A Causal Logic of Logic Programming

Alexander Bochman
Computer Science Department,
Holon Academic Institute of Technology, Israel
e-mail: bochmana@hait.ac.il

Abstract

The causal logic from (Bochman 2003b) is shown to provide a natural logical basis for logic programming. More exactly, it is argued that any logic program can be seen as a causal theory satisfying the Negation As Default principle (alias Closed World Assumption). Moreover, unlike well-known translations of logic programs to other nonmonotonic formalisms, the established correspondence between logic programs and causal theories is bidirectional in the sense that, for an appropriate causal logic, any causal theory is reducible to a logic program. The correspondence is shown to hold for logic programs of a most general kind involving disjunctions and default negations in heads of the rules. It is shown also to be adequate for a broad range of logic programming semantics, including stable, supported and partial stable models. The results strongly suggest that the causal logic can serve as a (long missing) logic of logic programming.

Introduction

In order to fulfil the role of a general-purpose computational formalism for knowledge representation and reasoning, logic programming should have a clear logical (declarative) basis that would allow a systematic and transparent representation of the real world information. From its very beginning, logic programming was based on the idea that *program rules* must have both a procedural and declarative meaning. Thus, the declarative meaning of a definite program rule was taken to be the meaning of the corresponding classical implication. This understanding has been challenged, however, with the introduction of negation as failure as a replacement of the classical negation in logic programs. At the first stage, Clark's completion (Clark 1978) and Reiter's Closed World Assumption (Reiter 1978) were commonly accepted as giving more adequate interpretations of logic programs and deductive databases. A formal exploration of these interpretations have led to three-valued completion (Fitting 1985), and to supported models of logic programs (Apt, Blair, & Walker 1988). Nevertheless, for some time these developments peacefully coexisted with an opinion that they reflect a purely pragmatic concession, and an ideal solution should still consist in the full use of classical negation in programs and queries (cf. (Shepherdson 1988)).

Things have changed, however, with the discovery that logic programs with negation as failure allow to represent significant parts of nonmonotonic reasoning. Moreover, general nonmonotonic formalisms inspired a new kind of semantics for logic programs, the stable and answer set semantics (Gelfond & Lifschitz 1988; 1991). These developments have advanced logic programming to the role of a general computational mechanism for knowledge representation and nonmonotonic reasoning (see, e.g., the overviews (Apt & Bol 1994; Baral & Gelfond 1994)). In addition, Przymusinski has developed a comprehensive semantic framework of partial stable models, with the stable and well-founded semantics as special cases.

The idea of a dual interpretation of logic programs, procedural and declarative, persisted in all these developments. However, the original question *"What is a declarative meaning of a program rule?"* has been replaced with the global question *"What is a declarative meaning of a logic program?"*, and an answer to this latter question has been commonly thought as settled by assigning logic programs some nonmonotonic semantics. Of course, there were reasons for this shift, since already the completion of a logic program does not say much about meaning of a single program rule.

Unfortunately, the above solution to the problem of determining declarative meaning of logic programs turns out to be problematic, the main source of the problem being that the nonmonotonic semantics of logic programs are *global*, and hence do not give a direct interpretation of program rules, but only of the program as a whole. As was rightly noted already in (Shepherdson 1988), this solution does not allow to see the written text of a program as its declarative meaning. For example, adding new rules to a program does not necessarily mean that the extended program contains more information (since the associated semantics are nonmonotonic). Worse still, quite diverse programs can have the same 'meaning' according to this understanding, witness such programs as $\{p \leftarrow q\}$ and $\{q \leftarrow p\}$ (both have the empty least model). Consequently, the association of logic programs with a meaning of this kind turns out to be useless for most purposes we could possibly have in invoking this notion (be it knowledge representation or program updates). As a result, the programming methodology and knowledge representation in logic programming lack a systematic basis, and are largely an art based on accumulated experience,

contrary to initial aspirations behind logic programming.

A general way of resolving the above problem that we suggest in this study is based on a clear separation between logical and nonmonotonic aspects of reasoning with logic programs. Once such a separation is made, a declarative meaning of a logic program (and even of a single program rule) could be identified with its logical meaning naturally determined by the associated monotonic logic and its semantics. The nonmonotonic semantics will play no direct role in this description; the declarative meaning will determine the nonmonotonic semantics of a program, but not vice versa.

A more specific idea that will be developed in the paper is that the causal logic introduced in (Bochman 2003b) allows to provide a natural and exact logical basis for logic programming. The causal logic has been shown to be adequate for reasoning with causal theories (McCain & Turner 1997; Giunchiglia *et al.* 2003), as well as for providing a syntax-independent representation of abductive reasoning (Bochman 2004). In this respect, the present study can also be seen as a further justification of our claim that the causal logic is a powerful general-purpose formalism for nonmonotonic reasoning that suggests a viable alternative to other nonmonotonic formalisms (such as default logic).

Causal inference relations

We will assume that our basic language is an ordinary propositional language with the classical connectives and constants $\{\wedge, \vee, \neg, \rightarrow, \mathbf{t}, \mathbf{f}\}$. In addition, $\vDash$ and Th will stand, respectively, for the classical entailment and the associated logical closure operator.

A *causal rule* is a rule of the form $A \Rightarrow B$, where A and B are classical propositions. Informally, such a rule says that, whenever A holds, it causes (or explains) B.

Definition 1. A *causal inference relation* is a relation $\Rightarrow$ on the set of propositions satisfying the following conditions:

(Strengthening) If $A \vDash B$ and $B \Rightarrow C$, then $A \Rightarrow C$;
(Weakening) If $A \Rightarrow B$ and $B \vDash C$, then $A \Rightarrow C$;
(And) If $A \Rightarrow B$ and $A \Rightarrow C$, then $A \Rightarrow B \wedge C$;
(Or) If $A \Rightarrow C$ and $B \Rightarrow C$, then $A \vee B \Rightarrow C$;
(Cut) If $A \Rightarrow B$ and $A \wedge B \Rightarrow C$, then $A \Rightarrow C$;
(Truth) $\mathbf{t} \Rightarrow \mathbf{t}$;
(Falsity) $\mathbf{f} \Rightarrow \mathbf{f}$.

Though causal relations satisfy most of the rules for the classical entailment, their distinctive feature is that they are not reflexive, that is, do not satisfy the postulate $A \Rightarrow A$.

The rule Or permits reasoning by cases; this feature displays one of the main advantages of causal reasoning as compared with, say, default logic. Speaking generally, the rule indicates that the causal logic is an *objective* (extensional) logical system, a system of reasoning about the world. In this respect, it is similar to classical logic (which is also extensional), and distinct from modal (intensional) formalisms that deal primarily with beliefs and knowledge we have about the world.

An important property of causal relations is the following decomposition of causal rules:

Lemma 1. *Any causal rule $A \Rightarrow B$ is equivalent to a pair of rules $A \wedge \neg B \Rightarrow \mathbf{f}$ and $A \wedge B \Rightarrow B$.*

By a *constraint* we will mean a causal rule of the form $A \Rightarrow \mathbf{f}$. Such constraints correspond to state constraints in action theories, and their role consists in describing ordinary factual information about the world: a fact A can be expressed as a constraint $\neg A \Rightarrow \mathbf{f}$. On the other hand, causal rules $A \wedge B \Rightarrow B$ are logically (factually) trivial, but they play an important explanatory role in causal reasoning. Namely, they say that in any situation in which A holds, we can freely accept B, since it is self-explanatory. Such rules could be called *explanatory* causal rules. Now the above lemma says that any causal rule can be decomposed into a (non-causal) constraint and an explanatory rule. This decomposition separates two kinds of information conveyed by causal rules. One is a logical information that constraints the set of admissible models, while the other is an explanatory information describing what propositions are caused (explainable) in such models.

We extend causal rules to rules having arbitrary sets of propositions as premises as follows: for any set u,

$$u \Rightarrow A \;\equiv\; \bigwedge a \Rightarrow A, \text{ for some finite } a \subseteq u.$$

The above stipulation secure the *compactness* of causal inference relations. $\mathcal{C}(u)$ will denote the set of propositions caused by u, that is

$$\mathcal{C}(u) = \{A \mid u \Rightarrow A\}.$$

The operator $\mathcal{C}$ will play much the same role as the usual derivability operator for consequence relations.

A possible worlds semantics

A semantic interpretation of causal relations can be given in terms of standard possible worlds models. As usual, a *possible worlds model* is a triple $\mathbb{W} = (W, R, V)$, where W is a set of worlds, R a binary accessibility relation on W, and V a function assigning each world a propositional interpretation. Intuitively, $\alpha R \beta$ means that α and β are, respectively an initial state (input) and a possible output state of a causal inference based on a given set of causal rules. A possible worlds model will be called *quasi-reflexive* if its accessibility relation satisfies the condition:

$$\text{If } \alpha R \beta, \text{ then } \alpha R \alpha.$$

The following definition provides the notion of validity for causal rules in such models:

Definition 2. $A \Rightarrow B$ will be said to be *valid* in a possible worlds model (W, R, V) if, for any worlds α, β such that $\alpha R \beta$, if A holds in α, then B holds in β.

The canonical semantics of a causal relation is determined by worlds (maximal deductively closed sets) α for which $\mathcal{C}(\alpha) \subseteq \alpha$. These are the worlds that are closed with respect to the causal rules of $\Rightarrow$. The accessibility relation on such worlds is defined as follows:

$$\alpha R \beta \;\equiv\; \mathcal{C}(\alpha) \subseteq \alpha \cap \beta$$

The canonical semantics can be shown to be adequate for the source causal relation, and hence we obtain the following completeness result.

Theorem 2. *A relation is causal if and only if it is determined by a quasi-reflexive possible worlds model.*

A four-valued representation

It turns out that causal inference relations can also be given a four-valued semantics. According to *Belnap's interpretation* of four-valued logics (Belnap 1977), any four-valued valuation function is representable as a function assigning each proposition some *subset* of the set of classical truth-values $\{t, f\}$. By this interpretation, a four-valued assignment can be seen as a pair of ordinary classical assignments corresponding, respectively, to (independent) assignments of truth and falsity to propositions. To be more exact, for any four-valued interpretation ν, we can define the following two assignments:

$$\begin{array}{rcl} \nu \models A & \text{iff} & t \in \nu(A) \\ \nu \dashv A & \text{iff} & f \in \nu(A) \end{array}$$

As is shown in (Bochman 1998a), the above representation allows to define any four-valued connective using pairs of definitions with respect to each of the above two valuations. In particular, the four connectives $\{\vee, \sim, \neg, \mathbf{N}\}$, defined below, determine what was termed in (Bochman 1998a; 1998b) the set of *classical* four-valued functions:

$$\begin{array}{rcl} \nu \models A \vee B & \text{iff} & \nu \models A \text{ or } \nu \models B \\ \nu \dashv A \vee B & \text{iff} & \nu \dashv A \text{ and } \nu \dashv B \\ \nu \models \sim A & \text{iff} & \nu \dashv A \\ \nu \dashv \sim A & \text{iff} & \nu \models A \\ \nu \models \neg A & \text{iff} & \nu \not\models A \\ \nu \dashv \neg A & \text{iff} & \nu \not\dashv A \\ \nu \models \mathbf{N}A & \text{iff} & \nu \not\dashv A \\ \nu \dashv \mathbf{N}A & \text{iff} & \nu \dashv A \end{array}$$

The negation connectives $\neg$ and $\sim$ will be called, respectively, a *local* and *global negation*. Each of them can be used together with the disjunction to define a natural *conjunction*:

$$A \wedge B \;\equiv\; \sim(\sim A \vee \sim B) \;\equiv\; \neg(\neg A \vee \neg B).$$

The above two negations satisfy de Morgan rules with respect to disjunction and conjunction, as well as Double Negation. The third negation connective $\mathbf{N}$ will be called an *ht-negation* do its close similarity with the negation in the logic of here-and-there (see (Lifschitz, Pearce, & Valverde 2001)). As can be verified, $\mathbf{N}$ satisfies the de Morgan rules. It does not satisfy, however, the Double Negation rule, only a weaker 'Triple Negation' rule $\mathbf{NNN}A \equiv \mathbf{N}A$.

The connectives generated by the set $\{\vee, \neg\}$ will be called *local classical connectives*. A distinctive feature of such connectives is that they behave in an entirely classical way with respect to each of the two valuations determining a four-valued assignment.

Note now that any pair of worlds determines a four-valued interpretation, and vice versa. Moreover, the classical connectives defined on worlds correspond in this sense precisely to the local classical connectives. In addition, a binary accessibility relation on worlds is actually a certain set of pairs of worlds. Consequently, a possible worlds semantics can be identified with a set of four-valued interpretations determined by pairs of worlds (α, β) such that $\alpha R \beta$. Then the following definition provides the corresponding notion of validity for causal rules in the language of the local classical connectives:

Definition 3. A causal rule $A \Rightarrow B$ is *valid* in a set of four-valued interpretations $\mathbf{I}$, if $\nu \not\dashv A$ implies $\nu \models B$, for any $\nu \in \mathbf{I}$.

Let us denote by $\Rightarrow_{\mathbf{I}}$ the set of all causal rules that are valid in $\mathbf{I}$. In addition, let us say that a set of of four-valued interpretations is *quasi-reflexive*, if (viewed as a set of pairs of interpretations) it determines a quasi-reflexive relation. Then we immediately obtain

Theorem 3. *A set $\Rightarrow$ of causal rules is a causal relation iff $\Rightarrow \;=\; \Rightarrow_{\mathbf{I}}$, for some quasi-reflexive set of four-valued interpretations $\mathbf{I}$.*

The above result shows that the four-valued semantics gives a complete semantic characterization of causal inference relations. Moreover, causal relations constitute in this sense a fully expressive four-valued formalism - any four valued connective can be characterized in a systematic way by adding a suitable set of postulates for causal relations. For example, the following rules provide a complete characterization of the global negation and ht-negation in the framework of causal inference:

$$\begin{array}{cc} \neg A \Rightarrow \sim A & \sim A \Rightarrow \neg A \\ A \Rightarrow \neg\sim A & \neg\sim A \Rightarrow A \\ \neg A \Rightarrow \mathbf{N}A & \mathbf{N}A \wedge A \Rightarrow \mathbf{f} \\ A \Rightarrow \neg\mathbf{N}A & \neg\mathbf{N}A \wedge \neg A \Rightarrow \mathbf{f} \end{array}$$

Moreover, any non-local connective can be *systematically eliminated* from the causal rules. For example, whereas the equivalences below show how they can be eliminated:

$$\begin{array}{c} A \wedge \sim C \Rightarrow B \;\equiv\; A \Rightarrow B \vee C \\ A \Rightarrow B \vee \sim C \;\equiv\; A \wedge C \Rightarrow B \\ A \wedge \neg\sim C \Rightarrow B \;\equiv\; A \Rightarrow C \rightarrow B \\ A \Rightarrow B \vee \neg\sim C \;\equiv\; A \wedge \neg C \Rightarrow B \end{array}$$

$$\begin{array}{c} A \wedge \mathbf{N}C \Rightarrow B \;\equiv\; A \wedge \neg C \Rightarrow B \\ A \Rightarrow B \vee \mathbf{N}C \;\equiv\; A \wedge C \Rightarrow B \\ A \wedge \neg\mathbf{N}C \Rightarrow B \;\equiv\; A \wedge C \Rightarrow B \\ A \Rightarrow B \vee \neg\mathbf{N}C \;\equiv\; A \wedge \neg C \Rightarrow B \end{array}$$

The above equivalences show, in effect, that both these negation connectives behave similarly in heads of the causal rules, and that $\mathbf{N}$ reduces to the classical negation in the bodies of such rules. On the other hand, they show also that using each of these negations, we can move all premises to heads of causal rules:

$$A \Rightarrow B \equiv \mathbf{t} \Rightarrow \mathbf{N}A \vee B \equiv \mathbf{t} \Rightarrow \sim A \vee B$$

The above reductions show, in effect, that causal inference relations can be translated into full-fledged four-valued

logics. In fact, this representation is closely related to the notion of a *nested* logic program; in particular, the distinction between the global and ht-negation is closely related to the distinction between the negations used in two formalizations of nested logic programs suggested, respectively, in (Lloyd & Topor 1984) and (Lifschitz, Tang, & Turner 1999).

As yet another general consequence of the above representation, we obtain that causal relations constitute a logical counterpart of biconsequence relations (Bochman 1998a; 1998b) for the language of local classical connectives. Thus, given a causal relation $\Rightarrow$, we can construct the associated biconsequence relation directly as the following set of bisequents:

$$\{a : b \Vdash c : d \mid \bigwedge(\neg b \cup d) \Rightarrow \bigvee(\neg a \cup c)\}$$

The above set constitutes a biconsequence relation satisfying all the rules for the local classical connectives.

The nonmonotonic semantics

In addition to the monotonic semantics, described above, a causal relation determines also a natural nonmonotonic semantics.

Due to non-reflexivity of $\Rightarrow$, only some causally consistent worlds are also worlds in which any proposition is caused by some causal rule. These are the worlds α that are fixed points of $\mathcal{C}$, that is, $\alpha = \mathcal{C}(\alpha)$. In such 'exact' worlds any fact is causally explained.

Definition 4. The *nonmonotonic semantics* of a causal relation $\Rightarrow$ is the set of all its *exact worlds*, that is, worlds α such that $\alpha = \mathcal{C}(\alpha)$.

Propositions that hold in all exact worlds can be considered as the nonmonotonic consequences determined by the causal relation.

By a *causal theory* we will mean an arbitrary set of causal rules. A nonmonotonic semantics of a causal theory Δ will be identified with the nonmonotonic semantics of the least causal relation $\Rightarrow_\Delta$ that includes Δ. As has been shown in (Bochman 2003b), this semantics coincides with the semantics of causal theories defined in (McCain & Turner 1997).

Let us introduce the following definitions:

Definition 5. Causal theories Δ and Γ will be called

- (nonmonotonically) *equivalent* if they determine the same set of exact worlds;
- *strongly equivalent* if, for any set Φ of causal rules, $\Delta \cup \Phi$ is equivalent to $\Gamma \cup \Phi$;
- *causally equivalent* if $\Rightarrow_\Delta = \Rightarrow_\Gamma$.

Two theories are causally equivalent if and only if each theory can be obtained from the other using the inference postulates of causal relations. Strongly equivalent theories are 'equivalent forever', that is, they are interchangeable in any larger causal theory without changing the nonmonotonic semantics. Consequently, strong equivalence can be seen as an equivalence with respect to the monotonic logic of causal theories. And the next result from (Bochman 2003b) shows that this logic is precisely the logic of causal relations.

Theorem 4. *Two causal theories are strongly equivalent if and only if they are causally equivalent.*

The above result implies that causal relations are maximal inference relations that are adequate for nonmonotonic reasoning with causal theories.

Causal Interpretations of Program Rules

Now we are going to interpret logic programs in terms of causal rules. Speaking generally, the suggested interpretation is based on a recurrent idea that logic program rules express causal, or explanatory, relationship between propositional atoms (see, e.g., (Dix, Gottlob, & Marek 1994; Schlipf 1994)).

In what follows, for a set u of propositions, $\overline{u}$ will denote its complement, while $\mathbf{not}\ u$ the set $\{\mathbf{not}\ p \mid p \in u\}$.

A *logic program* Π is a set of program rules of the form

$$\mathbf{not}\ d, c \leftarrow a, \mathbf{not}\ b \qquad (*)$$

where a, b, c, d are finite sets of propositional atoms.

A pair (u, v) of sets of atoms will be called an *interpretation*; u and v are the sets of atoms that are, respectively, true and false in the interpretation (in a four-valued sense). An interpretation (u, v) will be called a *bimodel* (or a *partial model*) of a program Π, if it is closed with respect to the rules of Π, that is, for any rule (*) from Π, if $a \subseteq u$ and $b \subseteq v$, then either $c \cap u \neq \emptyset$, or $d \cap v \neq \emptyset$. A set u is a *model* of a program Π, if $(u, \overline{u})$ is a bimodel of Π.

Despite the diversity of existing semantics for logic programs, a general understanding of their meaning presupposes an asymmetric treatment of positive and negative information, which is reflected in viewing $\mathbf{not}$ as *negation as failure*. Such an understanding can be uniformly captured in causal logic by accepting the following additional postulate:

(Default Negation) $\neg p \Rightarrow \neg p$, for any atom p.

The above condition reflects a principle of *negation by default*, according to which negations of propositional atoms are self-explainable propositions (or abducibles). The effect of this postulate is that negated atoms are explainable whenever they can be consistently assumed to hold. In this sense, Default Negation reflects also a kind of *Closed Word Assumption* (CWA), namely, that $\neg p$ can be safely assumed to hold whenever p is not caused (explained) by other facts (cf. (Reiter 1980) for a similar description of CWA).

In what follows, we will denote by DN the set of causal rules $\neg p \Rightarrow \neg p$, for all atoms p of the underlying language.

The supported interpretation

Mainly for historical reasons, we will begin with the causal interpretation of logic programs based on the completion semantics. A semantic counterpart of Clark's completion, called *supported models*, has been introduced in (Apt, Blair, & Walker 1988) for normal logic programs. A generalization to disjunctive programs has been suggested in (Baral & Gelfond 1994), while an extension to programs with negation in heads has been defined in (Inoue & Sakama 1998). In what follows, we will use this latter definition.

Definition 6. A model u of a program Π will be called *supported*, if, for any $p \in u$, Π contains a rule (*) such that $a, d \subseteq u$, $b \subseteq \overline{u}$, and $c \cap u = \{p\}$.

It turns out that the corresponding causal interpretation of logic programs can be obtained by interpreting a program rule (*) as the following causal rule:

$$a, d, \neg b \Rightarrow \bigvee c$$

We will call this causal interpretation a *supported interpretation* of program rules. Let us denote by $su(\Pi)$ the set of causal rules corresponding by this interpretation to the rules of a program Π. As we already mentioned, however, a full meaning of a logic program presupposes also the principle of Default Negation. Accordingly, let us denote by $SU(\Pi)$ a causal theory which is the union of $su(\Pi)$ and the set DN of default negation rules. Then we obtain our first basic result:

Theorem 5. *The nonmonotonic semantics of $SU(\Pi)$ coincides with the set of supported models of Π.*

The above result shows that the supported causal interpretation of program rules provides an adequate description of logic programs under the supported (or completion) semantics. Moreover, we will show later that by choosing a maximal causal logic adequate for such a semantics, any causal theory is also representable by some logic program.

The stable interpretation

To begin with, supported and stable semantics of logic programs are based on a different understanding of program rules. Thus, the program rule $p \leftarrow p$ can change the set of supported models (so it has a non-trivial meaning for the latter), but it is trivial with respect to the stable semantics. This indicates that program rules should have a different causal interpretation under the stable semantics.

An interpretation that will be shown to be adequate for the stable semantics amounts to interpreting a program rule (*) as the following causal rule:

$$d, \neg b \Rightarrow \bigwedge a \rightarrow \bigvee c$$

This interpretation of program rules will be called a *stable causal interpretation*. The only difference with the supported interpretation, described earlier, amounts to treating positive premises in a not as assumptions (causal explanations), but as part of what is explained, or caused. It is important to observe, however, that the two interpretations coincide for program rules without positive atoms in bodies.

The above causal rule describes the net causal information embodied in the corresponding program rule. However, by the properties of a global and ht-negation, this rule is equivalent to each of the following causal rules

$$\mathbf{t} \Rightarrow \bigwedge (a \cup \mathbf{N}b) \rightarrow \bigvee (c \cup \mathbf{N}d)$$

$$\mathbf{t} \Rightarrow \bigwedge (a \cup {\sim}b) \rightarrow \bigvee (c \cup {\sim}d)$$

The latter rules give a fully transparent and modular representation of program rules. Note, in particular, that, apart from the overall causal framework, the non-classicality of program rules is reduced in this representation to the interpretation of **not**. On the other hand, the source stable causal interpretation gives a classical understanding to **not**, but makes the resulting literals explanatory assumptions. As a result, the non-classicality of this representation amounts solely to the non-classicality of $\Rightarrow$.

We will denote by $st(\Pi)$ the set of causal rules obtained by the stable interpretation of a program Π, while $ST(\Pi)$ will denote $st(\Pi) \cup DN$. Then we have

Theorem 6. *The nonmonotonic semantics of $ST(\Pi)$ coincides with the stable semantics of Π.*

The above result[1] shows that the stable causal interpretation of program rules provides an adequate description of logic programs under the stable semantics. Moreover, later it will be shown that by choosing an adequate causal logic for such a semantics (which will be different from the supported case), any causal theory is also representable by some logic program.

A connection between the stable and supported interpretation can be established using the *inverse shift* construction from (Inoue & Sakama 1998) that transforms a logic program Π into the following program $invshift(\Pi)$:

$$\{\mathbf{not}\, a, \mathbf{not}\, d, c \leftarrow \mathbf{not}\, b \mid \mathbf{not}\, d, c \leftarrow a, \mathbf{not}\, b \in \Pi\}$$

As can be seen, $invshift(\Pi)$ contains only program rules without positive premises in bodies. Moreover, it is easy to see that the supported interpretation of Π coincides with the stable interpretation of $invshift(\Pi)$, that is,

$$su(\Pi) = su(invshift(\Pi)) = st(invshift(\Pi))$$

As a result, we immediately obtain the following

Corollary 7. *(Inoue & Sakama 1998) Supported semantics of a program Π coincides with the stable semantics of $invshift(\Pi)$.*

Causal Logics of Logic Programs

In this section we will describe two particular kinds of causal inference relations that will be shown to be adequate (and maximal) for reasoning with logic programs under the two semantic interpretations described above. For logical reasons, we will begin this time with the stable interpretation.

Negatively closed causal relations

A causal relation will be called *negatively closed*, if it satisfies the Default Negation postulate. As we already mentioned, this postulate reflects the main distinctive feature of logical reasoning behind logic programs.

It is important to observe that Default Negation is not a structural postulate, since propositional atoms p in it cannot be replaced by arbitrary formulas. Still, it can be shown to be equivalent to each of the following rules, for any atom p and arbitrary propositions A and B:

Positive Deduction If $A \wedge p \Rightarrow B$, then $A \Rightarrow p \rightarrow B$;

Negative Import If $A \Rightarrow B \vee p$, then $A \wedge \neg p \Rightarrow B$.

[1] For disjunctive programs, this result has actually been proved in (Giunchiglia *et al.* 2003).

For a world α, we will denote by $At(\alpha)$ the set of its propositional atoms. The next definition provides a semantics for negatively closed causal relations.

Definition 7. A possible worlds semantics $\mathbb{W} = (W, R, V)$ will be called *inclusive*, if, for any worlds $\alpha, \beta \in W$, $\alpha R \beta$ holds only if $At(\beta) \subseteq At(\alpha)$.

Then the following result can be obtained as a consequence of the general completeness theorem.

Corollary 8. *A causal relation is negatively closed iff it has an inclusive quasi-reflexive possible worlds semantics.*

The next proposition establishes, in effect, a connection between the nonmonotonic semantics of negatively closed causal relations and stable semantics of logic programs.

Lemma 9. *A world α is an exact world of a negatively closed causal relation if and only if $\mathcal{C}(\alpha) \subseteq \alpha$ and, for any world β such that $\mathcal{C}(\alpha) \subseteq \beta$, if $At(\beta) \subseteq At(\alpha)$, then $At(\beta) = At(\alpha)$.*

A world α is an exact world of a negatively closed causal relation if it is causally consistent (that is, closed with respect to the rules of $\Rightarrow$) and has a minimal number of positive atoms among the worlds consistent with $\mathcal{C}(\alpha)$.

Now we are going to show that negatively closed causal relations constitute a precise causal logic behind stable logic programming.

To begin with, let $\Rightarrow^c_\Delta$ denote a least negatively closed causal relation containing a causal theory Δ. Then by Theorem 6 we immediately obtain

Corollary 10. *The stable semantics of a program Π coincides with the nonmonotonic semantics of $\Rightarrow^c_{st(\Pi)}$.*

Thus, negatively closed causal relations can be seen as a causal logic of logic programs under the stable semantics. Moreover, we are going to show that this is actually a maximal such logic.

By (Lifschitz, Pearce, & Valverde 2001), programs Π_1 and Π_2 are *strongly equivalent*, if, for any program Π, $\Pi_1 \cup \Pi$ has the same stable models as $\Pi_2 \cup \Pi$. Let us say also that causal theories Δ and Γ are *stably equivalent* if they determine the same negatively closed causal relation, that is $\Rightarrow^c_\Delta = \Rightarrow^c_\Gamma$. In other words, each causal theory can be obtained from the other using the postulates of causal relations and Default Negation. Then we have

Theorem 11. *Programs Π_1 and Π_2 are strongly equivalent if and only if $st(\Pi_1)$ and $st(\Pi_2)$ are stably equivalent.*

The above result implies that negatively closed causal relations are maximal causal relations that are adequate for the stable semantics of the associated logic programs. This result constitutes a causal counterpart of the corresponding results about strong equivalence of logic programs (see (Lifschitz, Pearce, & Valverde 2001; Turner 2001)).

Note now that any causal rule is logically reducible to a set of 'clausal' causal rules $\bigwedge a \Rightarrow \bigvee b$, where a and b are sets of classical literals. Moreover, under the stable interpretation of logic programs, each such causal rule can be identified with some program rule. Accordingly, for a causal theory Δ, we will denote by $pr(\Delta)$ the set of program rules obtained in this way from Δ. Then the following result can be shown:

Theorem 12. *For any causal theory Δ, the nonmonotonic semantics of $\Rightarrow^c_\Delta$ coincides with the stable semantics of the program $pr(\Delta)$.*

The importance of the above result lies in the fact that, under the stable causal interpretation, any causal theory is reducible, in turn, to a logic program. Thus, for negatively closed causal relations, the correspondence between logic programs and causal theories turns out to be bidirectional.

Tight causal relations

A causal rule of the form $A \Rightarrow \bigvee a$, where a is a set of atoms, will be called *head-positive*. The important role of such rules for our study stems already from the fact that the supported interpretation of logic program rules produces only head-positive causal rules. It turns out that the reasoning with such rules admits a stronger causal logic described in the next definition.

Definition 8. A causal relation is *tight*, if it satisfies

(Tightness) $l \wedge m \Rightarrow l \vee m$, for distinct literals l and m,

and *positively tight* if it is tight and negatively closed.

Tight causal relations are 'highly explanatory', since they automatically explain disjunctions of any two distinct literals that hold in a world. In practice, this means that only single literals have to be explained in such causal systems. The Tightness postulate is also not structural, since it does not hold for arbitrary propositions. For negatively closed causal relations, it is reducible to its positive instance, namely to

(Positive Tightness) $p \wedge q \Rightarrow p \vee q$, for distinct atoms p, q.

A semantic description of tight causal relations is given below, where $\triangle$ denotes a symmetric set difference.

Definition 9. A possible world semantics will be called *tight*, if, for any worlds $\alpha, \beta \in W$, $\alpha R \beta$ holds only if $At(\beta) \triangle At(\alpha)$ contains at most one propositional atom.

Then we have

Corollary 13. *A causal relation is tight if and only if it has a tight quasi-reflexive possible worlds semantics.*

The next proposition describes the nonmonotonic semantics of tight causal relations.

Lemma 14. *A world α is an exact world of a tight causal relation $\Rightarrow$ if and only if $\mathcal{C}(\alpha) \subseteq \alpha$, and no world β that differs from α by one atom is such that $\mathcal{C}(\alpha) \subseteq \beta$.*

A world α is an exact world of a tight causal relation if it is causally consistent, and there are no worlds consistent with $\mathcal{C}(\alpha)$ that differ from α by a single atom[2].

For a causal theory Δ, we denote by $\Rightarrow^t_\Delta$ the least tight causal relation containing Δ, while $\Rightarrow^{ct}_\Delta$ will denote the least such relation that is positively tight.

The relevance of tight causal relations for head-positive causal theories is revealed by the following

Theorem 15. *If Δ is a head-positive causal theory, then the nonmonotonic semantics of Δ coincides with the nonmonotonic semantics of $\Rightarrow^t_\Delta$.*

[2]Notice a similarity with models of pointwise circumscription (Lifschitz 1987).

Thus, Tightness preserves the nonmonotonic semantics of head-positive causal theories, and hence tight causal relations constitute an adequate causal logic for the latter. As a consequence of this result, we obtain that positively tight causal relations constitute an adequate causal logic for logic programs under the supported semantics.

Corollary 16. *The supported semantics of a program* Π *coincides with the nonmonotonic semantics of* $\Rightarrow^{ct}_{su(\Pi)}$.

Moreover, positively tight causal relations constitute a maximal causal logic for the supported semantics.

Definition 10. Programs Π_1 and Π_2 will be called *strongly sup-equivalent* if, for any program Π, $\Pi_1 \cup \Pi$ has the same supported semantics as $\Pi_2 \cup \Pi$.

Let us say also that two causal theories are *tightly equivalent*, if they determine the same positively tight causal relation (that is, each can be obtained from the other using the postulates of causal relations, Default Negation and Tightness). Then we have

Theorem 17. *Programs* Π_1 *and* Π_2 *are strongly sup-equivalent iff* $su(\Pi_1)$ *and* $su(\Pi_2)$ *are tightly equivalent.*

The above result implies that positively tight causal relations are maximal causal relations that are adequate for the supported semantics of the associated logic programs. Furthermore, we are going to show now that any causal theory is tightly equivalent, in a sense, to some logic program. This fact is based on the following reduction property of such causal relations.

Lemma 18. *Let* $\Rightarrow$ *be a positively tight causal relation. Then, for any proposition* A *and any sets of propositional atoms* a, c *such that* $a \cap c = \emptyset$ *and* $c \neq \emptyset$*, the causal rule* $A \Rightarrow \bigvee(\neg a \cup c)$ *is equivalent to the set of rules*

$$\{A, a, \neg(c \backslash \{p\}) \Rightarrow p \mid p \in c\}$$

This result implies, in effect, that any causal theory is tightly equivalent to some *definite* head-positive causal theory. Moreover, since any causal rule is reducible to a set of corresponding clausal rules, we obtain that any causal theory is actually reducible to a set of rules of the form $a, \neg b \Rightarrow \tilde{p}$, where a and b are sets of atoms, while $\tilde{p}$ is either an atom p, or the falsity constant $\mathbf{f}$.

Notice now that, under the supported interpretation of program rules, any causal rule $a, \neg b \Rightarrow p$ corresponds to a unique *normal* program rule $p \leftarrow a, \mathbf{not}\, b$, while a causal rule $a, \neg b \Rightarrow \mathbf{f}$ corresponds to a constraint $\leftarrow a, \mathbf{not}\, b$. Let us denote by $npr(\Delta)$ the logic program obtained by this procedure from an arbitrary causal theory Δ. Then we obtain

Theorem 19. *Any causal theory* Δ *is tightly equivalent to* $su(npr(\Delta))$.

As a 'by-product' of our construction, we obtain that any program under the supported semantics is reducible to a normal program with constraints:

Corollary 20. *Any logic program is strongly sup-equivalent to a normal program with constraints.*

Actually, Lemma 18 implies that the corresponding modular transformation of logic programs can be obtained by replacing any program rule $\mathbf{not}\, d, c \leftarrow a, \mathbf{not}\, b$ such that $c \neq \emptyset$ with the set

$$\{p \leftarrow a, d, \mathbf{not}\, b, \mathbf{not}(c \backslash \{p\}) \mid p \in c\}$$

while any 'extended' constraint $\mathbf{not}\, d \leftarrow a, \mathbf{not}\, b$ is reduced to a 'normal' constraint $\leftarrow a, d, \mathbf{not}\, b$.

Partial Causation

As in other logical areas, three- and four-valued logics have been used in logic programming in order to capture 'partial' generalizations of existing semantics. A suitable generalization of causal relations, introduced below, will be shown to provide a logical representation for such semantics.

At first glance, the idea of a partial generalization of causal relations does not look promising since causal relations are already four-valued in some sense. Fortunately, an elegant way out has been suggested in relevant logics (see (Routley *et al.* 1982; Fagin, Halpern, & Vardi 1995)).

Definition 11. • An *invariant possible worlds model* is a quadruple $\mathbb{W} = (W, R, ^*, V)$, where$(W, R, V)$ is a possible worlds model, while * is a function on W such that, for any $\alpha \in W$, $\alpha^{**} = \alpha$.

- An invariant model will be called *regular*, if R is quasi-reflexive, and for any $\alpha, \beta \in W$, if $\alpha R \beta$, then $\alpha^* R \beta^*$.

By the intended understanding, pairs of worlds (α, α^*) determine four-valued interpretations. Accordingly, the global negation $\sim$ can now be interpreted as follows:

$$\alpha \models \sim A \quad \text{iff} \quad \alpha^* \not\models A$$

This definition coincides with the definition of the relevant negation in (Routley *et al.* 1982). As a result, we obtain an extension of the language of causal logic with a negation connective $\sim$ that has a new semantic interpretation. An axiomatization of this causal logic can be obtained by adding the following postulates for the new connective:

(de Morgan) $\mathbf{t} \Rightarrow \sim(A \wedge B) \leftrightarrow (\sim A \vee \sim B)$;

(Commutativity) $\mathbf{t} \Rightarrow \sim\neg A \leftrightarrow \neg\sim A$;

(Invariance) If $\neg A \Rightarrow \neg B$, then $\sim A \Rightarrow \sim B$.

Due to general properties of causal relations, a rule $\mathbf{t} \Rightarrow A \leftrightarrow B$ implies that A and B can replace each other both in heads and bodies of causal rules. In addition, the Invariance rule implies, ultimately, that $\neg A \Rightarrow \neg B$ is equivalent to $\sim A \Rightarrow \sim B$.

In what follows, we will use a derived unary connective $*A$ defined as $\neg\sim A$. This connective corresponds to the *conflation* connective in the bilattice theory (Fitting 1991).

For partial causal relations, we accept the same definition of the nonmonotonic semantics, namely the set of all worlds (i.e., maximal consistent sets of propositions in the language $\{\wedge, \neg, \sim\}$) that are fixed points of $\mathcal{C}$. It is important to keep in mind, however, that this notion of a world is a syntactic counterpart of a four-valued interpretation; it is uniquely determined by the literals of the form $p, \neg p, \sim p, \neg\sim p$.

The partial supported interpretation

A three-valued extension of Clark completion for normal logic programs has been suggested in (Fitting 1985) as a way of resolving some of the problems with the latter; in later papers, however, Fitting has extended this construction to a general four-valued setting. The semantic counterpart of this generalized completion has been defined as fixed-points of an 'immediate consequence' operator Φ on the set of four-valued interpretations. We will call such interpretations *partial supported models*. Below we will present a causal interpretation of this semantics (generalized to arbitrary logic programs).

A *partial supported interpretation* of logic programs is obtained by interpreting each program rule (*) as the following causal rule:

$$a, *d, \sim b \Rightarrow \bigvee c$$

By Invariance, the above rule is equivalent to

$$*a, d, \neg b \Rightarrow \bigvee *c$$

As a matter of fact, the above two causal rules correspond precisely to the *split*, or *doubling*, of program rules used in (Drabent & Martelli 1991; Wallace 1993; Schlipf 1994) for representing 3-completion.

Since the three-valued completion has been defined in the literature only for normal logic programs, let $PSU(\Pi)$ denote the causal theory containing the above translation of all the rules of a *normal* program Π, plus the set DN of default negation rules. Then we obtain

Theorem 21. *The nonmonotonic semantics of $PSU(\Pi)$ coincides with the set of partial supported models of Π.*

The above result implies that negatively closed partial causal relations constitute an adequate causal logic for logic programs under the four-valued supported semantics.

The partial stable interpretation

Just as in the main case, the distinction between the partial supported and partial stable interpretations amounts to treating positive premises in bodies of program rules not as causal assumptions, but as conditions on conclusions. More exactly, a *partial stable interpretation* interprets a program rule (*) as a causal rule

$$*d, \sim b \Rightarrow \bigwedge a \rightarrow \bigvee c$$

The above rule clearly shows that partial stable interpretation is just a four-valued version of a stable interpretation, and consequently the p-stable semantics below will be just a four-valued version of the stable semantics. Again, due to Invariance, the above causal rule is equivalent to

$$d, \neg b \Rightarrow \bigwedge *a \rightarrow \bigvee *c$$

This immediately suggests that the partial stable interpretation can be seen as a stable interpretation of a *doubled* logic program (cf. (Wallace 1993)). An abstract, argumentation-theoretic formulation of this construction can be found in (Bochman 2003a).

$PST(\Pi)$ will denote the causal theory containing the above translations of all the program rules from Π, plus DN. Then our main result will show that the nonmonotonic semantics of this causal theory will determine precisely the set of *p-stable models* of the source logic program. P-stable models have been introduced in (Bochman 1998b) as a slight modification of Przymusinski's partial stable models for disjunctive programs (Przymusinski 1991); the reason for the modification was that Przymusinski's semantics violated the principle of partial deduction (alias GPPE). The modification has not changed, however, the correspondence with partial stable models of normal logic programs.

Theorem 22. *The nonmonotonic semantics of $PST(\Pi)$ coincides with the set of p-stable models of Π.*

It turns out that the original partial stable semantics of Przymusinski can also be obtained in this framework, but it requires a further generalization of partial causation (namely dropping the Invariance postulate). Due to lack of space, however, we will leave it for another occasion.

On declarative meaning and informational content of logic programs

By the causal interpretation suggested in this study, the declarative meaning of a logic program can be defined as the logical meaning of the corresponding causal theory. Thus, the declarative meaning of a logic program Π under the stable semantics should be identified with that of $st(\Pi)$, while under the supported semantics it should be identified with that of $su(\Pi)$. This primary description can be refined, however, by choosing also the causal logic appropriate for each interpretation. If we choose, for example, a maximal causal logic in each case, then the declarative meaning of Π under the stable semantics could be taken to be the least negatively closed causal relation containing $st(\Pi)$, while for the supported semantics it would be the least positively tight causal relation containing $su(\Pi)$. As a minimal reasonable choice, however, we can choose in both cases only the least causal relation containing the corresponding causal theory. At this point, we still have no sufficient grounds for the best choice, and hence we have to look for applications of this notion in resolving practical problems.

A most immediate application of the above notion of declarative meaning consists in using it for determining the *informational content* of a logic program.

In many reasoning tasks associated with logic programs, it is important to have some notion of their informational content. Given such a notion, the programs can be naturally compared and systematically modified. Such a need is especially pressing in the theory of *program updates*, since an update is normally required to produce a minimal change of the informational content.

Intuitively, an informational content of a logic program should contain all the information needed to determine the properties and behavior of the program. Consequently, it is natural to suppose that, if two programs have the same informational content, they should be interchangeable in any larger program without changing its properties. Moreover, it seems natural to suppose also that adding new rules to

a program should normally increase its informational content (unless the added rules are derivable somehow from the source program). All these properties will hold if we would identify the informational content with the declarative meaning of a program, as defined above.

The information that is nonmonotonically derivable from the program is, of course, part of its informational content. However, as we already argued in the introduction, the nonmonotonic semantics is patently insufficient for capturing the full content of the program. Indeed, it is nonmonotonic precisely in the sense that adding new rules to a program may result in removal of some of the previously derivable information. Nevertheless, for a more fine-grained analysis it it seems desirable to make the nonmonotonic part of the informational content explicit.

It is a good moment to recall that causal rules serve, in general, simultaneously two informational roles. One consists in determining causally consistent worlds, while the other in establishing explanatory relations between propositions. Fortunately, in causal logic these two roles can be neatly separated by decomposing a causal rule into a constraint and an explanation (see Lemma 1). This suggests the following definition.

Definition 12. • The set of constraints $A \Rightarrow \mathbf{f}$ belonging to a causal relation will be called its *factual content*.

- The *explanatory content* of a causal relation is the set of its *explanatory rules*, namely the rules $A \Rightarrow B$ such that $A \models B$.

Constraints restrict the set of worlds that are admissible (causally consistent) with respect to a causal theory. In this sense they play the role of ordinary classical formulas, namely they just express facts about the world. However, they does not explain anything, and hence they can be seen as devoid of explanatory content. The later is expressed, however, by explanatory causal rules. Such rules are 'factually trivial', since they do not impose restrictions on admissible worlds; their only role consists in determining what explains what in admissible worlds. Consequently, the factual and explanatory contents are not only disjoint, but are actually independent of each other. Moreover, the informational content of causal theories and logic programs can be safely represented as a union of a factual and explanatory contents.

The interplay of the factual and explanatory content determines, eventually, the nonmonotonic semantics, and it is responsible, in particular, for the nonmonotonic properties of the latter. Such properties arise from the fact that the two kinds of content have opposite impacts on derivability. Thus, addition of constraints leads, as expected, to reduction of the set of admissible worlds (and hence to increase of factual information). However, the addition of explanatory rules leads, in general, to *increase* of exact worlds, and hence to decrease in nonmonotonically derived information.

Let us say that two causal relations are *factually equivalent*, if they have the same constraints. Then we have

Lemma 23. *If $\Rightarrow_1$ and $\Rightarrow_2$ are factually equivalent causal relations such that $\Rightarrow_1 \subseteq \Rightarrow_2$, then any exact world of $\Rightarrow_1$ is an exact world of $\Rightarrow_2$.*

The above result shows, in effect, that if we add causal rules that do not change the factual content, then we increase the set of explained worlds. But in all cases, the net effect of growth in informational content leads to a monotonic *reduction of the set of non-exact (unexplained) worlds*.

As a final observation, note that the additional postulates of Default Negation and Tightness, used earlier for a more precise description of the causal logics appropriate for particular logic programming semantics, produce only additional explanatory rules. In other words, they introduced only new ways of explaining facts that do not change the factual content of the program.

The above considerations show that the study of informational content of logic programs and causal theories is a very interesting topic that is still far from being completed.

Conclusions

In this section we will briefly discuss some of the questions, problems and opportunities that arise in connection with the suggested causal interpretation of logic programs.

One of the main theoretical benefits of the causal interpretation of logic programs is that it allows to analyze the properties of such programs directly in the logical framework of causal relations. For instance, since positively tight causal relations are stronger then negatively closed ones, the general properties of causal logic immediately imply that any stable model is also a supported model, while the reverse inclusion holds only under some acyclicity conditions (a lá Fages' theorem (Fages 1994)). Actually, as far as we can see, the main results about logic programs can be stated more clearly and transparently in the logical formalism of causal relations. In addition, the causal interpretation suggests a more systematic view of the declarative meaning and informational content of logic programs, and hence can provide, for instance, a firm theoretical basis for the theory of program updates. But what is even more important, the formalism of causal inference has already shown its potential in representing problems in various domains of AI. This suggests that the causal interpretation of logic programs can serve as a basis for a knowledge representation methodology that allows a systematic translation of raw empirical information into logic programs.

The suggested causal interpretation of logic programs requires, however, a reconsideration of the relationship between logic programs and nonmonotonic formalisms such as default and autoepistemic logic. A distinctive feature of the latter is that they are inherently epistemic, or *intensional* formalisms. Namely, they are essentially based on such notions as belief and knowledge, unlike the extensional classical logic that gives a direct representation of facts about the world. In this respect, the fact that the correspondence between logic programs and these nonmonotonic logics is unidirectional (namely, an embedding) is not accidental; it shows, in effect, that logic programming constitutes a more specific formalism, and hence a formalism with a stronger underlying logic. For example, while the causal logic allows for reasoning by cases, the invalidity of such a reasoning in default logic has long been considered a shortcoming of the latter, witness numerous (unsuccessful) attempts to improve

it on this score. But what is more important, the causal logic can also be seen as an extensional formalism that provides a direct description of factual and causal (explanatory) information about the world. In this respect, causal logics constitute an important conceptual shift in the development of nonmonotonic reasoning, since they form a most immediate generalization of classical logic that allow for such a reasoning.

The above considerations require us also to take a second look on the role and use of the so-called second, classical negation in logic programming. As was rightly mentioned in (Gelfond & Lifschitz 1991), the formalism of extended logic programs meets the default logic halfway. And indeed, most of the examples used in (Gelfond & Lifschitz 1991) for justifying the need for the second negation (such as legal or administrative regulations) have a distinct epistemic flavor, including extreme cases like

$$Interview(x) \leftarrow \mathbf{not}\, Eligible(x), \mathbf{not}\, \neg Eligible(x),$$

or inference rules for actions like $Cross \leftarrow \mathbf{not}\, Train$. All such examples presuppose an epistemic reading of $\mathbf{not}\, p$ as "*p is not known to hold*". This reading is distinct from an objective interpretation of $\mathbf{not}$ as a classical negation in causal assumptions. It seems that these examples actually delineate a proper applicability area for such program rules, namely an epistemic (modal) reasoning, or, more exactly, a representation of such a reasoning in logic programs. Moreover, as has been argued in (Gelfond 1994), a comprehensive representation of such a reasoning in logic programming requires a further extension of the language with proper epistemic operators as well. On the other hand, this delineation frees room for an objective use of $\mathbf{not}$ as a classical negation in causal assumptions. The relationship between the two possible uses of default negation, however, is far from being trivial, and requires further study.

References

Apt, K. R., and Bol, R. N. 1994. Logic programming and negation: A survey. *Journal of Logic Programming* 19,20:9–71.

Apt, K.; Blair, H.; and Walker, A. 1988. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. San Mateo, CA: Morgan Kaufmann. 89–148.

Baral, C., and Gelfond, M. 1994. Logic programming and knowledge representation. *Journal of Logic Programming* 19,20:73–148.

Belnap, Jr, N. D. 1977. A useful four-valued logic. In Dunn, M., and Epstein, G., eds., *Modern Uses of Multiple-Valued Logic*. D. Reidel. 8–41.

Bochman, A. 1998a. Biconsequence relations: A four-valued formalism of reasoning with inconsistency and incompleteness. *Notre Dame Journal of Formal Logic* 39(1):47–73.

Bochman, A. 1998b. A logical foundation for logic programming, I and II. *Journal of Logic Programming* 35:151–194.

Bochman, A. 2003a. Collective argumentation and disjunctive logic programming. *Journal of Logic and Computation* 13:405–428.

Bochman, A. 2003b. A logic for causal reasoning. In Gottlob, G., and Walsh, T., eds., *Proceedings Int. Joint Conference on Artificial Intelligence, IJCAI'03*, 141–146. Acapulco: Morgan Kaufmann.

Bochman, A. 2004. Production inference, nonmonotonicity, and abduction. In *Proc. Eight International Symp. On Artificial Intelligence and Mathematics*.

Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*. Plenum Press. 293–322.

Dix, J.; Gottlob, G.; and Marek, V. 1994. Causal models for disjunctive logic programs. In Hentenryck, P., ed., *Proc. 11th Int. Conf. On Logic Programming, S. Margherita Ligure*, 290–302. MIT Press.

Drabent, W., and Martelli, M. 1991. Strict completion of logic programs. *New Generation Computing* 9:69–79.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science* 1:51–60.

Fagin, R.; Halpern, J. Y.; and Vardi, M. Y. 1995. A nonstandard approach to the logical omniscience problem. *Artificial Intelligence* 79:203–240.

Fitting, M. C. 1985. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming* 2:295–312.

Fitting, M. C. 1991. Bilattices and the semantics of logic programming. *J. of Logic Programming* 11:91–116.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Proc. 5th International Conf./Symp. on Logic Programming*, 1070–1080. Cambridge, MA: MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation and disjunctive databases. *New Generation Computing* 9:365–385.

Gelfond, M. 1994. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence* 12:89–116.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2003. Nonmonotonic causal theories. *Artificial Intelligence* (to appear).

Inoue, K., and Sakama, C. 1998. Negation as failure in the head. *Journal of Logic Programming* 35:39–78.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2:526–541.

Lifschitz, V.; Tang, L. R.; and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence* 25:369–389.

Lifschitz, V. 1987. Pointwise circumscription. In Ginsberg, M., ed., *Readings in Non-Monotonic Reasoning*. San Mateo, CA: Morgan Kaufmann. 179–193.

Lloyd, J., and Topor, R. 1984. Making Prolog more expressive. *Journal of Logic Programming* 3:225–24–.

McCain, N., and Turner, H. 1997. Causal theories of action and change. In *Proceedings AAAI-97*, 460–465.

Przymusinski, T. C. 1991. Stable semantics for disjunctive programs. *New Generation Computing* 9:401–424.

Reiter, R. 1978. On closed world data bases. In Gallaire, H., and Minker, J., eds., *Logic and Data Bases*. Plenum Press. 119–140.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

Routley, R.; Plumwood, V.; Meyer, R.; and Brady, R. 1982. *Relevant Logics and their Rivals*. Ridgeview.

Schlipf, J. S. 1994. A comparison of notions of negation as failure. In Levi, G., ed., *Advances in Logic Programming Theory*. Oxford: Clarendon Press. 1–53.

Shepherdson, J. C. 1988. Negation in logic programming. In Minker, J., ed., *Deductive Databases and Logic Programming*. M. Kaufmann. 19–88.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In Eiter, T.; Faber, W.; and Truszczyński, M., eds., *Proceedings LPNMR'01*, 81–92. Vienna: Springer. LNAI 2173.

Wallace, M. 1993. Tight, consistent, and computable completions for unrestricted logic programs. *J. of Logic Programming* 15:243–273.