A Pertinence Logic Characterization of Stable Models (Preliminary Report)

Ramón P. Otero*
AI Lab. - Dept. of Computer Science
University of Corunna
15071 Corunna, Galicia, SPAIN
email otero@dc.fi.udc.es

Abstract

On the line of Here-and-There logic by D. Pearce, this work introduces a semantics based on Pertinence Logic for the Stable Models semantics of Logic Programs proposed by M. Gelfond and V. Lifschitz. This semantics is useful for the strong equivalence of programs. Pertinence Logic is a causal formalism suitable for action descriptions, thus this characterization contributes to the relationship between the two major non-monotonic areas, namely, Logic Programming and Reasoning about Actions.

Introduction

Consider the following two logic programs[1]

P1	P2
$p \leftarrow not\, q$	$p \leftarrow$
$p \leftarrow q$	

there is only one model for both, $\{p\}$. But when a formula

$$q \leftarrow p$$

is added to both, the new program $\mathrm{P1} \cup \{q \leftarrow p\}$ has no stable model while $\mathrm{P2} \cup \{q \leftarrow p\}$ has $\{p, q\}$ as model[2]. So P1 and P2 have the same stable models but they do not behave the same way to the addition of more rules. The stable models of a program do not guarantee that the program will behave equivalently—the same models—to any other program with the same stable models, when the same rules are added to both (*strong equivalence*).

Pertinence Logic, that has been introduced as an action formalism based on causality, can be used to give an extended semantics to logic programs, thus the same set of models for two programs would mean that the programs are strongly equivalent.

This semantics defines for a program a set of models that is a superset of the stable models of the program, then a selection criterion will identify which ones of the models are stable. But in order to characterize strong equivalence the whole set of models should be used.

Pertinence Logic

The formulas of some subset of the propositional clausal Pertinence Logic (Otero 1997) are of the form

$$A_0 \leftarrow A_1, \ldots, A_m, notp\, A_{m+1}, \ldots, notp\, A_n \qquad (1)$$

where $n \geq m \geq 0$, and each A_i is an atom. A theory is a set of formulas. If a formula or a theory does not contain the *notp* operator we call it *positive*.

An interpretation gives to every atom *two* truth values simultaneously, one from the set {T, F}, meaning true and false, and another from the set {P, N}, meaning *pertinent* and *nonpertinent*. Resulting in four possible combined truth values for an atom: TP, TN, FP, FN; where, e.g., TP denotes *true* and *pertinent*, and FN denotes $false$ and *nonpertinent*.

An interpretation can be represented also by two sets of atoms (T, P), where the set T contains the atoms that are true independently on its pertinence value—TP and TN atoms—and the set P contains the atoms that are pertinent.

Intuitively, an atom is TP when it is true and *caused*, while TN means that the atom is true but not caused. Closer to Logic Programming intuitions, TP can be understood as "true and proved", while TN would be "true but not proved".

We define the truth and pertinence of the formula (1) in an interpretation I as follows.

- The body $A_1, \ldots, A_m, notpA_{m+1}, \ldots, notpA_n$, is true iff
  (i) $A_i \in T$, for all $i : 1, \ldots, m$ (all are TP or TN) and
  (ii) $A_j \notin T$, for all $j : m+1, \ldots, n$ (all are FP or FN).

  It is pertinent iff
  (iii) $A_i \in P$, for all $i : 1, \ldots, m$ (all are TP or FP) and
  (iv) $A_j \notin P$, for all $j : m+1, \ldots, n$ (all are TN or FN).
- $A_0 \leftarrow A_1, \ldots, A_m, notp\, A_{m+1}, \ldots, notp\, A_n$ is true iff it is not the case that the body is true and pertinent (TP) and the head (A_0) is not TP.

  The implication formula is pertinent in I iff the body is pertinent or the head is pertinent.

*Most of this work was done while I was visiting professor at CS, University of Texas at El Paso in Spring 2000.

[1]From Problem 51 by V. Lifschitz on the Texas Action Group (TAG) list.

[2]Answer by M. Gelfond to Problem 51 on TAG list.

A formula $A_0 \leftarrow$ is true iff A_0 is TP in I, and it is always pertinent. It is an abbreviation for $A_0 \leftarrow \top_{TP}$, where $\top_{TP}$ is always true and pertinent. The negation *notp* swaps $T \leftrightarrow F$ and $P \leftrightarrow N$.

Note that the implication formula is true if it is not pertinent, i.e., no interpretation makes it FN. Furthermore, the implication formula is true if the body is not pertinent.

Intuitively, in Pertinence Logic, if all the positive atoms $A_1, \ldots, A_m$ are true, and all the negative atoms $A_{m+1}, \ldots, A_n$ are false, this is not enough to imply a particular value in the head of the formula. Only when all the positive atoms have the pertinent value TP and all the negative atoms have the nonpertinent value FN, then the value TP is implied for the head.

An interpretation I is a *model* of a Pertinence Logic theory Γ iff every formula of Γ is true in I.

Example 1 Consider the Pertinence theory $\{p \leftarrow q\}$, there are 16 interpretations. The counter-models of $p \leftarrow q$ are $(\{p,q\},\{q\})$, $(\{q\},\{p,q\})$ and $(\{q\},\{q\})$, i.e., when q is TP, p cannot be TN, FP nor FN. The models are the other 13 interpretations. □

We will be interested also in a selection of the models of a theory Γ. A model $M = (T, P)$ of a theory Γ is a *causal* model of Γ iff there is no other model $M' = (T', P')$ of Γ such as $T = T'$ and $P' \subset P$.

For causal models, the models with the minimal subset of the pertinent atoms are preferred, when they have the same true atoms. The causal models define a Nonmonotonic Pertinence Logic.

Example 2 The causal models of $\Gamma = \{p \leftarrow q\}$ are $(\{p,q\},\{\})$, $(\{p\},\{\})$, $(\{q\},\{\})$ and $(\{\},\{\})$. Instead, the unique causal model of $\Gamma' = \{q \leftarrow, p \leftarrow q\}$ is $(\{p,q\},\{p,q\})$. □

Logic Programs and Pertinence Logic

A logic program is a set of rules of the form

$$A_0 \leftarrow A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n \qquad (2)$$

where $n \geq m \geq 0$, and each A_i is an atom. If a rule or a program does not contain the *not* operator it is called *positive*.

The stable model semantics of a logic program (Gelfond & Lifschitz 1988) is defined in two steps. First let Π be a positive program, then the *stable models* are the minimal sets of atoms M that satisfy the condition: For each rule

$$A_0 \leftarrow A_1, \ldots, A_m$$

from Π, if $A_i \in M$, for all $i : 1, \ldots, m$, then $A_0 \in M$.

Now let Π be a general program. For any set M of atoms, let Π^M be the program obtained from Π by deleting

1. each rule that has a formula *not* A in its body with $A \in M$, and
2. all formulas of the form *not* A in the bodies of the remaining rules.

The program Π^M is positive; if M is the stable model of this program then M is a stable model of Π.

Given a logic program rule ϕ, in the form (2), we identify the rule with the formula $Pert(\phi)$ in the form (1) in Pertinence Logic. The negation as failure *not* , become negation on pertinence and true *notp* , the implication become the Pertinence implication. A logic program Π will be identified with the theory $Pert(\Pi)$ containing the formulas corresponding to its rules.

Definition 1 (p-stable model) Given a Pertinence Logic theory Γ, the models that only mention atoms TP and FN are the *p-stable* models. □

A p-stable model $M = (T, P)$ verifies that $T = P$. Define also *p-stable causal models* the causal models that are p-stable.

Example 3 The p-stable causal model of the theory $\Gamma = \{p \leftarrow q\}$ is $(\{\},\{\})$. The p-stable causal model of $\Gamma' = \{q \leftarrow, p \leftarrow q\}$, is $(\{p,q\},\{p,q\})$. □

The procedure to get the stable models of a program Π will be:

1. Translate the program Π to the Pertinence theory $Pert(\Pi)$. Get the models of $Pert(\Pi)$ (monotonic semantics).
2. Get the causal models of $Pert(\Pi)$ (minimize pertinence).
3. Get the p-stable causal models of $Pert(\Pi)$, these are the stable models of Π simply considering TP is true and FN is false.

The intuition on this relation between Pertinence Logic and the stable models semantics of LP is as follows. The minimization of pertinence does the justification of the atoms in the stable model, i.e., if there is a possibility for an atom not to be pertinent then it is nonpertinent so it is not proved. The idea behind choosing only models with TP and FN is: every atom that is true in LP must be 'proved', so it must be pertinent and an atom TN does not make sense in LP; every atom that is false in LP is false by default so an atom FP (proved false) does not make sense in LP (at least in non-extended programs).

Example 4 a) For a program $\Pi = \{p \leftarrow not\ q, q \leftarrow not\ p\}$, the models of the corresponding theory $Pert(\Pi) = \{p \leftarrow notp\ q, q \leftarrow notp\ p\}$ are

$(\{p,q\},\{p,q\})$, $(\{p,q\},\{p\})$, $(\{p\},\{p,q\})$, $(\{p\},\{p\})$, $(\{p,q\},\{q\})$, $(\{p,q\},\{\})$, $(\{p\},\{q\})$, $(\{q\},\{p,q\})$, $(\{q\},\{p\})$, $(\{\},\{p,q\})$, $(\{q\},\{q\})$,

If p is FN then q must be TP, and symmetrically. The causal models are $(\{p,q\},\{\})$, $(\{p\},\{p\})$, $(\{p\},\{q\})$, $(\{q\},\{p\})$, $(\{q\},\{q\})$, and $(\{\},\{p,q\})$.

The p-stable causal models are $(\{p\},\{p\})$ and $(\{q\},\{q\})$.

b) For a program $\Pi = \{p \leftarrow not\ p\}$, the models of the corresponding theory $Pert(\Pi) = \{p \leftarrow notp\ p\}$ are

$p \setminus q$	TP		TN	FP		FN
TP	$\underline{(\{p,q\},\{p,q\})}$	$\longrightarrow$	$(\{p,q\},\{p\})$	$(\{p\},\{p,q\})$	$\longrightarrow$	$\underline{(\{p\},\{p\})}$
	$\downarrow$	$\searrow$	$\downarrow$	$\downarrow$	$\searrow$	$\downarrow$
TN	$(\{p,q\},\{q\})$	$\longrightarrow$	$(\{p,q\},\{\})$	$(\{p\},\{q\})$	$\longrightarrow$	$(\{p\},\{\})$
FP	$(\{q\},\{p,q\})$	$\longrightarrow$	$(\{q\},\{p\})$	$(\{\},\{p,q\})$	$\longrightarrow$	$(\{\},\{p\})$
	$\downarrow$	$\searrow$	$\downarrow$	$\downarrow$	$\searrow$	$\downarrow$
FN	$\underline{(\{q\},\{q\})}$	$\longrightarrow$	$(\{q\},\{\})$	$(\{\},\{q\})$	$\longrightarrow$	$\underline{(\{\},\{\})}$

Table 1: Pertinence models for two variables.

$(\{p\},\{p\})$, $(\{p\},\{\})$ and $(\{\},\{p\})$. So p cannot be FN. The causal models are $(\{p\},\{\})$ and $(\{\},\{p\})$. Because p TP is minimized by p TN. There is no p-stable causal model because in both $T \neq P$. □

Example 5 For a program $\Pi_1 = \{p \leftarrow not\, q, p \leftarrow q\}$, the models of the corresponding theory $Pert(\Pi_1) = \{p \leftarrow notp\, q, p \leftarrow q\}$ are

$$\begin{array}{llll} (\{p,q\},\{p,q\}), & (\{p,q\},\{p\}), & (\{p\},\{p,q\}), & (\{p\},\{p\}), \\ & (\{p,q\},\{\}), & (\{p\},\{q\}), & \\ & (\{q\},\{p\}), & (\{\},\{p,q\}), & \\ & (\{q\},\{\}), & (\{\},\{q\}), & \end{array}$$

If q is FN or TP then p must be TP. The causal models are $(\{p,q\},\{\})$, $(\{p\},\{p\})$, $(\{p\},\{q\})$, $(\{q\},\{\})$, and $(\{\},\{q\})$. The p-stable causal model is $(\{p\},\{p\})$.

For a program $\Pi_2 = \{p \leftarrow\}$, the models of the corresponding theory $Pert(\Pi_2) = \{p \leftarrow\}$ are

$$(\{p,q\},\{p,q\}), \quad (\{p,q\},\{p\}), \quad (\{p\},\{p,q\}), \quad (\{p\},\{p\}),$$

Thus p is TP. The causal models are $(\{p,q\},\{p\})$, and $(\{p\},\{p\})$. The p-stable causal model is $(\{p\},\{p\})$.

Note that in Pertinence Logic $Pert(\Pi_2) \models Pert(\Pi_1)$, but $Pert(\Pi_1) \not\models Pert(\Pi_2)$. In Nonmonotonic Pertinence Logic (causal models) $Pert(\Pi_2) \not\models Pert(\Pi_1)$, and $Pert(\Pi_1) \not\models Pert(\Pi_2)$. But considering the p-stable causal models $Pert(\Pi_2) \models Pert(\Pi_1)$, and $Pert(\Pi_1) \models Pert(\Pi_2)$.

The models of the formula $q \leftarrow p$ are

$$\begin{array}{llll} (\{p,q\},\{p,q\}), & & & \\ (\{p,q\},\{q\}), & (\{p,q\},\{\}), & (\{p\},\{q\}), & (\{p\},\{\}), \\ (\{q\},\{p,q\}), & (\{q\},\{p\}), & (\{\},\{p,q\}), & (\{\},\{p\}), \\ (\{q\},\{q\}), & (\{q\},\{\}), & (\{\},\{q\}), & (\{\},\{\}), \end{array}$$

If p is TP then q must be TP.

When this formula is added to $Pert(\Pi_1)$ the causal models are $(\{p,q\},\{\})$, $(\{p\},\{q\})$, $(\{q\},\{\})$, and $(\{\},\{q\})$. There is not p-stable causal model.

When this formula is added to $Pert(\Pi_2)$ there is only one model, $(\{p,q\},\{p,q\})$, which is causal and p-stable. □

A Graphic Device for Logic Programs

Table 1 shows the 16 interpretations in Pertinence Logic for two propositional variables. The arrows follow the minimization directions for causal models. The underlined models are the four p-stable models.

Figure 1 is a graphic representation of table 1. The p-stable models are represented by a black dot, while the rest of the possible models are white dots.

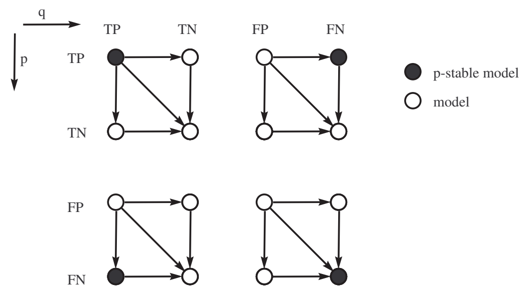

Figure 1: Pertinence models for two variables.

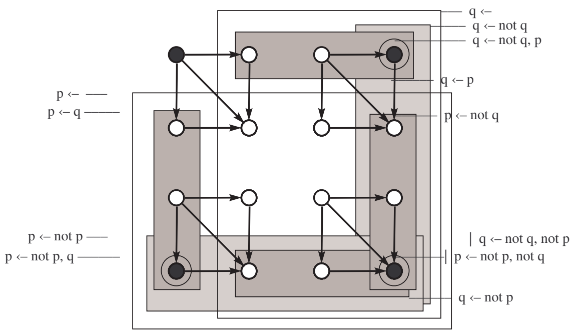

Figure 2: Pertinence counter-models for the 'LP-form' formulas with two variables.

Figure 2 is an extension of figure 1. The grey rectangles include the counter-models of the corresponding formula with two variables. This figure can be used to identify the stable models of a program with two variables. The procedure is:

1. For each rule ϕ of the program Π, *delete* the dots that are included in the rectangle named after the rule. The dots that remain represent the models of the corresponding theory $Pert(\Pi)$.
2. Work the minimization: From each dot M, if you can *move* following one arrow to another dot M' then delete the first one, M. The dots that remain represent the causal models of the corresponding theory $Pert(\Pi)$.
3. Look at the *black* dots remaining from the previous step, they represent the p-stable causal models of the corresponding theory $Pert(\Pi)$, and if you look only at T and F, they represent the stable models of the program.

Figure 2 and the previous procedure can be used to identify if two programs are strong equivalent. Do only step 1 of the previous procedure on the two programs, if the dots that remain for both programs are the same then the programs are strong equivalent.

Working with figure 2 it is clear that some dots go always together, e.g. it is impossible to delete in step 1, separately, one of the four inner white dots without deleting the other three. The restricted form (1) of the formulas that is enough to translate any normal logic program to Pertinence, does not have enough syntax to represent formulas that would distinguish these models.

As we are interested in logic programs, we can reduce the set of possible models in Pertinence Logic, so to better match them with logic programs.

Definition 2 (p-normal model) Given a Pertinence theory Γ, the models that only mention atoms TP, TN and FN are the *p-normal* models. □

Given a model $M = (T, P)$ it is p-normal if and only if $P \subseteq T$.

Alternatively, a *Normal* Pertinence Logic can be defined with only three truth values for the atoms: TP, TN and FN.

Intuitively, from a Logic Programming point of view, Normal Pertinence will consider only the truth values that minimize p-stable models, which is equivalent to forget (or do not have) the models that include FP values.

Table 2 shows the 9 interpretations in Normal Pertinence for two propositional variables. This table is table 1 without the row and the column for value FP.

$p \setminus q$	TP		TN	FN
TP	$(\{p,q\},\{p,q\})$	$\longrightarrow$	$(\{p,q\},\{p\})$	$(\{p\},\{p\})$
	$\downarrow$	$\searrow$	$\downarrow$	$\downarrow$
TN	$(\{p,q\},\{q\})$	$\longrightarrow$	$(\{p,q\},\{\})$	$(\{p\},\{\})$
FN	$(\{q\},\{q\})$	$\longrightarrow$	$(\{q\},\{\})$	$(\{\},\{\})$

Table 2: Normal Pertinence models for two variables.

Figure 3 is the corresponding reduction of figure 2. The formulas $p \leftarrow p$ and $q \leftarrow q$, and the others with any other literal in the body, e.g. $p \leftarrow p, q$, are tautologies (no counter-models). As well as formula $p \leftarrow q, not\, q$ and similar[3]. This figure is enough to identify the stable models of a program with two variables. The procedure is the same as before.

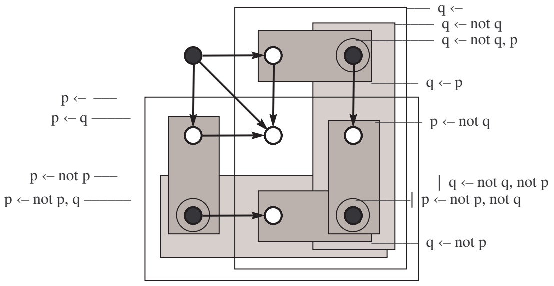

Figure 3: Normal Pertinence counter-models for the 'LP-form' formulas with two variables.

Furthermore, figure 3 can be used to identify strong equivalence among programs. The programs are strong equivalent if after step 1 of the previous procedure, the models are the same.

Compared with figure 2, in figure 3 every dot belongs to a different set of rectangles.

Example 6 Consider the program Π_1

$$\begin{array}{l} q \leftarrow p \\ q \leftarrow not\, p \end{array}$$

is it strong equivalent to itself after the rule $p \leftarrow not\, q$ is added to it, $\Pi_2 = \Pi_1 \cup \{p \leftarrow not\, q\}$?

There is only one stable model for both, $\{q\}$.

After step 1 of the procedure both programs have the same models in Pertinence except that Π_1 has one more model than Π_2, viz. $(\{p\},\{\})$.

Thus after step 1 we can not say whether these two programs are equivalent or not. But if we try all the possible rules to be added, all the possible extended programs have the same stable models.

Looking at the square of figure 4, only two black dots remain after Π_1 is runned in it. The white dots only affect the black dots to which they are connected by the minimization arrows. So white dots for which the connected black dot is not a model, as it is the case in this example, never show up. □

Definition 3 (p-dynamic-normal model) Given a Pertinence theory Γ, a model $M = (T, P)$ is a *p-dynamic-normal* model iff $P \subseteq T$ and there is a p-stable model $M' = (T', T')$ of Γ such as $T = T'$. □

Every model that is not p-normal is not a p-dynamic-normal model, for any theory. Also every p-stable model is a p-dynamic-normal model.

[3]There are two kinds of tautologies: i) when the head is also present in the body, and ii) when the body has a complementary pair of literals.

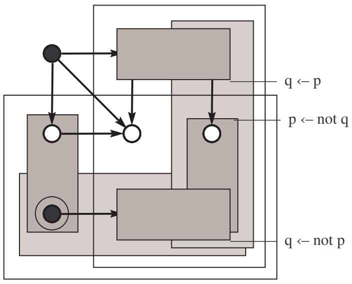

Figure 4: Isolated Models in Normal Pertinence.

Let us extend the graphic procedure to identify strong equivalence. The first step is as before but a new second step is added. The procedure is:

1. For each rule ϕ of the program Π, *delete* the dots that are included in the rectangle named after the rule. The dots that remain represent the models of the corresponding theory $Pert(\Pi)$.
2. Normalize: Delete all white dots that do not have an incoming arrow from a remaining black dot.

To find whether two programs are strong equivalent, apply the procedure; if the dots that remain are the same for both programs, they are strong equivalent.

Characterization Results

In this section it is shown that given a logic program, the p-stable causal models of the corresponding pertinence theory characterize the stable models of the program. To this end a 'Gelfond-Lifschitz'-like transformation is defined in pertinence.

Definition 4 (G-L transformation in Pertinence) Let $M = (T, P)$ be an interpretation of a Pertinence Logic theory Γ with formulas in the form (1), let Γ^T be the theory obtained from Γ by deleting

1. each implication formula that has a subformula $notp\, A$ in its body with $A \in T$, and
2. all subformulas of the form $notp\, A$ in the bodies of the remaining implications. □

The theory Γ^T is positive.

Lemma 1 *If $M = (T, P)$ is a model of a Pertinence Logic theory Γ, with formulas in the form (1), such that $P \subseteq T$ (p-normal model), then $M = (T, P)$ is a model of Γ^T.*

Proof. It is only needed to show that M is a model of the formulas that remain—case (2)—reduced in Γ^T. In case (2), when the implication has subformulas $notp A_j$, then all $A_j \notin T$, otherwise the whole implication is deleted—case (1).

Let $\phi^T \in \Gamma^T$ be the reduced formula of $\phi \in \Gamma$. Consider that M is not a model of ϕ^T, then the body is true and pertinent and the head is not TP. If the body of ϕ^T is TP in M, so it is the body of ϕ in M, because the rest of the subformulas $notp\, A_j$ of ϕ are true and pertinent (all $A_j \notin T$, and as $P \subseteq T$, all $A_j \notin P$). (If the formula is a fact in Γ its 'body' is of course TP.) If the head is not TP then M is not a model of ϕ what contradicts the assumption. □

There are models of Pertinence theories that are not models of its G-L transformation. Consider the theory $\Gamma = \{p \leftarrow notp\, p\}$ and the model $M = (T, P) = (\{\}, \{p\})$, i.e., p is FP. It is a model of Γ because $notp\, p$ is TN in M, so the body is true but it is not pertinent, then the head does not need to be TP, and FP in the head satisfies the implication. The theory Γ^T is $\{p \leftarrow\}$ because $p \notin T$. There is only one model of Γ^T, namely $M' = (\{p\}, \{p\})$, thus M is not a model of Γ^T.

This also shows that the causal models of a Pertinence theory Γ does not need to be causal models of Γ^T. But the p-stable models of a theory are p-stable models of its G-L transformation.

Lemma 2 *Given a logic program Π, and $M = (T, P)$ a model of the corresponding Pertinence theory $Pert(\Pi)$, such that $P \subseteq T$, then P is a model of the positive program Π^T.*

Proof. Let $M = (T, P)$ be a model of $Pert(\Pi)$, such that $P \subseteq T$, by Lemma 1, $M = (T, P)$ is a model of the reduct $Pert(\Pi)^T$. Suppose that P is not a model of Π^T. Then there is a reduced rule ϕ^T in Π^T that is not satisfied by P, i.e., the body is in P and the head is not in P. Consider the corresponding formula $Pert(\phi)^T = A_0 \leftarrow A_1, \ldots, A_m$ in $Pert(\Pi)^T$. Then $A_i \in P$ for all $i : 1, \ldots, m$ and $A_0 \notin P$. As $P \subseteq T$, $A_i \in T$ for all $i : 1, \ldots, m$. So the body of the formula is true and pertinent and the head is not pertinent in M, what contradicts the fact that M is a model of $Pert(\phi)^T$. □

Lemma 3 *Given a logic program Π, and a set P satisfying Π^T, such that $P \subseteq T$, then $M = (T, P)$ is a model of the corresponding Pertinence theory $Pert(\Pi)$.*

Proof. Let Π be a logic program and P a set that satisfies Π^T. Consider $M = (T, P)$, we verify that this is a model of $Pert(\Pi)$.

Consider the rules ϕ that are in Π^T but reduced, we verify that M is a model of the corresponding formulas $Pert(\phi) \in Pert(\Pi)$. Since P satisfies the condition on stable models for each reduced rule ϕ^T in Π^T, if it is the case that $A_i \in P$ for all $i : 1, \ldots, m$, then $A_0 \in P$.

Also $A_0 \in T$, since $P \subseteq T$. So when it is the case of the semantics of the implication formula in Pertinence, namely the body is true and pertinent, then the head is also true and pertinent in M.

Consider the rules ψ of Π that are not in Π^T, then $A_j \in T$ for some $j : m+1, \ldots, n$, implying that the body of the implication $Pert(\psi)$ is not true in M, thus M is a model of $Pert(\psi)$. □

The condition on $P \subseteq T$ in Lemma 3 is essential. Consider the program $\Pi = \{p \leftarrow\}$, and $T = \{\}$, then $P = \{p\}$ satisfies Π^T. But $M = (T, P)$ is not a model of $Pert(\Pi)$.

Theorem 1 *Every stable model of a logic program* Π *is a p-stable causal model of* $Pert(\Pi)$.

Proof. Suppose that T is any stable model of Π; then it is the minimal set satisfying Π^T. Consider the corresponding p-stable model $M = (T, P)$, where $T = P$. So it is a p-stable model. By Lemma 3 M is a model of $Pert(\Pi)$.

We show that M is a causal model of $Pert(\Pi)$. Suppose not, then there is a model $M' = (T', P')$ of $Pert(\Pi)$ with $T' = T$ and $P' \subset P$, i.e. $P' \subset T$. By Lemma 2, P' is a model of Π^T, and as $P' \subset T$ then T is not a minimal set satisfying Π^T, what contradicts the fact that T is a stable model. $\square$

Theorem 2 *Given a logic program* Π*, every p-stable causal model of* $Pert(\Pi)$ *is a stable model of* Π.

Proof. Let Π be a logic program and $M = (T, P)$ a p-stable causal model of $Pert(\Pi)$, thus $T = P$. We verify that P is the minimal set satisfying Π^T. By Lemma 2, P is a model of Π^T. Suppose it is not the minimal set that satisfies Π^T then there is a set $P' \subset P$ that satisfies Π^T. Consider $M' = (T, P')$ by Lemma 3 this is a model of $Pert(\Pi)$ what contradicts the fact that $M = (T, P)$ is a causal model. $\square$

Related Work

This work is inspired in previous work by D. Pearce (Pearce 1997). He characterizes stable models semantics as a nonmonotonic logic—Equilibrium Logic—a form of minimal model reasoning in the logic of here-and-there. The monotonic logic of here-and-there would correspond to (monotonic) Pertinence Logic while Equilibrium Logic would correspond to NM-Pertinence logic.

The models structure of these logics can be represented by a pair of sets of atoms (H, T), denoted 'here' and 'there', where[4] $H \subseteq T$.

For an implication formula in the form (2) T satisfies the formula iff whenever

(i) $A_i \in T$, for all $i : 1, \ldots, m$ and

(ii) $A_j \notin T$, for all $j : m+1, \ldots, n$,

then $A_0 \in T$.

And H satisfies the formula iff T satisfies the formula and whenever

(i) $A_i \in H$, for all $i : 1, \ldots, m$ and

(ii) $A_j \notin T$, for all $j : m+1, \ldots, n$,

then $A_0 \in H$.

If both T and H satisfy the formula then (H, T) is a model of it.

This defines the monotonic logic of here-and-there. The minimization criterion for the (nonmonotonic) Equilibrium Logic is as follows.

[4]This relation belongs to the structure of the model, so no models are defined that do not verify this. This condition is essential, in the sense that the semantics is not enough to forbid these structures.

Given a theory Γ, a model (H, T) is h-minimal iff there is no model of Γ, (H', T') such as $T = T'$ and $H' \subset H$.

Given a theory Γ, a model (H, T) is an equilibrium model iff it is h-minimal and $H = T$.

The direct relation with Pertinence identify H with the set P of pertinent atoms, and T with the set T of true atoms. Thus the h-minimal models would correspond to causal models in Pertinence. And equilibrium models to p-stable causal models. (Both corresponding to the stable models of the program.)

Table 3 shows the 9 interpretations in here-and-there logic for two propositional variables. This table is similar to table 2 in Normal Pertinence but the sets are (T, H) instead of (T, P). The arrows follow the minimization directions for h-minimal models. The underlined models are the four that verify $T = H$ (possible equilibrium models).

$p \setminus q$	TH		T	
TH	$\underline{(\{p,q\},\{p,q\})}$	$\longrightarrow$	$(\{p,q\},\{p\})$	$\underline{(\{p\},\{p\})}$
	$\downarrow$	$\searrow$	$\downarrow$	$\downarrow$
T	$(\{p,q\},\{q\})$	$\longrightarrow$	$(\{p,q\},\{\})$	$(\{p\},\{\})$
	$\underline{(\{q\},\{q\})}$	$\longrightarrow$	$(\{q\},\{\})$	$\underline{(\{\},\{\})}$

Table 3: Here-and-there models (T, H) for two variables.

Figure 5 is a graphic representation of table 3. The possible equilibrium models are represented by a black dot, while the rest of the possible models are white dots. The grey rectangles include the counter-models of the corresponding formula with two variables.

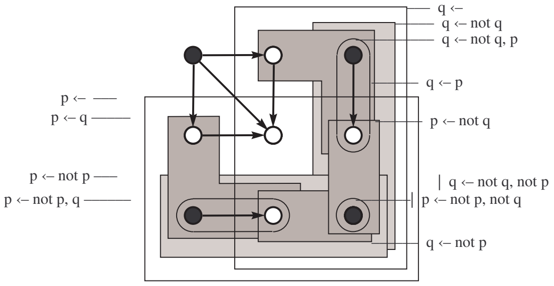

Figure 5: Here-and-there counter-models for the 'LP-form' formulas with two variables.

This figure is enough to identify the stable models of a program with two variables. The procedure is similar to the one in pertinence logic.

Furthermore, figure 5 can be used to identify strong equivalence among programs.

Compared with the square of Normal Pertinence (figure 3), figure 5 has four rules with a different set of counter-models. The rule $p \leftarrow q$, has one additional counter-model (same for $q \leftarrow p$), and the rule

$p \leftarrow not\, p, q$ (same for $q \leftarrow not\, q, p$), has also one additional counter-model.

This is a graphic explanation how here-and-there is closer to stable models than pertinence. These interpretations are models of the pertinence implication, and only have not to be considered for strong equivalence of programs (p-dynamic-normal models). But these interpretations are already counter-models of the here-and-there implication. Here-and-there requires rules to satisfy T, while pertinence logic does not. An additional formula can be added to the translation of each rule to pertinence logic to achieve this behavior

$$A_1 \wedge \ldots \wedge A_m \wedge \neg A_{m+1} \wedge \ldots \wedge \neg A_n \supset A_0 \qquad (3)$$

with a 'classical' semantics such that when the body is true the head has to be true (independently on its pertinence).

Here-and-there as a intuitionistic logic, verifies the contraposition of implication in one direction

$$\psi \leftarrow \phi \models \neg\phi \leftarrow \neg\psi \qquad (4)$$

Pertinence logic, as a causal formalism, does not verify this contraposition. (This fact is basic for the solution to the ramification problem in actions.)

Discussion

While action descriptions in pertinence logic use the same causal implication than here for action laws and domain constraints, the difference is in the use of negation. The negation typically used in actions is more 'classical', viz., negation only in truth. But other kind of negations have shown its usefulness, e.g., a negation in pertinence for diagnosis of dynamic systems (Otero & Otero 2000).

The minimization criterion used here is the same as in the other applications of pertinence logic: minimize pertinence for the same truth. In this sense the axiom for LP would be

$$nop\, p \supset notp\, p$$

meaning "what is nonpertinent has to be false (and nonpertinent)". This axiom actually selects the p-stable models.

In action descriptions, the focus is not restricted to a fixed subset of the models (p-stable) but all the models are considered. A typical inertia axiom in action is $nop\, p_i \supset (p_i \equiv_T p_{i-1})$, meaning "what is nonpertinent has the same truth as before (persist)".

Pertinence logic shows up as a common basis for the two major non-monotonic areas.

Pertinence logic constitutes also an alternative definition of 'strong' negation in logic programs—the atoms that are false and pertinent—that is suitable for actions. This is also a difference with intuitionistic logics, that are defined with the requirement of inclusion between two consecutive worlds.

Action domains is one of the most important application of stable models. It would be interesting to compare, in the light of this characterization, these two ways to describe a domain.

The differences between here-and-there and pertinence do not seem important when the subject is stable models. Though there are differences, for instance, $not\, q \leftarrow not\, p$ is equivalent to $p \leftarrow not\, p, q$ in here-and-there, but it is not the case in pertinence. There are other characterizations of stable models based on 3-valued logics (e.g. (Przymusinski 1990)) but they do not constitute a monotonic characterization.

For strong equivalence, the addition of the same set of formulas to two pertinence theories corresponding to two logic programs deletes the same models in both (pertinence logic is monotonic). Thus if the two theories had the same set of models before the addition, they will continue to have the same subset after any addition, and the stable models will be the same.

Logic programming has a restricted syntax so actually it is not needed to have a complete coincidence on the monotonic models of the pertinence theories. (Furthermore, stable models focus on a fixed set of models (p-stable), though this would not be the case for other semantics of LP when viewed from pertinence logic.) A similar study can be carried out in action descriptions where the complete syntax of pertinence logic is rarely used.

Acknowledgements. This work benefits from discussions with Michael Gelfond, I am very grateful to him. This research is partially supported by Government of Spain grant PB97-0228 and in part by the Government of Galicia, under grant 10501B98.

References

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Logic Programming: Proc. of the Fifth Int'l Conf. and Symp.*, 1070–1080.

Otero, M., and Otero, R. P. 2000. Using causality for diagnosis. In Darwiche, A., and Provan, G., eds., *Eleventh Int'l Workshop on Principles of Diagnosis. Proc. DX 00*, 171–176.

Otero, R. P. 1997. Pertinence logic for reasoning about actions and change. Technical Report TR-AI-97-03, AILab, University of Corunna, Galicia, Spain.

Pearce, D. 1997. A new logical characterisation of stable models and answer sets. In J. Dix, L. P., and Przymusinski, T., eds., *Non-monotonic Extensions of Logic Programming. Proc. NMELP 96, LNAI 1216.*

Przymusinski, T. C. 1990. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae* 13(4):445–464.