accept this rule because most birds do fly. We might even call the English sentence true for this reason. But as a rule, we interpret the sentence to mean something like 'Take a thing's being a bird as evidence that it flies'. This is an imperative and does not have a truth value. We understand a rule, not by knowing what would make it true or false, but by knowing what would be involved in complying with it. So defeasible rules have compliance conditions rather than true conditions. We expect that any suitable semantics for defeasible logic will be procedural and will derive directly from proof theory. For further discussion, see [13].

Our formal language lacks the power to describe some situations that we might find interesting, but it is adequate for a wide range of examples. Adding disjunction and existential quantification to the language would increase the expressive power and disjunctive permissions raise interesting logical issues, but these possibilities will not be considered here.

3 Defeasible logic

Definition 1 A defeasible theory is a set of literals and rules. If T is a defeasible theory, then TR is the set of rules in T with nonempty antecedents.

We only apply a defeasible rule when we think that it is not defeated. In a formal system for defeasible reasoning, we sometimes need to show that something is not derivable from available information in order to apply a defeasible rule. For example, we know that birds typically fly and penguins do not. We shouldn't use the first of these rules to infer that a particular bird flies when we have evidence that the bird is a penguin. To demonstrate that flight evidently follows from what information we have about a particular bird, we need to demonstrate that available information will not support the conclusion that the bird is a penguin.

We write T ⊨ p to indicate that p is derivable from T using only the strict rules in T. We write T ⊢ p to indicate that the derivation of p from T may require the use of defeasible rules in T. We use T ′ ⊢ p to indicate that demonstrably p is not derivable from T using only the strict rules in T. And we use T ′″ p to indicate that demonstrably p is not derivable from T using both the strict and defeasible rules in T. Read T ⊨ p as T proves p, T ⊢ p as T supports p, T ′ ⊢ p as T won't prove p, and T ′″ p as T won't support p. Note that T ′ p is quite different from T ′″ p (p is not strictly derivable from T), and T ′″ p is quite different from T ′″ p (p is not defeasibly derivable from T). Just because a conclusion is not derivable does not mean that we can demonstrate that it is not derivable.

To define a defeasible logic Σ, we must specify when a defeasible theory T proves, won't prove, supports, or won't support a literal p in Σ. We do this by identifying Σ with a set of inference rules defining ⊨, ′, ″, and ′″. 

Definition 2 Where Σ is a set of rules defining ⊨, ′, ″, and ′″, T is a defeasible theory, and p is a
literal, \( T \vdash_\Sigma p \) iff we can show \( T \vdash p \) using finitely many applications of the rules in \( \Sigma \). We define \( T \vdash_\Sigma p \), \( T \vdash_\Sigma p \), and \( T \models_\Sigma p \)
similarly.

Not just any set of rules for \( \vdash, \ll, \models, \) and \( \vdash \) will define a reasonable logic. Since \( T \vdash p \) means \( p \) is derivable from \( T \) and \( T \vdash p \) means \( p \) demonstrably is not derivable from \( T \), a set of rules that allowed us to show both would be incoherent. The same is true for \( \models \) and \( \vdash \).

**Definition 3** \( \Sigma \) is a **defeasible logic** iff \( \Sigma \) is a set of rules defining \( \vdash, \ll, \models, \) and \( \vdash \), and there is no defeasible theory \( T \) and literal \( p \) such that either \( T \vdash_\Sigma p \) and \( T \vdash_\Sigma p \), or \( T \models_\Sigma p \) and \( T \models_\Sigma p \).

**Definition 4** \( M = \{ M+, M-, E+, EE- \} \).

**Theorem 1** \( M \) is a defeasible logic.

The system \( M \) represents a minimal defeasible logic, one that includes little more than the monotonic core of a defeasible logic. This system is quite close to the minimal system defined in [9] except that the rule \( EE- \) replaces the slightly weaker rule \( E- \) in that paper.

We could allow detachment of the consequent of a strict rule whenever its antecedent is defeasibly derivable. But consider the case when we have competing strict rules and the antecedents of both are only defeasibly derivable. In this case, we will not detach the consequent of either strict rule. This localizes contradictions. We call defeasible logics with this feature **semi-strict**.

**SS+**: If

1. \( T \vdash \neg p \),
2. there is \( A \rightarrow p \in T \) such that for all \( a \in A \), \( T \models a \), and
3. for each \( B \rightarrow p \in T \), there is \( b \in B \) such that \( T \models b \)
then \( T \models p \).

Now we introduce a rule for detaching the consequent of a defeasible rule.

**Definition 5** \( SD = M \cup \{ SS^+, D^\ll, SD^\ll \} \).

**Theorem 2** \( SD \) is a defeasible logic.
4 Defeasible graphs

The argumentation based system we envision will allow the user to build a graph that represents defeasible theories. Initial assumptions of the theory and the antecedents and consequents of rules in the theory become nodes in the graph, and rules in the theory become links in the graph. The user will build a graph by interacting with the screen with the keyboard and the mouse. Our defeasible language allows variables, but our ABS will only support graphs whose structure is propositional. In the rest of this paper, 'atom' will mean a propositional constant and 'literal' will mean an atom or its negation. Since users will only construct finite graphs, we restrict ourselves to graphs corresponding to finite theories. Special problems arise when there are loops in the reasoning. If we linked compound nodes to all the atoms that occur in them, these loops would show up in graphs as cyclical paths. We restrict our investigations to acyclic graphs.

We will mark a node in a graph with a + to indicate that it is (evidently) true and with a - to indicate that it is (evidently) false. We will sometimes establish that a particular literal cannot be shown to be true. We will indicate this by marking that literal with ?. When we use a graph to reason, we first mark some nodes with + or - to show our initial assumptions. Then we propagate markers through the graph. We want to develop a method for doing this which is sound and complete relative to our defeasible logic.

Definition 6 A defeasible graph is a labeled graph satisfying the following conditions.

1. Each node in the graph is labeled with finitely many literals or with the symbol T.
2. No two nodes have the same label.
3. Each arc in the graph is labeled -+, -*, +, or +.
4. If an arrow points toward a node or if a node is isolated, then the node is labeled with an atom. 

Definition 7 A defeasible graph G is acyclic iff there is no sequence σ₁,...,σₙ of links in G satisfying the following conditions:

1. for each i < n, if σᵢ is a positive (negative) link with head p, then p (∼p) is a member of the antecedent of σᵢ₊₁, and
2. if σₙ is a positive (negative) link with head p, then p (∼p) is a member of the antecedent of σ₁.

Definition 8 A marked graph is a defeasible graph in which some of the nodes are marked +, -, or ?.

Definition 9 Let G be a marked graph and let p be an atom.

1. The marked graph G* is the result of marking p with + (-) in G iff p is unmarked in G and G* is exactly like G except that
   (a) p is marked + (-) in G*;
   (b) for each unmarked compound node A in G* if
      i. p e A (∼p e A),
      ii. for every atom a e A, a is marked + in G*, and
      iii. for every literal ∼a e A, a is marked - in G*;
      then A is marked + in G*; and
   (c) for each unmarked compound node B in G*.

2. The marked graph G* is the result of marking p with ? in G iff p is unmarked in G and G* is exactly like G except that
   (a) p is marked ? in G*, and
   (b) for each unmarked compound node A in G such that p e A, A is marked ? in G*.

3. The marked graph G* is the result of marking ∼p with ? in G iff p is either unmarked in G or p is marked ? in G, and G* is exactly like G except that for each unmarked compound node A in G such that ∼p e A, A is marked ? in G*.

If a node in graph G is marked +, we will say the node is satisfied in G. If a node in graph G is marked - or ?, we will say the node is failed in G. We will say an arrow is satisfied (failed) in G if the antecedent of the arrow is satisfied (failed) in G.
Definition 10 Let $T$ be a defeasible theory. Then the initial graph for $T$ (in symbols, $G^T$) is the marked graph satisfying the following conditions.

1. For each atom or set of literals $x$, there is a node labeled $x$ iff either $x$ is the consequent or antecedent of some rule in $T$, or $x$ is an atom and either $x \in T$ or $\neg x \in T$.

2. $A \rightarrow p$ is in $G^T (A \rightarrow p$ is in $G^T, A \negrightarrow p$ is in $G^T)$ iff $A \rightarrow p \in T (A \rightarrow p \in T, A \negrightarrow p \in T)$.

3. $A \rightarrow p$ is in $G^T (A \rightarrow p$ is in $G^T, A \negrightarrow p$ is in $G^T)$ iff $A \rightarrow \neg p \in T (A \rightarrow \neg p \in T, A \negrightarrow \neg p \in T)$.

4. For each node $n$, either
   (a) $n$ is labeled by an atom $p$, $p \in T$, and $n$ is marked $+$, or
   (b) $n$ is labeled by an atom $p$, $\neg p \in T$, and $n$ is labeled $-$, or
   (c) $n$ is not marked.

Once we have generated the initial graph for a theory, we need a mechanism for marking additional nodes to represent the inferences the graph supports. Let’s look at some examples that illustrate some of the features this mechanism should have.

Example 1 (Tweety Triangle) Birds normally fly and penguins normally don’t, but penguins are birds. Tweety is a penguin. The corresponding defeasible theory is $\{Bx \rightarrow Fx, Px \rightarrow Fx, Px \rightarrow Bx, Pt\}$. But what we are really interested in is the corresponding propositional theory $\{b \rightarrow f, p \rightarrow f, p \rightarrow b, p\}$. The initial graph is shown in Figure 1.

In the Tweety Triangle, we begin with a single node marked $+$. Only a satisfied positive arrow points toward $b$ (Tweety is a bird). So we have evidence that Tweety is a bird and no evidence to the contrary. We mark node $b$ with $+$. Now we have conflicting evidence about whether Tweety flies: a satisfied positive arrow and a satisfied negative arrow pointing toward node $f$. We resolve the conflict by noting that there is a satisfied positive arrow pointing from $p$ to $b$. So we could infer (as we just did) that Tweety is a bird from the information that he is a penguin. Given the knowledge represented in the graph, $p$ provides more specific information than $b$. So $p \rightarrow f$ is the superior arrow and we mark $f$ with $-$. 

Example 2 (The University Student) Adults are adults, normally are not employed, and normally do not support themselves. Jane is a university student and Jane is employed. The propositional version of this theory is $\{u \rightarrow e, e \rightarrow s, u \rightarrow u, u \rightarrow \sim e, u \rightarrow \sim s, u, e\}$. The initial graph is shown in Figure 2.

In Example 2, we begin with $u$ and $e$ marked $+$. We only have a satisfied positive arrow pointing toward $a$; so we conclude that Jane is probably an adult and mark $a$ with $+$. Now we have conflicting evidence about $s$. There is a path from $u$ to $e$ made up entirely of satisfied positive arrows, which suggests that $u$ is more specific than $e$. But we also have a satisfied negative arrow directly from $u$ to $e$. So the argument to show that $u$ is more specific than $e$ is itself defeated. We must mark $s$ with $\sim$.

Example 3 (The Election) The Right Party and
the Left Party will enter candidates in the Presidential election. Presumably, the Rightists will nominate Arnold and not Barber or Cook. But the Rightists likely will not nominate Arnold if Douglas does not support him. If the Rightists don’t nominate Arnold, apparently they will nominate Barber. Douglas is unlikely to support either Arnold or Barber if Cook votes for Douglas’ crime bill, and apparently Cook will vote for Douglas’ bill. Olson is expected to win the Leftist nomination if she runs; but apparently she will stay out of the race. If Olson is not a candidate, the Leftists will likely nominate Nelson. And if the Leftists nominate neither Nelson nor Olsen, they will likely go with Miller. Olson is expected to win the Leftist nomination if she runs; but apparently she will stay out of the race. If Olson is not a candidate, the Leftists will likely nominate Nelson. If Olson is not a candidate, the Leftists will likely nominate Nelson. Nelson should win.

Barber should take the election from Nelson, but Nelson should win if Douglas doesn’t support Barber. The defeasible theory describing this example is {Jra, =I=b, =bG waj+Ta, Wa=N-b, vf+sa, v+sb, \(\Rightarrow v, w, =I-c, co, \sim co, \sim (ln) \Rightarrow ln, ra\Rightarrow or, rb\Rightarrow or, \{or, lo\} \Rightarrow pa, \{rb, ln\} \Rightarrow pb, \{rb, ln\} \Rightarrow pm, \{rb, ln, \sim sb\} \Rightarrow pm, \{rb, ln, \sim sb\} \Rightarrow pb}. The initial graph is shown in Figure 3.

The Election is a fairly complex example. We first mark all the T nodes +. Now rc, v, and co each have exactly one satisfied arrow pointing toward them. We mark v with + and we mark rc and co with -. But since co is marked -, we mark \(\sim co\) with +. This should allow us to mark ln with +. Furthermore, since v is marked + and the only arrows pointing toward sa and sb are negative arrows with v as antecedent, we mark sa and sb with -. But then we should mark \(\sim sa\) with +. ra now has a satisfied positive and a satisfied negative arrow pointing toward it. But T contains no information and thus must be less specific than any other antecedent. So by specificity we mark ra with + and \(\sim ra\) with +. Similarly, we can now mark rb with +. Evidently Barber will be the Right Party candidate and Nelson will be the Left Party candidate.

The node labeled or in our graph represents the proposition that the Rightists nominate either Arnold or Barber. By treating this disjunction as an atomic proposition, we sneak a disjunction into our graph. Since we only have positive arrows pointing toward or and one of them is satisfied, we mark or with +. Now we can mark \{rb, ln\} and \{rb, ln, \sim sb\} with +. Since the latter is clearly more specific than the former, we mark pm with + and pb with -. It appears that Nelson will be the next President. We mark all the other nodes with ? since we cannot show them to be either evidently true or evidently false.

These examples use presumptions and other defeasible rules almost exclusively. In fact, strict rules and defeaters do not occur that often in natural examples. If we had a strict and a defeasible arrow pointing toward the same conclusion, one positive, one negative, and both satisfied, we should prefer the strict arrow. This is built into our defeasible logic and should be a part of our inference mechanism for defeasible graphs. For more patterns of defeasible arguments, see [9].

5 Extensions of defeasible graphs

We define monotonic extensions of the initial graph for a defeasible theory T, intuitively, as graphs generated by marking nodes which are strictly derivable from the theory T.

Definition 11 Let T be any defeasible theory. Then G is a monotonic extension of GT iff either

1. G = GT, or

2. for some monotonic extension G* of GT, there is a satisfied A+p in G* (A+p in G*) such that p is unmarked in G* and G is the result of marking p with + (-) in G*.

Definition 12 Let T be a defeasible theory and let G be a monotonic extension of GT. Then G is a maximal monotonic extension of GT iff for every monotonic extension G* of GT and every node p in G*, if p is marked in G*, then p is marked similarly in G.

It doesn’t matter in what order we mark the strict consequences of a theory in the corresponding defeasible graph. Once a strict arrow in a graph is satisfied, nothing can prevent us from marking the consequent of the arrow either + or -. Thus, we get the following theorem.

Theorem 3 For each finite, defeasible theory T, there exists a unique maximal monotonic extension of GT.

Definition 13 GL is the maximal monotonic extension of the defeasible theory T.

Each monotonic extension of GT corresponds to the result of finitely many applications of the rule M+ to T. GM is the result of applying M+ repeatedly until there is no further opportunity to apply it.

Theorem 4 If T is a finite defeasible theory and p is an atom, then
Definition 14 For each defeasible theory \( T \), \( G_T^T \) is exactly like \( G_M^T \) except that
1. if \( T \) is in \( G_T^T \), then \( T \) is marked + in \( G_T^T \), and
2. if a literal \( p \) occurs unmarked in \( G_M^T \) and there is no positive (negative) link which has \( p \) as its head in \( G_M^T \), then \( p \) (\(-p\)) is marked ? in \( G_T^T \).

Condition (1) in Definition 14 merely notes that the node marked \( \top \) is true since \( \top \) just means ‘true’. This is just a convenient device to satisfy all presumptions. Condition (2), on the other hand, corresponds to applying rule \( SD\top \) as many times as possible to our graph after we have marked all the monotonic consequences. If a node is marked ?, we know that we will never be able to mark it +. So we can establish at this point that certain links are failed.

Theorem 5 If \( T \) is a finite defeasible theory, \( p \) is a literal in \( G_T^T \), and \( p \) is marked ? in \( G_T^T \), then \( T \vdash p \), and \( T \vdash \neg p \).

Definition 15 Let \( T \) be any defeasible theory. Then \( G \) is a defeasible extension of \( G_T^T \) iff any of the following hold.
1. \( G = G_T^T \).
2. For some defeasible extension \( G^* \) of \( G_T^T \) and some atom \( p \), \( G \) is the result of marking \( p \) with + (-) in \( G^*_M \), and there is \( A \vdash p \) (\( A \vdash \neg p \)) in \( G^*_M \) such that
   (a) \( A \) is satisfied in \( G^*_M \),
   (b) for all \( B \vdash p \) (\( B \vdash \neg p \)) in \( G_T^T \), \( B \) is failed in \( G^*_M \).
3. For some defeasible extension \( G^* \) of \( G_T^T \) and some atom \( p \), \( G \) is the result of marking \( p \) with + (-) in \( G^*_M \), and there is \( A \vdash p \) (\( A \vdash \neg p \)) in \( G_T^T \) such that
   (a) \( A \) is satisfied in \( G^*_M \),

Figure 3: The Election
6 Soundness and completeness

Theorem 6 [Soundness] Let $T$ be a defeasible theorem, $p$ an atom, and $G$ a defeasible extension of $G^T$.

1. If $p$ is marked $+$ in $G$, then $T \vdash p$.
2. If $p$ is marked $-$ in $G$, then $T \vdash \neg p$.
3. If $p$ is marked $?$ in $G^*$, then $T \vdash p$.
4. If $\neg p$ is marked $?$ in $G^*$, then $T \vdash \neg p$.

Theorem 7 [Completeness, first version] Let $T$ be a finite defeasible theory such that $G^T$ is acyclic, and let $p$ be any atom.

1. If $T \vdash p$, then there is a defeasible extension $G^*$ of $G^T$ such that $p$ is satisfied in $G^*$.
2. If $T \vdash \neg p$, then there is a defeasible extension $G^*$ of $G^T$ such that $p$ is marked $-$ in $G^*$.
3. If $T \vdash p$, then there is a defeasible extension $G^*$ of $G^T$ such that $p$ is failed in $G^*$.
4. If $T \vdash \neg p$, then there is a defeasible extension $G^*$ of $G^T$ such that $p$ is marked $+$ in $G^*$ or $\neg p$ is marked $?$ in $G^*$.

Definition 18 Let $T$ be a defeasible theory and let $G^T$ be a maximal defeasible extension of $G^T$. Then $G^T$ has a unique maximal defeasible extension.

Theorem 9 [Completeness, second version] Let $T$ be a finite defeasible theory such that $G^T$ is acyclic, and let $p$ be any atom.

1. If $T \vdash p$, then $p$ is satisfied in $G^T_D$.
2. If $T \vdash \neg p$, then $p$ is marked $-$ in $G^T_D$.
3. If $T \vdash p$, then $p$ is failed in $G^T_D$.
4. If $T \vdash \neg p$, then $p$ is marked $+$ in $G^T_D$ or $\neg p$ is marked $?$ in $G^T_D$. 

Conditions (a), (3), and (4) in Definition 15 correspond roughly to the rules $SS^+$, $SD^T$, and $SD$ in our defeasible logic. The specificity definitions could be inserted directly into conditions (3) and (4) but are stated separately here to make Definition 15 more readable.
7 Implementation

We plan to develop an ABS in LPA-Prolog running under Microsoft Windows based on the method of defeasible logic graphs described in this paper. LPA-Prolog provides good support for the graphical interface and an excellent environment for development of the underlying inference mechanism. The system will use a drag-and-drop method to build a graph from standard components. We plan to use color to distinguish the different kinds of links in the graph and to indicate the markings of the nodes. We think this use of color should make it easy for the user to understand the graph quickly. The user will be able to attach both a short descriptor and text of moderate length to a node. The short descriptor will appear in the graph, but the longer text will be readily available. A table of literals will also be built together with their marks. Users will explore different scenarios by marking different initial sets of assumptions with + or −.

The system will build the defeasible theory internally as the graph is built. It will be able to complete the marking of the graph at the user's command, identify nodes which can be marked at each step, or deduce the value of a given node selected by the user. An advantage of the proposed ABS is that it will be sound and complete with respect to defeasible logic.

The defeasible logic SD has been implemented as an extension of Prolog called d-Prolog [14]. An important difference between d-Prolog and the method of defeasible graphs described here is that d-Prolog is a backward-chaining inference engine while an inference mechanism based on defeasible graphs would be forward-chaining.

Other systems which might be described as ABSs ([1, 2]) include a hypertext system. Documents containing the underlying information and knowledge are linked to components in the user's graphical representation of propositions and their support relations. We do not plan a hypertext component in our prototype ABS. In principle, this could be added later or our prototype could provide an alternative defeasible inference system for existing systems that support such hypertext links.

References


