

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 [Boc03b] is shown to provide a natural logical basis for logic programming. More exactly, any logic program can be seen as a causal theory satisfying the Negation As Default principle (alias CWA). Moreover, unlike known translations to other nonmonotonic formalisms, the correspondence between logic programs and causal theories is bidirectional - 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, and covers a broad range of 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.

1 Introduction

From its very beginning, logic programming was based on the idea that program rules must have both a procedural and declarative (logical) meaning, the latter being the meaning of the corresponding classical logical formula. This understanding was challenged, however, with the introduction of negation as failure. Clark's completion [Cla78] and Reiter's Closed World Assumption [Rei78] have been commonly accepted as more adequate interpretations of logic programs. A development of these ideas have led to three-valued completion [Fit85] and supported models of logic programs [ABW88]. 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 (cf. [She88]).

Things have changed, however, with the discovery that logic programs with negation as failure allow to represent nonmonotonic reasoning. Moreover, nonmonotonic logics inspired a new kind of semantics for logic programs, the stable and answer set semantics [GL88, GL91]. This advanced logic programming to the role of a general computational mechanism for knowledge representation and reasoning (see, e.g., [AB94, BG94]). In addition, Przymusinski has developed a comprehensive semantics of partial stable models, with the stable and well-founded semantics as special cases.

The idea of a dual interpretation of logic programs persisted in all these developments. However, the original question “*What is a logical 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 the latter has been commonly seen as settled by assigning logic programs some nonmonotonic semantics. Such a solution, however, turns out to be problematic. Thus, the text of a program does not reflect directly such a declarative meaning, and adding new rules to a program does not necessarily mean that the new program contains more information. Worse still, quite diverse programs can have the same ‘meaning’, e.g., $\{p \leftarrow q\}$ and $\{q \leftarrow p\}$ (both have the empty least model). As a result, a meaning of this kind turns out to be useless for most purposes we could possibly have in invoking this notion (including knowledge representation).

A general way of resolving the above problem that we are suggesting 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) can be identified with its logical meaning determined by the associated *monotonic* logic and its semantics. Moreover, the main claim (and hence the main subject) of this study is that the causal logic introduced in [Boc03b] allows to provide precisely such a logical basis for logic programming.

The plan of the paper is as follows. We briefly describe first the causal logic that possesses both a monotonic and nonmonotonic semantics. Then we give two causal interpretations of program rules that will be shown to be adequate, respectively, for the supported and stable semantics of logic programs. After that we introduce two special causal logics that constitute maximal logics for each of these interpretations. Finally, we will consider a partial generalization of causal relations that will be shown to provide an

interpretation for three- and four-valued completion, as well as for partial stable semantics of logic programs.

2 Causal production relations

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

A *causal rule* is a rule of the form $A \Rightarrow B$, where A and B are classical propositions. Such a rule says that, when A holds, B is caused (or explained).

Definition 2.1. A *causal production relation* (or simply a *causal 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 can be seen as one of the main advantages of causal reasoning as compared with, say, default logic. It 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, and distinct from modal (intensional) formalisms that deal primarily with beliefs and knowledge.

We extend causal rules to rules having arbitrary sets of propositions as premises: for a set u , $u \Rightarrow A$ will be taken to hold, if $\bigwedge a \Rightarrow A$, for some finite $a \subseteq u$. $\mathcal{C}(u)$ 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.

2.1 A possible worlds semantics

A semantic interpretation of causal relations can be given in terms of possible worlds (or Kripke) models (W, R, V) , where W is a set of possible worlds, R an accessibility relation on W , and V a function assigning each world a propositional interpretation. Intuitively, $\alpha R \beta$ means that α is an initial state, and β a possible output state of a causal production process. A Kripke model is *quasi-reflexive* if it satisfies the condition that if $\alpha R \beta$, then $\alpha R \alpha$.

Definition 2.2. A rule $A \Rightarrow B$ is *valid* in a Kripke 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 is defined as follows: $\alpha R \beta \equiv \mathcal{C}(\alpha) \subseteq \alpha \cap \beta$. An adequacy of the canonical semantics immediately implies the completeness result:

Theorem 2.1. *A relation on propositions is causal if and only if it is determined by a quasi-reflexive Kripke model.*

2.2 The nonmonotonic semantics

In addition to the monotonic semantics, a causal relation determines also an important nonmonotonic semantics. Due to non-reflexivity of \Rightarrow , only some causally closed 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 2.3. The *nonmonotonic semantics* of a causal relation \Rightarrow is the set of all its *exact worlds*, that is, worlds α such that $\alpha = \mathcal{C}(\alpha)$.

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_{Δ} that includes Δ . As has been shown in [Boc03b], this semantics coincides with the semantics of causal theories from [MT97]. Moreover, causal production relations constitute a maximal causal logic that agrees with this nonmonotonic semantics.

3 Causal Interpretations of Program Rules

Now we turn to a causal interpretation of logic programs. The interpretation makes precise a recurrent idea that logic programs express causal, or explanatory, relationship between propositions (see, e.g., [DGM94, Sch94]).

In what follows, for a set u of propositions, \bar{u} will denote its complement, while **not** u will denote the set $\{\mathbf{not} p \mid p \in u\}$ (and similarly for $\neg u$).

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

$$\mathbf{not} d, c \leftarrow a, \mathbf{not} b \tag{*}$$

where a, b, c, d are finite sets of propositional atoms. An *interpretation* is a pair (u, v) of sets of atoms; u and v are the atoms that are, respectively, true and false in the interpretation (in a four-valued sense). An interpretation (u, v) is a *bimodel* of a program Π , if it is closed with respect to the rules of Π : 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, \bar{u}) is a bimodel of Π .

Logic programs presuppose a particular treatment of negative information, which is reflected in viewing **not** as negation as failure. Such an understanding will be captured in causal logic by the following additional postulate:

(Default Negation) $\neg p \Rightarrow \mathbf{not} p$, for any propositional atom p .

This condition reflects a principle of *Negation as Default*, according to which negations of propositional atoms are self-explainable whenever they can be consistently assumed to hold. In this sense, it reflects also a kind of *Closed World Assumption* (CWA), namely, that $\mathbf{not} p$ can be assumed to hold whenever p is not explained by other facts (cf. [GL91] for a similar description of CWA). In what follows, DN will denote the set of causal rules $\neg p \Rightarrow \mathbf{not} p$, for all propositional atoms p of the underlying language.

3.1 The supported interpretation

Mainly for historical reasons, we 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 [ABW88] for normal logic programs. A generalization to disjunctive programs has been suggested in [BG94], while an extension to programs with negation in heads has been defined in [IS98]. Below we will use this latter definition.

Definition 3.1. A model u of a program Π will be called *supported*, if, for any $p \in u$, Π contains a rule (*) such that $a, d \subseteq u$, $b \subseteq \bar{u}$, and $c \cap u = \{p\}$.

The corresponding *supported causal interpretation* of logic programs is obtained by interpreting a program rule (*) as the following causal rule:

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

Let $su(\Pi)$ denote the causal theory corresponding to a program Π by this interpretation. The full meaning of a logic program presupposes, however, also the principle of Default Negation. So, let $SU(\Pi)$ denote the union of $su(\Pi)$ and the set DN of default negation rules. Then the following result shows that the supported interpretation is adequate for logic programs under the supported (or completion) semantics.

Theorem 3.1. *The nonmonotonic semantics of $SU(\Pi)$ coincides with the set of supported models of Π .*

3.2 The stable interpretation

The stable semantics of logic programs is based on a different understanding of program rules. Thus, it is well-known that the program rule $p \leftarrow p$ can change the set of supported models, but it is trivial with respect to the stable semantics. The corresponding *stable causal interpretation* amounts to interpreting a program rule (*) as the following causal rule:

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

The difference with the supported interpretation amounts to treating positive premises in a not as causal assumptions, but as part of what is explained.

Let $st(\Pi)$ be the causal theory corresponding to a program Π by this interpretation, while $ST(\Pi)$ the union $st(\Pi) \cup DN$. Then we have¹

Theorem 3.2. *The nonmonotonic semantics of $ST(\Pi)$ coincides with the stable semantics of Π .*

Thus, the stable causal interpretation of program rules provides an adequate description of logic programs under the stable semantics.

¹For disjunctive programs, this result has been proved in [GLL⁺04].

4 Causal Logics of Logic Programs

We will describe now two particular kinds of causal relations that will be shown to be adequate (and maximal) for logic programs under the supported and stable semantics.

4.1 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.

For a world α , $At(\alpha)$ will denote the set of its propositional atoms. A possible worlds semantics (W, R, V) is *inclusive*, if, for any $\alpha, \beta \in W$, $\alpha R \beta$ holds only if $At(\beta) \subseteq At(\alpha)$. Then the general completeness theorem implies

Corollary 4.1. *A causal relation is negatively closed if and only if it has an inclusive quasi-reflexive Kripke model.*

Let \Rightarrow_{Δ}^c denote a least negatively closed causal relation containing a causal theory Δ . Then by Theorem 3.2 we immediately obtain

Corollary 4.2. *The stable semantics of a program Π coincides with the nonmonotonic semantics of $\Rightarrow_{st(\Pi)}^c$.*

Thus, negatively closed causal relations constitute an adequate causal logic for logic programs under the stable semantics. Moreover, we are going to show now that this is actually a maximal such logic.

By [LPV01], 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 $\Rightarrow_{\Delta}^c = \Rightarrow_{\Gamma}^c$. Then we have

Theorem 4.3. *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 constitutes a causal counterpart of the corresponding results about strong equivalence of logic programs (see [LPV01, Tur01]).

Finally, note that any causal rule is reducible to a set of ‘clausal’ causal rules $\bigwedge a \Rightarrow \bigvee b$, where a and b are sets of literals (see [MT97]). Moreover, under the stable causal interpretation, each such causal rule corresponds to some program rule. Let $pr(\Delta)$ denote the program obtained in this way from a causal theory Δ . Then the following result shows, in effect, that any causal theory is reducible to a ‘stably equivalent’ logic program.

Theorem 4.4. *Any causal theory Δ is stably equivalent to $st(pr(\Delta))$.*

4.2 Tight causal relations

It turns out that the reasoning about logic programs under the supported interpretation admits a stronger causal logic described in the next definition.

Definition 4.1. A causal relation will be called *tight*, if it satisfies

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

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

Tight causal relations are ‘highly explanatory’ - they always explain disjunctions of two distinct literals that hold in a world. In practice, this means that only single literals have to be explained in such causal systems. For negatively closed causal relations, Tightness is reducible to

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

A possible world semantics will be called *tight*, if, for any worlds $\alpha, \beta \in W$, $\alpha R \beta$ holds only if $At(\beta) \div At(\alpha)$ contains no more than one propositional atom. Then the general completeness theorem gives us this time the following

Corollary 4.5. *A causal relation is tight if and only if it has a tight quasi-reflexive Kripke model.*

We denote by $\Rightarrow_{\Delta}^{ct}$ the least positively tight causal relation containing a causal theory Δ . Then the next result shows that positively tight causal relations are also adequate for logic programs under the supported semantics.

Theorem 4.6. *The supported semantics of a program Π coincides with the nonmonotonic semantics of $\Rightarrow_{su(\Pi)}^{ct}$.*

Moreover, we can show also that positively tight causal relations constitute a maximal causal logic for the supported semantics.

Definition 4.2. 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 causal theories Γ and Δ are *tightly equivalent*, if $\Rightarrow_{\Gamma}^{ct} = \Rightarrow_{\Delta}^{ct}$ (that is, each can be obtained from the other using the postulates of causal relations, Default Negation and Tightness). Then we have

Theorem 4.7. *Programs Π_1 and Π_2 are strongly sup-equivalent if and only if $su(\Pi_1)$ and $su(\Pi_2)$ are tightly equivalent.*

We are going to show now that any causal theory is ‘tightly equivalent’ to some logic program. This fact is based on the following reduction property:

Lemma 4.8. *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*

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

This result implies, in effect, that any causal theory is 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 \mathbf{f} . Notice now that, under the supported interpretation of program rules, a 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 4.9. *Any causal theory Δ is tightly equivalent to $su(npr(\Delta))$.*

As a ‘by-product’ of our construction, we obtain also

Corollary 4.10. *Any logic program is strongly sup-equivalent to a normal program with constraints.*

5 Partial Causation

Three- and four-valued logics have been used in logic programming for describing ‘partial’ generalizations of existing semantics. A generalization of causal relations, introduced below, will provide a logical representation for such partial semantics. The generalization is based on the ‘Australian Plan’ semantics for relevant logics [RPMB82] that suggested a ‘star-based’ alternative to the four-valued semantics (see also [FHV95]).

Definition 5.1. *A possible worlds *-model is a quadruple $(W, R, *, V)$, where (W, R, V) is a Kripke model, while $*$ is a function on W such that, for any $\alpha \in W$, $\alpha^{**} = \alpha$. A *-model is *regular*, if R is quasi-reflexive, and for any $\alpha, \beta \in W$, if $\alpha R \beta$, then $\alpha^* R \beta^*$.*

A new, *relevant negation* \sim is defined in this setting as follows:

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

As a result, we obtain also an extension of the language of causal logic with the new connective \sim . An axiomatization of this *partial causal logic* is 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 $\sim A \Rightarrow \sim B$.

In what follows, we will use a derived connective $*A$ defined as $\neg\sim A$. We will use also the same definition of the nonmonotonic semantics, namely the set of worlds (i.e., maximal consistent sets of propositions in the language $\{\wedge, \neg, \sim\}$) that are fixed points of \mathcal{C} . This notion of a world, however, is a syntactic counterpart of a four-valued interpretation; it is uniquely determined by the literals of the form $p, \neg p, *p, \sim p$.

5.1 The partial supported interpretation

A three-valued extension of Clark completion for normal programs has been suggested in [Fit85]; in later papers, Fitting has extended this construction to a four-valued setting. The semantic counterpart of this generalized completion are fixed-points of an ‘immediate consequence’ operator Φ on four-valued interpretations; we will call them *partial supported models*. Below we will briefly 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, this rule is equivalent to $*a, d, \neg b \Rightarrow \bigvee *c$. The effect of these two rules amounts to a *split*, or *doubling*, of a logic program, a construction used in [DM91, Wal93, Sch94] for representing 3-completion.

Let $PSU(\Pi)$ denote the causal theory corresponding to a program Π by the above interpretation, plus the set DN of default negation rules.

Theorem 5.1. *If Π is a normal program, then 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 the four-valued completion semantics.

5.2 The partial stable interpretation

The distinction between the partial supported and partial stable interpretations amounts again to treating positive premises in bodies 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. However, due to Invariance, the rule is equivalent to $d, \sim b \Rightarrow \bigwedge *a \rightarrow \bigvee *c$; taken together, these rules give again a logical counterpart of a doubled program [Wal93] (see also [Boc03a]). Now, if $PST(\Pi)$ denotes the causal theory obtained from Π by the above interpretation, plus DN , then the following result can be shown:

Theorem 5.2. *The nonmonotonic semantics of $PST(\Pi)$ coincides with the set of p -stable models of Π .*

P -stable models have been introduced in [Boc98] as a modification of partial stable models for disjunctive programs [Prz91] that satisfy the principle of partial deduction (alias GPPE). The modification has not changed, however, the correspondence with partial stable models of normal logic programs. The original partial stable semantics of Przymusiński can also be obtained in this framework, but this requires a further generalization of partial causation. Due to lack of space, however, we will leave it for another occasion.

6 Conclusions

The causal interpretation of logic programs suggests a natural and transparent logical representation of their meaning. In particular, it 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 . In addition, it allows to analyze the properties of logic programs directly in causal logic.

The emerging notion of a declarative meaning of logic programs can be immediately put to use in determining their *informational content*. The need in such a notion is especially pressing in the theory of *program updates*, since an update is normally required to produce a minimal change of the informational content.

It should be clear, however, that the suggested causal interpretation of logic programs requires 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, it constitutes a most immediate generalization of classical logic that allows for nonmonotonic 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 [GL91], the formalism of extended logic programs meets the default logic halfway. And indeed, most of the examples used in [GL91] for justifying the need for the second negation (such as legal or administrative regulations) have a distinct epistemic flavor. All such examples presuppose an epistemic reading of **not** p as “*p is not known to hold*”. This reading is distinct from an objective interpretation of **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 a further study.

References

- [AB94] K. R. Apt and R. N. Bol. Logic programming and negation: A survey. *Journal of Logic Programming*, 19,20:9–71, 1994.
- [ABW88] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, San Mateo, CA, 1988.
- [BG94] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19,20:73–148, 1994.
- [Boc98] A. Bochman. A logical foundation for logic programming II: Semantics of general logic programs. *Journal of Logic Programming*, 35:171–194, 1998.
- [Boc03a] A. Bochman. Collective argumentation and disjunctive logic programming. *Journal of Logic and Computation*, 9:55–56, 2003.
- [Boc03b] A. Bochman. A logic for causal reasoning. In *Proceedings IJCAI'03*, Acapulco, 2003. Morgan Kaufmann.
- [Cla78] K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.
- [DGM94] J. Dix, G. Gottlob, and V. Marek. Causal models for disjunctive logic programs. In P. Van Hentenryck, editor, *Proc. 11th Int. Conf. On Logic Programming, S. Margherita Ligure*, pages 290–302. MIT Press, 1994.
- [DM91] W. Drabent and M. Martelli. Strict completion of logic programs. *New Generation Computing*, 9:69–79, 1991.
- [FHV95] R. Fagin, J. Y. Halpern, and M. Y. Vardi. A nonstandard approach to the logical omniscience problem. *Artificial Intelligence*, 79:203–240, 1995.
- [Fit85] M. C. Fitting. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming*, 2:295–312, 1985.

- [GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proc. 5th International Conf./Symp. on Logic Programming*, pages 1070–1080, Cambridge, MA, 1988. MIT Press.
- [GL91] M. Gelfond and V. Lifschitz. Classical negation and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
- [GLL⁺04] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004.
- [IS98] K. Inoue and C. Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35:39–78, 1998.
- [LPV01] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
- [MT97] N. McCain and H. Turner. Causal theories of action and change. In *Proceedings AAAI-97*, pages 460–465, 1997.
- [Prz91] T. C. Przymusiński. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.
- [Rei78] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 119–140. Plenum Press, 1978.
- [RPMB82] R. Routley, V. Plumwood, R. Meyer, and R. Brady. *Relevant Logics and their Rivals*. Ridgeview, Ca., 1982.
- [Sch94] J. S. Schlipf. A comparison of notions of negation as failure. In G. Levi, editor, *Advances in Logic Programming Theory*, pages 1–53. Clarendon Press, Oxford, 1994.
- [She88] J. C. Shepherdson. Negation in logic programming. In J. Minker, editor, *Deductive Databases and Logic Programming*, pages 19–88. M. Kaufmann, 1988.

- [Tur01] H. Turner. Strong equivalence for logic programs and default theories (made easy). In T. Eiter, W. Faber, and M. Truszczyński, editors, *Proceedings Logic Programming and Nonmonotonic Reasoning, LPNMR'01*, volume 2173 of *LNAI*, pages 81–92, Vienna, 2001. Springer.
- [Wal93] M. Wallace. Tight, consistent, and computable completions for unrestricted logic programs. *Journal of Logic Programming*, 15:243–273, 1993.