Encyclopedia of Cognitive Science
Non-monotonic Logic (Article 62)

Michael Gelfond
Professor, Department of Computer Science
Texas Tech University
Box 43104
Lubbock, TX 79409-3104
Office: (806) 742-3527
Fax: (806) 742-3519
michael.gelfond@coe.ttu.edu

Richard Watson
Assistant Professor, Department of Computer Science
Texas Tech University
Box 43104
Lubbock, TX 79409-3104
Office: (806) 742-3527
Fax: (806) 742-3519
rwatson@cs.ttu.edu

September 25, 2001

Non-monotonic Logic

keywords:

common-sense reasoning, logic programming, circumscription, default logic, closed world assumption

Contents

Article Definition:

A logic is called *non-monotonic* if, given a theory in the logic, adding new information to the theory may cause one to retract some conclusions which were previously made. This article illustrates why such logics are useful and outlines some of the major achievements of research in this area.

1 The non-monotonic character of common-sense reasoning

The use of formal logic as a means for computer programming was first seen in the work of Newell, Simon, and Shaw with their “Logic Theory Machine” (Newell and Simon, 1956). The work described a system for “discovering proofs for theorems in symbolic logic.” A few years later, (McCarthy 1959), John McCarthy advocated a logical approach to Artificial Intelligence (AI) which would lead to the development of non-monotonic logics. According to this approach, an intelligent agent should have knowledge of its world and its goals and the ability to use this knowledge to infer its course of action. To successfully perform tasks in a changing environment the agent should be able to make tentative conclusions based on available information but be prepared to withdraw some of these conclusions at a later time when further information becomes available. For instance, I may know that my car is normally in working condition and plan to use it tomorrow for driving to work. Suppose in the morning I discover that the lights were left on and the battery of the car is dead. I then have to abandon my previous plan and look for alternative ways to get to work. The logic-based approach to AI suggests that a mathematical model of an agent capable of such behavior should contain a formal language capable of expressing common-sense knowledge about the world, a precise characterization of valid conclusions which can be derived from theories stated in this language, and a means which will allow the agent to arrive at these conclusions. In the late seventies AI researchers tried to build such a model in the framework of classical logic. They quickly discovered that the task is far from being trivial. The difficulty is rather deep and is related to the so-called monotonicity of the entailment relation of this logic. An entailment relation, $\approx$, between sentences of some logical language, $\mathcal{L}$, is called monotonic if for any formulae A, B, and C of $\mathcal{L}$, if $A \approx C$ then $A, B \approx C$ (i.e. if A entails C then so does any extension of A). Monotonicity is natural for classical logic, which was primarily aimed at formalization of mathematical reasoning where axioms are rarely expanded and inferences are long and complex. In mathematics, truth is not invalidated by the addition of new axioms; once proven a theorem stays proven. In common-sense reasoning, additions to an agent's knowledge are frequent and inferences are usually short but are often based on tentative conclusions. These properties show the inherent non-monotonicity of this type of reasoning and suggest that it may be better modeled by logics with non-monotonic entailment relations. Early definitions of such relations are connected with the attempts to clarify reasoning in taxonomic hierarchies (Touretzky, 1986), the meaning of the closed world assumption in databases (Reiter, 1978), and the semantics of the negation as failure operator of logic programming (Clark, 1978). More powerful and general non-monotonic reasoning formalisms of Circumscription (McCarthy, 1980), Default Logic (Reiter, 1980), and Non-monotonic Modal Logics (McDermott and Doyle, 1980 and Moore, 1985) appeared almost simultaneously in the early eighties.

2 Taxonomic Hierarchies

Often the knowledge of a reasoning agent contains a collection of classes of objects organized in a taxonomic hierarchy. In the hierarchies pictured in figure 1, $c_1 \ldots c_4$ denote classes of objects, $c_i \rightarrow c_j$ indicates that c_i is a proper subclass of c_j ($c_i \subset c_j$) and $x \rightarrow c$ states that x is a member of c ($x \in c$). Double arrows, such as in figure 1(a), are used to link classes to properties; $c \Rightarrow p$ means that every element of class c has property p. Hierarchies containing only the elements mentioned above are called strict. They are often used to compactly encode the hierarchical structure of a domain. To establish that x has property p (denoted by the formula $p(x)$) it is sufficient to find a path from x to p. If no such path exists then x does not have property p. This reasoning can be easily justified by translating a hierarchy, H, into a first-order theory, $T(H)$, such that for every object, x, of the hierarchy $T(H) \models p(x)$ iff H contains a path from x to p. Here $\models$ denotes the entailment relation of classical logic.

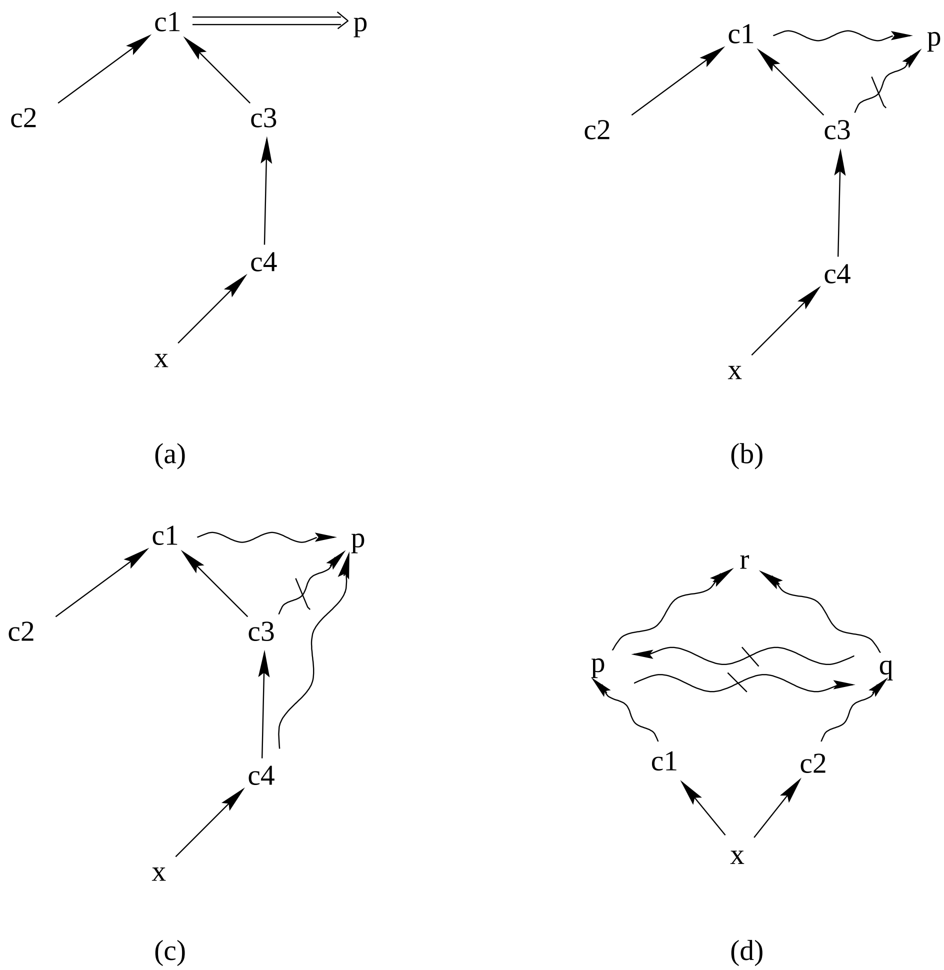

figure 1

The situation changes dramatically when double arrows from classes to properties are changed to the squiggly ones seen in hierarchies (b) and (c) from figure 1. Here a link $c \rightsquigarrow p$ says that elements of class c *normally* satisfy property p. Similarly, $c \not\rightsquigarrow p$ indicates that *normally* elements of c do not satisfy this property. Statements of this form are called *defaults*. They do not occur in mathematics but seem to constitute a large portion of our common-sense knowledge about the world. One can say that a substantial part of our education consists of learning defaults and their exceptions as well as various ways of reasoning with this knowledge. Given the hierarchy from figure 1(b) a rational agent will have no difficulty in following the path $x \rightarrow c_4 \Rightarrow c_3 \not\rightsquigarrow p$ which leads to the conclusion that x does not satisfy p. This is despite the fact that the path $x \rightarrow c_4 \Rightarrow c_3 \Rightarrow c_1 \rightsquigarrow p$ corresponds to an argument which contradicts this conclusion. This seems to happen because of the use of a common sense idea known as the Inheritance Principle (Touretzky, 1986) which states that, in our reasoning, *more specific information prevails over less specific information*. Since c_3 is a proper subset of c_1, we know that the default $c_3 \not\rightsquigarrow p$ is more specific than $c_1 \rightsquigarrow p$ which allows us to prefer the first argument. It is easy to see that inheritance reasoning is non-monotonic. Expanding the hierarchy by a link $c_4 \rightsquigarrow p$ (see figure 1(c)) will lead to the creation of a new preferred path, $x \rightarrow c_4 \rightsquigarrow p$. Given this new information we will be forced to change our view and conclude that x satisfies p after all. The Inheritance Principle was one of the first useful principles of common-sense reasoning discovered by AI researchers. Attempts to formalize this principle (i.e. to precisely characterize the correct conclusions which can be drawn by a rational agent whose knowledge is represented by an inheritance hierarchy) produced two distinct approaches to the problem. Direct theories of inheritance focus on paths of the network, viewed as possible arguments, and on defining relative strength of these arguments.

For the hierarchy from figure 1(b), the argument $x \rightarrow c_4 \Rightarrow c_3 \Rightarrow c_1 \rightsquigarrow p$ is defeated by the argument $x \rightarrow c_4 \Rightarrow c_3 \not\rightsquigarrow p$ which is not defeated by any other argument. Hence the conclusion $p(x)$ is justified. In a more complex hierarchy (figure 1(d)) properties p and q are typically mutually exclusive but elements satisfying either one of them normally satisfy property r. Here the neither the argument $x \rightarrow c_1 \rightsquigarrow p$ nor $x \rightarrow c_2 \rightsquigarrow q \not\rightsquigarrow p$ is more specific and hence they defeat each other. A similar case can be seen for the property q and therefore we can conclude neither $p(x)$ nor $q(x)$. The answer given by this hierarchy to the query $r(x)$ depends on the precise definition of plausible counterargument. A 'sceptical' reasoner will argue that since the truth of $p(x)$ and $q(x)$ are unknown we should refrain from making any conclusions about $r(x)$. A 'credulous' reasoner can take a view that the net from figure 1(d) sanctions the conclusion $p(x) \vee q(x)$ and hence $r(x)$ must be true. A detailed discussion of direct theories of inheritance can be found in (Horty, 1994). Indirect approaches to inheritance are based on mapping the network together with the form of inheritance principle used in its semantics to more general theories of non-monotonic reasoning. Some recent insight on the relationship between direct and indirect approaches can be found in (You et al, 1999).

3 The closed world assumption

The following example illustrates another typical situation requiring non-monotonic reasoning. Suppose the list of faculty of a small computer science department is posted on the wall of the main office. One may naturally assume that the list is complete, i.e. if Michael is not mentioned on it then he is not a faculty member in this department. This is, of course, only an assumption. It is possible that the name of a recently hired faculty member was not yet added to the list. Assumptions like this are called closed world assumptions (CWA) (Reiter, 1978). They are frequently used by people acting in every day situations. Such assumptions are also built in the semantics of databases and logic programming languages such as Datalog and Prolog. In databases, a table describing a relation, r, contains the objects satisfying this relation. Objects which do not occur in the table are assumed not to satisfy r. In logic programming, an answer *no* to a query q is often interpreted as "q is false" (instead of a more precise but less useful "Prolog interpreter cannot prove q from information given in a program"). The closed world assumption is expressible in practically all of the general purpose non-monotonic formalisms, including those which are discussed below.

4 Non-monotonic reasoning in logic programming

Some of the earliest non-monotonic reasoning systems were developed in the framework of logic programming. Originally, logic programs were viewed as collections of clauses (or rules) of the form

$$p \leftarrow \Gamma$$

which is read as p if Γ. Here the head, p, is an atom and the body, Γ, is a sequence of atoms. A clause with an empty body is called a *fact*. With the advent of practical logic programming languages the rules became more complex. For instance, in the logic programming language Prolog, the body of a rule is allowed to contain expressions of the form *not* q interpreted as a statement "Prolog interpreter failed all the attempts to prove q". Note that *not* is different than negation in classical logic, $\neg$, in that $\neg q$ is interpreted as a statement "q is false". As a result a program, π_0, consisting of the two rules

$$p(X) \leftarrow not\ q(X); \quad p(X) \leftarrow r(X)$$

will answer *yes* to a query $p(a)$. The Prolog interpreter will attempt to prove $q(a)$, fail to do so, and conclude $p(a)$. If we extend π_0 by a new fact, $q(a)$, the previous conclusion will be withdrawn and the answer to $p(a)$ will become *no*. The connective *not* was called "negation as finite failure" and viewed primarily as a procedural device. The first declarative semantics of this connective was suggested in (Clark, 1978). Under this semantics, the collection of rules with the head $p(X)$ in π_0 is viewed as a definition of p - a shorthand for the formula

$$\forall X\ (p(X) \equiv \neg q(X) \vee r(X)). \tag{1}$$

This idea is used to translate a program π into a set $comp(\pi)$ of first-order formulae called the *predicate completion* of π. The theory $comp(\pi_0)$ consists of statement 1 above, as well as the statements

$$\forall X\ \neg r(X)$$

and

$$\forall X\ \neg q(X) \tag{2}$$

and the collection of equality axioms guaranteeing that distinct names in the language represent distinct objects of the domain. The completion, $comp(\pi_1)$, of the program $\pi_1 = \pi_0 \cup \{q(a\}$ is obtained by replacing the axiom 2 in $comp(\pi_0)$ by $\forall X\ q(X) \equiv X = a$. By definition a logic program, π, entails a literal, l, (not containing variables) iff l is classically entailed by $comp(\pi)$. Obviously, $comp(\pi_0) \models p(a)$ but $comp(\pi_1) \models \neg p(a)$.

Work on the semantics of logic programs continued in several directions. The meaning of *not* was refined to make it less dependent on the particular inference mechanisms associated with the Prolog interpreter, as in (Gelfond and Lifschitz, 1988), and to allow abductive reasoning, as in (Kakas et al, 1992 and Denecker and De Schreye, 1992). These approaches were generalized to accommodate programs with rules which are formed using more complex formulas. The resulting logical languages and entailment relations provide powerful means for representing and reasoning with common-sense knowledge. Suppose we wish to represent the facts that Tweety is a bird and birds normally fly. This can be done by the program consisting of two rules:

$fly(X) \leftarrow bird(X), not\ \neg fly(X)$

$bird(tweety).$

Since there is no reason to believe that $tweety$ does not fly, the program concludes $fly(tweety)$. If we learn that $tweety$ is not a flying bird and expand our program by a new fact, $\neg fly(tweety)$, the resulting program will be consistent and entail that Tweety does not fly.

5 Circumscription

The basic ideas and intuition behind another powerful non-monotonic system called circumscription were described by John McCarthy in (McCarthy, 1977) and later formalized in (McCarthy, 1980). In circumscription, theories are written in classical first-order logic, however the entailment relation is not classical. A formula F is entailed by a circumscriptive theory T if it is true in all of minimal models of T. A model, M, of a theory, T, is called minimal with respect to some ordering, $<$, of models if there is no model, N, of T such that $N < M$. A circumscription policy is used to determine a particular partial ordering, $<$, used to circumscribe a theory. In the basic case a single predicate, P, is chosen to be circumscribed. Given two models, M_1 and M_2, which differ only in the interpretation of P, we say that $M_1 \leq M_2$ if the extent of P in M_1 is a subset of its extent in M_2. We write $circ(T; P) \models F$ to indicate that F is entailed from T by circumscription with the policy of minimizing P. Here $circ(T; P)$ can be viewed as a second-order formula expressing the above definition. (Recall that second-order formulas allow quantifiers over relations.)

Let us apply this idea to a version of the flying birds example above. We can attempt to express that Tweety is a bird and birds normally fly by first-order sentences

$$T = \{bird(tweety) \text{ and } \forall X\ (bird(X) \land \neg ab(X) \supset fly(X))\}$$

where $ab(X)$ means that X is abnormal with respect to flying. Obviously, classical first-order logic does not allow us to reach the desired common-sense conclusion that Tweety can fly. If, however, we use circumscription and circumscribe ab, then all minimal models under this policy contain $fly(tweety)$ and hence $circ(T; ab) \models fly(tweety)$. This basic form of circumscription is often too restrictive so many other standard circumscriptive policies have been formalized. One common policy is to specify certain predicates which are allowed to vary. In this case, models are comparable if they differ in the extent of the varied predicates as well as the circumscribed predicate. As before, the ordering of comparable models is based solely on the extent of the circumscribed predicate in the models. Suppose we add to our example the fact that penguins are birds which are abnormal with respect to flying ($penguin(X) \supset bird(X)$ and $penguin(X) \supset ab(X)$). Since a model, M_0, in which Tweety is a penguin, is minimal with respect to ordering defined by the first policy we no longer have

$circ(T; ab) \models fly(tweety)$. If we modify the policy so that *penguin* can vary, the model M_0 will not be minimal with respect to the new ordering. It is easy to check that, under this policy, T concludes $fly(tweety)$. Selection of the right circumscriptive policy lies at the heart of representing knowledge in circumscriptive theories. Even though computing consequences of circumscribed theories is generally intractable (even in propositional case), for some theories there are reasonably efficient algorithms based on reducing the circumscribed theory to a logic program or a set of first-order formulae.

6 Default logic

Another nonmonotonic formalism which uses classical first-order logic as its base is default logic (Reiter, 1980). A default theory is a pair (D, W) where W is a collection of statements of first-order logic and D is a set of default rules, i.e. statements of the form

$$\frac{A : MB_1, \ldots, MB_n}{C}$$

where A, each B_i, and C are classical formulae. A default rule can be read as, "if A is provable and each B_i is possible then conclude C," and used as a non-monotonic rule of inference. For instance, a proof of r in a default theory $T_1 = (\{\frac{q:Mr}{r}\}, \{p, p \supset q\})$ consist of a 'classical' step, deriving q, and an application of the default rule of T_1, deriving r. The second step is justified by T_1's inability to prove $\neg r$. A collection of formulae which can be derived from a default theory, T, in this fashion is called an extension of T. Extensions are often interpreted as sets of beliefs which can be formed by a rational agent on the basis of T. The default theory T_1 above has one such extension, $Cn(p, p \supset q, r)$, where $Cn(S)$ stands for the set of all 'classical' consequences of S. Default theory $T_2 = (\{\frac{r:M\neg p}{q}, \frac{r:M\neg q}{p}\}, \{r\})$

has two extensions, $Cn(r,q)$ and $Cn(r,p)$, and default theory $T_3 = \{\{\frac{:M\neg p}{p}\}, \emptyset\}$ has none. (The latter fact can be attributed to the irrationality of the default rule of T_3). The precise notion of an extension of a default theory was first given by a fixed-point construction in (Reiter, 1980). Even though some variants of Reiter's notion of an extension have been advocated by different researchers, the basic idea of the construction remains the same. Fixed-point constructions somewhat similar to that of Reiter were used in several other early non-monotonic systems. In particular they are used to define the semantics of 'modal' non-monotonic logics of McDermott and Doyle (McDermott and Doyle, 1980) and their variants (Moore, 1985). For more details see (Marek and Truszczyński, 1993).

Default logic is a powerful tool which can be used to give semantics of other non-monotonic formalisms. For instance, the inheritance hierarchy in figure 1(d) can be represented by a default theory $T_4 = \{D, W\}$ where $D = \{\{\frac{c_1(X):Mp(X)}{p(X)}; \frac{c_2(X):Mq(X)}{q(X)}; \frac{p(X):Mr(X)}{r(X)}; \frac{q(X):Mr(X)}{r(X)}\}$ and $W = \{c_1(x), c_2(x), \forall X\ p(X) \supset \neg q(X)\}\}$. (Here variable-containing rules of T_4 can be viewed as a shorthand for the set of their ground instantiations). The theory has two extension: one containing $p(x), \neg q(x), r(x)$ and another containing $q(x), \neg p(x), r(x)$. The set of valid conclusions which can be made by a 'credulous' inheritance reasoner on the basis of such hierarchies can now be defined as the set of formulae which belong to all extensions of the corresponding default theory. . Default theories can also be used to give a semantics to logic programs. A rule $r = l_0 \leftarrow l_1, \ldots, l_m, not\ l_{m+1}, \ldots, l_n$, where l's are literals, can be translated to a default rule

$$d(r) = \{\frac{l_1, \ldots, l_m : M\bar{l}_{m+1}, \ldots, M\bar{l}_n}{l_0}\}.$$

Note that $\bar{l}$ stands for the complement of l, i.e. if l is an atom then $\bar{l} = \neg l$ and if $l = \neg a$, where a is an atom, then $\bar{l} = a$. A program, π, consisting of such rules entails formula F if F belongs to all the extensions of the default theory $d(\pi) = \{\{d(r): \text{ for all } r \in \pi\}, \emptyset\}$. It can be shown that the semantics defined in this way coincides with the answer set semantics (Gelfond and Lifschitz, 1991) of logic programs. There is also a close connection with truth-maintenance systems (TMS) (Doyle, 1979) - programs which manage sets of nodes and sets of justifications. A node can be viewed as a set of formula while justifications correspond to logic programming (or default) rules. Each justification states that one node should be believed if some others are respectively believed or disbelieved. The precise mapping establishing this correspondence helped increase the understanding of what is computed by TMS's and at the same time linked theoretical work on non-monotonic reasoning with implemented reasoning systems.

It remains to be seen if the impressive power of default logic, circumscription, and other 'superclassical' non-monotonic systems will make them tools of choice for representing common-sense knowledge or if weaker but simpler languages (similar to those based on logic programming) will be preferred by knowledge engineers. In recent years a substantial progress was achieved in building practical systems capable of non-monotonic reasoning. In addition to Prolog and TMS's we now have efficient systems, such as Smodels (Niemelä and Simons, 1997), CCALC (McCain, 1997), and dlv (Citrigno et al, 1997), capable of computing answer sets of various classes of logic programs and default theories. These systems form a basis for a style of programming in which application problems are encoded so that their solutions are fully described by answer sets of the corresponding program. There are applications of this approach to classical AI problems such as planning and diagnosis. Non-monotonic systems also played an important role in the development of theories of action and change. In such theories one needs to describe causal laws which define the effects of performing actions as well as causal relations between the fluents (propositions whose values depend on time). One also needs to specify what fluents do not change. The problem of finding a compact representation of what changes and what does not is called the Frame Problem. In 1986, McCarthy suggested to solve the second part of this problem by the use of the default which states that actions normally do not change values of fluents. Attempts to solve this and other related problems motivated a substantial portion of the ongoing work on non-monotonic logics. More details on this subject can be found in (Shanahan, 1997).

7 Non-monotonicity and Probabilistic Reasoning

The formalisms discussed above allow reasoning about the truth or falsity of logical statements. In some cases the truth or falsity can be determined only with a certain degree of plausibility. AI researchers used a variety of different approaches, such as Bayesian theory, Dempster-Shafer theory, fuzzy logic, and certainty factors, to develop formalisms for reasoning with such information. These formalisms are closely related to work on non-monotonic reasoning. A collection of important papers on probabilistic reasoning methods can be found in (Pearl and Shafer, 1990). To illustrate the use of such approaches, we consider Bayesian belief networks. Belief networks are applicable in situations when plausibility of the truth or falsity of a statement can be given by the mathematical probability of the corresponding event.

Consider the following situation. A person has a burglar alarm in their house and lives in an area where burglaries are fairly infrequent. If there was a burglary, the alarm would probably sound. False alarms sometimes occur as well. Suppose that, while at work, the person gets a call from a neighbor who says that the alarm is sounding. Suppose the person also knows that, while this particular neighbor could be trusted to call if the alarm really was on, the neighbor is also practical joker and may say the alarm is ringing as a prank. The person has a second neighbor who he can ask about the alarm. This neighbor lives farther away from the person's house and, because of the distance, may not hear the alarm even if it is active. A belief network formalizing information from this example is given in figure 2.

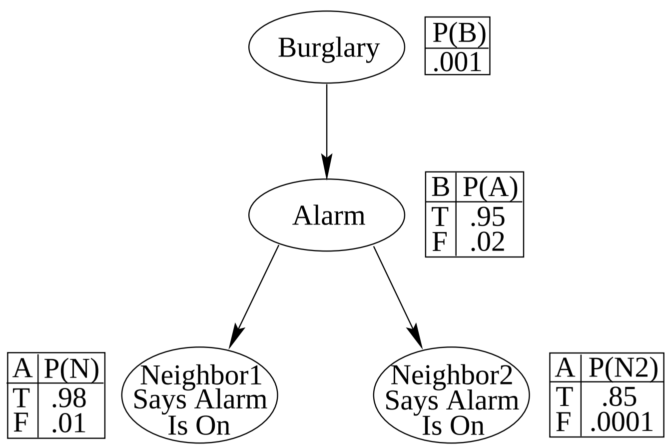

figure 2

In the figure, events are depicted by ovals. Beside each oval is a table giving the conditional probabilities of the event. The statement, from the example above, that "burglaries are fairly infrequent" is expressed by assigning a very low probability to the event (i.e. P(B) = .001). An arrow from event X to event Y indicates that Y can be caused by X. The chances of this happening are given by the conditional probabilities in the corresponding table. In figure 2, the table next to the alarm event states that the alarm goes of 95% of the time when there is a burglary and 2% of the time when there is not one. Using the information given in the belief network, algorithms based on Bayesian theory can be used to compute the probability of an event given knowledge of the occurrences of other events. The agent's belief is then based on the computed probability. For example, the probability that the alarm sounded given the fact that the first neighbor called is 68% and hence the agent would believe that alarm was on. This approach is non-monotonic in that if the agent gets further information it will most likely change the computed probability and may therefore alter his beliefs. In the example given, if the second neighbor calls and says that they do not hear the alarm then the computation returns only a 24% probability that the alarm is on and therefore the agent changes his belief. A more complete discussion of the relationship between non-monotonic logics and probabilistic reasoning methods can be found in (Goldszmidt and Pearl, 1996).

References

Clark K (1978) Negation as failure. In: Gallaire H and Minker J (eds) *Logic and Data Bases*, pp. 293–322, New York:Plenum.

Citrigno S, Eiter T, Faber W, Gottlob G, Koch C, Leone N, Mateis C, Pfeifer G, and Scarcello F (1997) The dlv system: Model generator and application frontends. In: *Proceedings of the 12th Workshop on Logic Programming*, pp. 128–137.

Denecker M and De Schreye R (1992) SLDNFA: an abductive procedure for normal abductive logic programs. In: *Proceedings of JICSLP*, pp. 686–700.

Doyle J (1979) A truth maintenance system. *Artificial Intelligence* 12(3):231–272.

Gelfond M and Lifschitz V (1988) The stable model semantics for logic programming. In: Kowalski R and Bowen K (eds) *Logic Programming: Proceedings of the 5th International*

Conference and Symposium, pp. 1070–1080.

Gelfond M and Lifschitz V (1991) Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Goldszmidt M and Pearl J (1996) Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence* 84(1,2):57–112.

Horty J (1994) Some direct theories of nonmonotonic inheritance. In: Gabbay D, Hogger C, and Robinson J (eds) *Handbook of Logic in Artificial Intelligence and Logic Programming - Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning.* New York:Oxford University Press.

Kakas A, Kowalski R, and Toni F (1992) Abductive logic programming. *Journal of Logic and Computation* 2(6):719–771.

Marek V and Truszczyński M (1993) *Nonmonotonic Logic*, Berlin:Springer-Verlag Berlin.

McCain N (1997) *Causality in Commonsense Reasoning about Actions*, PhD Thesis, The University of Texas at Austin.

McCarthy J (1959) Programs with common sense. In: *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pp 75–91, London:Her Majesty's Stationary Office.

McCarthy J (1977) Epistemological problems of artificial intelligence. In: *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pp. 1038–1044.

McCarthy J (1980) Circumscription: a form of non-monotonic reasoning. *Artificial Intelligence* 13(1,2):27–39, 171–172.

McDermott D and Doyle J (1980) Non-monotonic logic I. *Artificial Intelligence* 13(1–2):41–72.

Moore R (1985) Semantical considerations on nonmonotonic logic. *Artificial Intelligence* 25(1):75–94.

Newell A and Simon H (1956) The logic theory machine: A complex information processing system. *IRE Transactions on Information Theory* IT-2(3), pp. 61–79.

Niemelä I and Simons P (1997) Smodels - an implementation of the stable model and well-founded semantics for normal logic programs. In *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning*, pp. 420–429.

Pearl J and Shafer G (eds) (1990) *Readings in Uncertain Reasoning*. San Mateo:Morgan Kaufmann.

Reiter R (1978) On closed world databases. In: Gallaire H and Minker J (eds) *Logic and Data Bases*, pp. 119–140, New York:Plenum.

Reiter R (1980) A logic for default reasoning. *Artificial Intelligence* 13(1–2):81-132.

Shanahan M (1997) *Solving the Frame Problem: a mathematical investigation of the common sense law of inertia*. Cambridge:MIT Press.

Touretzky D (1986) *The Mathematics of Inheritance Systems*, Los Altos:Morgan Kaufmann.

You J, Wang X, and Yuan L (1999) Compiling defeasible inheritance networks to general logic programs. *Artificial Intelligence* 113(1–2):247–268.

Bibliography

Brewka G, Dix J, and Konolige K (1997) *Nonmonotonic Reasoning: An Overview.* Stanford:CSLI Publications.

Gabbay D, Hogger C, and Robinson J (eds) (1994) *Handbook of Logic in Artificial Intelligence and Logic Programming - Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning.* New York:Oxford University Press.

Marek V and Truszczyński M (1993) *Nonmonotonic Logic*, Berlin:Springer-Verlag Berlin.

McCarthy J (1990) *Formalizing Common Sense: Papers by John McCarthy*, Lifschitz V (eds). Norwood:Ablex Publishing Company.

Shanahan M (1997) *Solving the Frame Problem: a mathematical investigation of the common sense law of inertia.* Cambridge:MIT Press.

Glossary

Abductive Reasoning - The process of finding explanations for observations in a given theory.

Classical Logic - Logic whose statements are formed from atoms (sentences expressing relations between objects), boolean connectives, and quantifiers.

First-Order Logic - Classical logic in which quantification is limited to objects. ($\forall X p(X)$ is a sentence of first-order logic while $\forall P P(0)$ is not).

Formal Language - A language with a precisely defined syntax and semantics. Typical examples include mathematical and programming languages.

Prolog Interpreter - A reasoning system used by the programming language Prolog to answer queries to Prolog programs.

Semantics - The meaning of language. A formal semantics of a logical language provides the meaning of its logical connectives and characterizes the sets of consequences of its theories.