An Abductive Paraconsistent Semantics – MH_P

Mário Abrantes[1]* and Luís Moniz Pereira[2]

[1] Escola Superior de Tecnologia e de Gestão
Instituto Politécnico de Bragança
Campus de Santa Apolónia, 5300-253 Bragança, Portugal
mar@ipb.pt

[2] Centro de Inteligência Artificial (CENTRIA), Departamento de Informática
Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
2829-516 Caparica, Portugal
lmp@fct.unl.pt

Abstract. In this paper we present a paraconsistent abdutive semantics for extended normal logic programs, the *paraconsistent minimal hypotheses* semantics MH_P. The MH_P is a semantics of *total paraconsistent models* wich combines the merits of two already existing semantics: it inherits the existence property of the abductive *minimal hypotheses* semantics MH [1], which is a semantics of *total models*, and the property of detection of support on contradiction of the paraconsistent *well-founded semantics with explicit negation* $WFSX_P$ [2], which is a semantics of *partial paraconsistent models*. The MH_P enjoys also the property of *simple relevance*, which permits top-down query answering for brave reasoning purposes. Besides, the MH_P lends itself to various types of skeptical and brave reasoning, which include the possibility of drawing conclusions from inconsistent models in a nontrivial way. The MH_P coincides with the MH on normal logic programs, and with the $WFSX_P$ on stratified extended programs.

Keywords: Hypotheses, Semantics, Abduction, Total Paraconsistent Model, Partial Paraconsistent Model, Paraconsistency.

1 Introduction

In this work we present an abduction based paraconsistent semantics, for extended logic programs, the MH_P, wich combines the merits of two already existing semantics: an abductive semantics, the *minimal hypotheses* semantics MH [1], and a paraconsistent semantics, the *well-founded semantics with explicit negation* $WFSX_P$ [2]. We first expound on the general merits of abductive semantics, and then highlight the advantages of paraconsistent semantics.

Abductive logic programming is an extension of logic programming to perform abductive reasoning [3]. The following example gives a glimpse on how it can be used to get a 2-valued semantics for every normal logic program.

* Partially supported by Fundação para a Ciência e Tecnologia and Instituto Politécnico de Bragança grant PROTEC : SFHR/49747/2009.

Example 1. In order for program $P = \{a \leftarrow not\ b,\ b \leftarrow not\ c,\ c \leftarrow not\ a\}$, that has no stable models, to have a 2-valued model, a subset of $\{a, b, c\}$ must be a part of the positive literals of the model. This can be achieved by considering *abductive extensions* [3] $P \cup H$ of program P, where H is a subset of $\{a, b, c\}$ such that $P \cup H$ has a single stable model [4]. For example, if we consider the explanation $H = \{a\}$, the stable model obtained for $P \cup H$ is $\{a, b\}$. In case we take, for instance, the explanation $\{b\}$ (resp. $\{c\}$), we get the model $\{b, c\}$ (resp. $\{c, a\}$). Each set H is called *hypotheses set* [1] for the corresponding model, and is an *abductive explanation* [3] for program P.

Abduction allows us to envisage loops in normal logic programs as semantic choice devices. This is one of the main features of the *minimal hypotheses* semantics MH, presented in section 3. MH takes as assumable hypotheses of a normal logic program P the atoms that appear default negated in the *layered remainder* $\mathring{P}$, which is a transformed of the original program P. In the example above, as $P = \mathring{P}$, the MH models of program P are $\{a, b\}$, $\{b, c\}$, $\{c, a\}$, with hypotheses sets respectively, $\{a\}$, $\{b\}$, $\{c\}$. The hypotheses sets are minimal with respect to set inclusion. We next highlight the advantages of having paraconsistent semantics. Several authors, [5–10], have stressed the need to endow normal logic programs with a second kind of negation operator, the explicit negation '$\neg$', for representing contradictory knowledge, in addition to the negation-as-failure (or default negation) operator, '*not*', used for representing incomplete information. There are plenty of examples that display the need for explicitly negated literals in logic programming, both in the heads and in the bodies of the rules. The following is a typical example.

Example 2. (adapted from [11]) Consider the statement "Penguins do not fly". This statement may be represented within logic programming by $no\_fly(X) \leftarrow penguin(X)$. Meanwhile, if additionally we wish to represent the statement "Birds fly", $fly(X) \leftarrow birds(X)$, no connection results between the predicates $no\_fly(X)$ and $fly(X)$, although the intention of the programmer is to set them as contradictory. In this case it is suitable to have the rule $\neg fly(X) \leftarrow penguin(X)$ instead of $no\_fly(X) \leftarrow penguin(X)$, since a semantics that deals with the operator $\neg$ will by definition consider predicates $fly(X)$ and $\neg fly(X)$ as contradictory opposites.

As a consequence of the need for an explicit negation operator, a number of semantics that interpret this type of operator have been proposed – those for *extended normal logic programs*, *ELPs*. Among them are the *paraconsistent* semantics [2, 12–15], i.e. semantics that admit non trivial models containing contradictory literals, say l and $\neg l$. This has been shown an important property for frameworks of knowledge and reasoning representation. The *well-founded semantics with explicit negation* $WFSX_P$, is a paraconsistent semantics for extended normal logic programs that envisages default negation and explicit negation necessarily related through the *coherence principle* [7]: if $\neg l$ holds, then *not* l should also hold (similarly, if l then *not* $\neg l$). In section 5 we define the *paraconsistent minimal hypotheses* semantics MH_P. As exposed there, the MH_P semantics endows $WFSX_P$ with choice mechanisms, in the fashion of MH, that allow reasoning by cases. MH_P models are total paraconsistent models, meaning that no MH_P model of an extended normal logic program contains undefined literals. MH_P inherits the advantages of the MH semantics, being existential, using loops as choice mechanisms and being *simple relevant* (see subsection 5.1). It also inherits the advantages of the $WFSX_P$ by not enforcing default consistency and hence producing models that allow to spot support on contradiction.

The rest of this paper proceeds as follows. In section 2 we define the language of extended normal logic programs and the terminology to be used in the sequel. For self-containment we present in sections 3 and 4 the definitions of the MH semantics and $WFSX_P$ semantics. In section 5 we exhibit in technical detail the definiton and characterization of the MH_P semantics. Section 6 is dedicated to conclusions and future work.

2 Language and Terminology of Logic Programs

An *extended normal logic program* is a finite set of ground rules, each one of the form $l_0 \leftarrow l_1, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n$, where l_i, $0 \leq i \leq n$, is an *objective literal* (either an atom b or its explicit negation, $\neg b$, where $\neg\neg b = b$); m, n are natural numbers, the operator ',' stands for the classical conjunctive connective and the operator *not* stands for default negation (*not* l is called a *default literal*). The set of all atoms that appear in an extended normal logic program is named the *Herbrand base* of P, denoted $\mathcal{H}_P$. If $m = n = 0$ a rule is called *fact*. If $\mathcal{H}_P$ contains no explicitly negated literals, the program is called *normal*; a rule with no explicitly negated literals is also called normal. Given a program P, program Q is a *subprogram* of P if $Q \subseteq P$, where Q and P are envisaged as sets of rules. Given a rule $r = l_0 \leftarrow l_1, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n$, the objective literal l_0 is the *head* of the rule and $l_1, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n$ is the *body* of the rule.[3] Some terminology used in the sequel is now established, concerning the dependencies among the elements (atoms and rules) of ground normal logic programs, triggered by the dependency operator '$\leftarrow$'.
Complete Rule Graph. (adapted from [4]) The *complete rule graph* of an extended normal logic program P, denoted by $CRG(P)$, is the directed graph whose vertices are the rules of P. Two vertices representing rules r_1 and r_2

[3] For ease of exposition, we henceforth use the following abbreviations: $Atoms(E)$, is the set of all *atoms* that appear in the ground structure E, where E can be a rule, a set of rules, a set of logic expressions, etc; $Bodies(E)$, is the set of all bodies that appear in the set of rules E; if E is unitary, we may use '*Body*' instead of '*Bodies*'; $Heads(E)$, is the set of all atoms that appear in the heads of the set of rules E; if E is unitary, we may use '*Head*' instead of '*Heads*'; $Facts(E)$, is the set of all facts that appear in the set of rules E; $Loops(E)$, is the set of all rules that appear involved in some loop contained in the set of rules E (see definition of *loop* in the sequel). We may compound some of these abbreviations, as for instance $Atoms(Bodies(P))$ whose meaning is immediate.

define an arc from r_1 to r_2, iff $Head(r_1) \in Atoms(Body(r_2))$, or $Head(r_1) = \neg Head(r_2)$, or $Head(r_2) = \neg Head(r_1)$.
Subprogram Relevant to an Objective Literal. (adapted from [4]) Let P be an extended normal logic program. We say that a rule $r \in P$ is *relevant* to an objective literal $l \in \mathcal{H}_P$, iff there is a rule s such that $Heads(s) = l$, and there is a direct path from r to s in the complete rule graph of P. In particular, rule s is relevant to l. The set of all rules of P relevant to l is represented by $Rel_P(l)$, and is named *subprogram relevant* to l.
Loop. (adapted from [16]) We say that a finite set of normal ground rules P forms a loop, iff P is of the form $\{h_1 \leftarrow l_2, B_1,\ h_2 \leftarrow l_3, B_2,\ \cdots,\ h_n \leftarrow l_1, B_n\}$, where $l_i = h_i$, or $l_i = not\ h_i$, and each B_i stands for a conjunction (possibly empty) of a finite number of literals. We say that each rule $h_i \leftarrow l_{i+1}, B_i$ is *involved* in the loop *through the literal* l_{i+1} or *through the atom* involved in the literal l_{i+1}, where $i+1$ is replaced by 1 if $i = n$.

Given a 3-valued interpretation I of an extended normal logic program, we represent by I^+ (resp. I^-) the set of its positive objective literals (resp. objective literals whose default negation is true with respect to I), and by I^u the set of undefined objective literals with respect to I. We represent I by the 3-tuple $I = \langle I^+, I^u, I^- \rangle$.[4]

The following operator shall be used in the sequel.

Definition 1. $\triangle$ operator. Given a normal logic program Q, we denote by $\triangle Q$ the 3-valued interpretation that can be read from Q in the following way: $b \in (\triangle Q)^+$ iff $(b \leftarrow) \in Q$; $b \in (\triangle Q)^u$ iff $(b \leftarrow) \notin Q$ and there is a rule r in Q such that $Head(r) = b$; $b \in (\triangle Q)^-$ iff b has no rule in Q.

3 The *MH* Semantics

In [17] the authors propose a reduction system comprised of the following five operators, each of which transforming normal logic programs into normal logic programs while keeping invariant the *well-founded model* [18] of the involved programs: *positive reduction*, $\mapsto_P$, *negative reduction*, $\mapsto_N$, *success*, $\mapsto_S$, *failure*, $\mapsto_F$, and *loop detection*, $\mapsto_L$ [5]. We here represent this reduction system by $\mapsto_{WFS} := \mapsto_P \cup \mapsto_N \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L$. Given a normal logic program P, the transformation $P \mapsto^*_{WFS} \widehat{P}$ [6] is such that $WFM(\widehat{P}) = WFM(P)$. Program $\widehat{P}$ will be here called the *WFS remainder* of P or *remainder* of P [7]. The system $\mapsto_{WFS}$ is both *terminating* and *confluent*, meaning that for any finite ground normal logic program the number of operations needed to reach $\widehat{P}$ is finite,

[4] We also write $b = +$, $b = u$, $b = -$, to mean respectively, $b \in I^+$, $b \in I^u$, $b \in I^-$.
[5] See INAP 11 paper [1], defs. 8, 9, 11, 12, 13, for definitions of these operators.
[6] Where $\mapsto^*_{WFS}$ means the nondeterministic performing of operations of the system $\mapsto_{WFS}$, until the resulting program becomes invariant.
[7] See INAP 11 paper [1], def. 16.

and the order in which the operations are executed is irrelevant, as long as the preconditions for the application of each one of them are verified. To compute the MH semantics of a normal logic program, we need the variant $\mapsto_{LWFS}$ of the reduction system $\mapsto_{WFS}$, which results from $\mapsto_{WFS}$ by substituting the negative reduction operator, $\mapsto_N$, by the *layered negative reduction* operator, $\mapsto_{LN}$ – see [1], def. 10. Given a normal logic program P, the transformation $P \mapsto^*_{LWFS} \mathring{P}$ [8] is such that $\triangle\mathring{P} = LWFS(P)$, where $LWFS(P)$ [4] stands for the *layered well-founded model* of P – the transformed program $\mathring{P}$ is called *$LWFS$ remainder* or *layered remainder* of P. The system $\mapsto_{LWFS}$ is terminating and confluent when applied to finite ground normal logic programs.

Example 3. The layered remainder of program P (left column) is program $\mathring{P}$ (center column), and the remainder of P is program $\widehat{P}$ (right column) – rules and literals stripped out, are eliminated during $\mathring{P}$ and $\widehat{P}$ computations.[9]

$b \leftarrow h$	$b \leftarrow h$	$b \leftarrow$ ~~h~~
$h \leftarrow not\ p, b$	$h \leftarrow not\ p,$ ~~b~~	$h \leftarrow$ ~~$not\ p, b$~~
$p \leftarrow not\ b$	$p \leftarrow not\ b$	~~$p \leftarrow not\ b$~~
$a \leftarrow not\ c, b$	$a \leftarrow$ ~~$not\ c, b$~~	$a \leftarrow$ ~~$not\ c, b$~~
$d \leftarrow not\ b$	~~$d \leftarrow not\ b$~~	~~$d \leftarrow not\ b$~~
$b \leftarrow$	$b \leftarrow$	$b \leftarrow$

The layered well-founded model of P is thus $LWFM(P) = \triangle\mathring{P} = \langle\{a, b\}^+, \{h, p\}^u, \{c, d\}^-\rangle$, and the well-founded model of P is $WFM(P) = \triangle\widehat{P} = \langle\{a, b, h\}^+, \{\}^u, \{c, d, p\}^-\rangle$.

MH being an abductive semantics, we shall now define the *assumable hypotheses set* and the *minimal hypotheses* model of a normal logic program.

Definition 2. Assumable Hypotheses Set of a Program. (adapted from [1]) Let P be a finite ground normal logic program. We write $Hyps(P)$ to denote the *assumable hypotheses set* of P: those atoms that appear default negated in the bodies of rules of $\mathring{P}$ and which are not facts in $\mathring{P}$, i.e. they belong to $(\triangle\mathring{P})^u$.

[8] $\mapsto^*_{LWFS}$ means the nondeterministic performing of operations of the system $\mapsto_{LWFS}$ until the resulting program becomes invariant.

[9] Rule $d \leftarrow not\ b$ in P is eliminated, both in $\mathring{P}$ and $\widehat{P}$, by layered negative reduction in the first case and by negative redution in the second – the two operations coincide here, since the rule is not in loop via literal *not* b; the body of rule $a \leftarrow not\ c, b$ in P, becomes empty by success (which eliminates b) and by positive reduction (which eliminates *not* c), both in $\mathring{P}$ and $\widehat{P}$; rule $p \leftarrow not\ b$ in is eliminated in $\widehat{P}$ by negative reduction, but not in $\mathring{P}$, since the rule is in loop through literal *not* b and layered negative redution does nothing is such cases; b is eliminated from the body of rule $h \leftarrow not\ p, b$ by success, both in $\mathring{P}$ and $\widehat{P}$, whilst positive reduction eliminates also the literal *not* p in $\widehat{P}$, because p is an atom without rule due to the elimination of rule $p \leftarrow not\ b$, turning h into a fact; h is eliminated from the body of rule $b \leftarrow h$, by success, in $\widehat{P}$.

The purpose of computing $\mathring{P}$ is to find the set of assumable hypotheses of P, which are then used to compute the minimal hypoteses models of the program.

Definition 3. Minimal Hypotheses Model. (adapted from [1]) Let P be a finite ground normal logic program. Let $Hyps(P)$ be the assumable hypotheses set of P (cf. def. 2), and H a subset of $Hyps(P)$. A 2-valued interpretation M of P is a *minimal hypotheses* model of P, iff $WFM^u(P \cup H) = \emptyset$, where $H = \emptyset$ or H is a nonempty set that is minimal with respect to set inclusion (set inclusion minimality disregards any empty H). I.e. the hypotheses set H is minimal but sufficient to determine (via the well-founded model) the truth-value of all literals in the program.

Theorem 1. Every normal logic program has at least one MH model.

Every stable model of a normal logic program is also a minimal hypotheses model of the program. This justifies the catering for whole models with empty hypotheses set, which are stable models of programs whose layered remainders are stratified programs. The reason hypotheses minimization does not contemplate empty hypotheses set models, is to allow loops to be taken as choice devices, also in these cases. For instance, program P in example 3 has the assumable hypotheses set $Hyps(P) = \{p\}$, since $not\ p$ appears in $\mathring{P}$ and p is not a fact of this program [10]. The MH models of P are $\{a, b, not\ c, h, not\ p\}$ with hypotheses set $\emptyset$, and $\{a, b, not\ c, not\ h, p\}$ with hypotheses set $\{p\}$. If the empty hypotheses set were allowed in the hypotheses minimization, the non-empty hypotheses set model would be discarded and we would be left with just the stable model $\{a, b, not\ c, h, not\ p\}$.

Example 4. Consider the following variation of the *vacation problem*[11] [1], $P = \{a \leftarrow not\ b,\ b \leftarrow not\ a, not\ c,\ c \leftarrow not\ d,\ d \leftarrow not\ e, not\ a,\ e \leftarrow not\ a, not\ c\}$, where $P = \mathring{P}$. The hypotheses set of P is $Hyps(P) = \{a, b, c, d, e\}$. The MH models of P are: $\{a, not\ b, c, not\ d, not\ e\}$ with hypotheses set $\{a\}$, $\{not\ a, b, not\ c, d, e\}$ with hypotheses set $\{b, d\}$[12] and $\{a, not\ b, c, not\ d, e\}$ with hypotheses set $\{e\}$.

This example shows this type of problem is not solvable by resorting to answer sets semantics [19], if we stick to the set of rules of P, since models may not be minimal (e.g. $\{a, not\ b, c, not\ d, e\}$ above). Should there be a transformation on normal logic programs, let it be $\mapsto_Y$, such that $P \mapsto_Y P^*$, where the MH models of P could be extracted from the stable models of P^*, then P^* would have a different set of rules and/or a different language, with respect to P, which

[10] Notice that although $not\ b$ appears in $\mathring{P}$, b is not an assumable hypothesis of P since it is a fact of $\mathring{P}$.

[11] Five friends are planning a joint vacation. First friend says "If we don't go to place b, then we should go to place a", which corresponds to rule $a \leftarrow not\ b$; the same rationale for the remaining rules.

[12] There are no MH models with hypotheses sets $H = \{b\}$ or $H = \{d\}$, since in these cases $WFM^u(P \cup H) \neq \emptyset$.

means that this type of problem is specified in a more elegant way if the solution is to be obtained via the MH semantics. Yet, it is an open problem whether such a transformation exists. [13]

4 The $\mathbf{WFSX_P}$ Semantics

The $WFSX_P$ model of an extended normal logic program may be computed by means of a dedicated fixpoint operator [2]. In this section, however, we instead present a definition of the $WFSX_P$ by means of a program transformation for extended normal logic programs, dubbed $t-o$ *transformation*[14] [20], which embeds the $WFSX_P$ into the WFS. This means that the $WFSX_P$ model of an extended normal logic program P, denoted by $WFM_P(P)$, may be extracted from the well-founded model of the transformed program P^{t-o}.

Definition 4. $\mathbf{t-o}$ Transformation. (adapted from [20]) The $t-o$ *transformation*, maps an extended normal logic program P into a normal logic program P^{t-o}, by means of the two following steps:
1. Every explicitly negated literal in P, say $\neg b$, appears also in the transformed program, where it must be read as a new atom, say $\neg\_b$. Let P^* be the program resulting from making these transformations on P.
2. Every rule $r = (Head(r) \leftarrow Body(r))$ in P^* is substituted by the following pair of rules: (i) A rule also designated r, for simplicity, obtained from $r \in P^*$ by placing the superscript 'o' in the default negated atoms of $Body(r)$; (ii) A rule r^o, obtained from $r \in P^*$ by adding to $Body(r)$ the literal $not\ \neg Head(r)$ (where $\neg\neg l = l$)[15], and by placing the superscript 'o' in $Head(r)$ and in every positive literal of $Body(r)$.
We call *co-rules* to each pair r, r^o of rules in P^{t-o} (each rule is the *co-rule* of the other one), and *co-atoms* to each pair b, b^o of atoms in P^{t-o} (each atom is the *co-atom* of the other one). To each objective literal l of the language of P^*, there corresponds the pair of co-atoms l, l^o of the language of P^{t-o} and vice-versa.[16]

[13] The solution proposed by MH semantics for the vacation problem has the following rationale: rule $a \leftarrow not\ b$, for example, states that the first friend prefers place b to place a, because a is suggested in case b fails; each model of MH tries to satisfy the first options of the friends, by considering them as hypotheses. The answer set solution to this type of problem, when it exists, retrives models that statisfy all the friends' demands (rules) with the smallest (with respect to set inclusion) possible number of places to visit (due to the minimality of models).

[14] The original designation is $T-TU$ *transformation*. Our definition alters the notation, for simplicity purposes, while keeping the meaning of the original one.

[15] The literal $not\ \neg Head(r)$ is added to enforce the coherence principle.

[16] The rules with superscript 'o' in the heads are used to derive the literals that are true or undefined in the $WFSXp$ model, and the rules with no superscript in the head are used to derive the true literals of the $WFSXp$ model.

Example 5. (adapted from [20]) Program P (left column) has the $t-o$ transformed P^{t-o} (two columns on the right):

$a \leftarrow$	$a \leftarrow$	$a^o \leftarrow not\ \neg a$
$\neg a \leftarrow$	$\neg a \leftarrow$	$\neg a^o \leftarrow not\ a$
$b \leftarrow a$	$b \leftarrow a$	$b^o \leftarrow a^o, not\ \neg b$
$c \leftarrow not\ b$	$c \leftarrow not\ b^o$	$c^o \leftarrow not\ b, not\ \neg c$
$d \leftarrow not\ d$	$d \leftarrow not\ d^o$	$d^o \leftarrow not\ d, not\ \neg d$

The following theorem states how to read the $WFM_P(P)$ model from the $WFM(P^{t-o})$, where P is any extended normal logic program.

Theorem 2. WFSX$_\mathbf{P}$ is embeddable into WFS. (adapted from [20]) Let P be an extended normal logic program. Then the following equivalences hold for an arbitrary atom b of the language of P: $b \in WFMp(P)$, iff $b \in WFM(P^{t-o})$; $not\ b \in WFMp(P)$, iff $not\ b^o \in WFM(P^{t-o})$; $\neg b \in WFMp(P)$, iff $\neg b \in WFM(P^{t-o})$; $not\ \neg b \in WFMp(P)$, iff $not\ \neg b^o \in WFM(P^{t-o})$, where the symbols $\neg b, \neg b^o$ on the right sides of the 'iffs' must be taken as names of atoms (they are not explicitly negated literals), in accordance with definition 4.

Definition 5. $\bigtriangledown$ operator. Given a 3-valued interpretation $I = \langle I^+, I^u, I^- \rangle$, where I^+, I^u, I^- may contain 'o' superscript or otherwise nonsuperscript atoms, we denote by $\bigtriangledown I$ the 3-valued interpretation obtained by means of the lexical correspondences stated in theorem 2.

Using this operator we may write, for example, $WFM_P(P) = \bigtriangledown WFM(P^{t-o})$. The next proposition and corollary characterize the valuations of the pairs of co-atoms b, b^o of the language of P^{t-o}, with respect to the $WFM(P^{t-o})$: it is the case that $b^o \leq_t b$ for any such pair (b^o, b standing here for their valuations with respect to the $WFM(P^{t-o})$), where $\leq_t$ is the truth ordering[17].

Proposition 1. Let P^{t-o} be the $t-o$ transformed of a finite ground extended normal logic program P. Let $\Gamma^n_{P^{t-o}}(\emptyset)$ be the result of the n-th self composition of the Gelfond-Lifschitz Γ operator [21] on program P^{t-o}, with argument $\emptyset$. Then, for every $n \in \mathbb{N}$ and for every atom b of the Herbrand base of P we have $b^o \in \Gamma^n_{P^{t-o}}(\emptyset) \Rightarrow b \in \Gamma^n_{P^{t-o}}(\emptyset)$.

Corollary 1. (of proposition 1) Let b, b^o be two co-atoms of the Herbrand base of P^{t-o}. Then it is not possible to have any of the following three types of valuations with respect to the $WFM(P^{t-o})$: $(b, b^o) = (-, +)$, $(b, b^o) = (-, u)$, $(b, b^o) = (u, +)$. The only possible valuations are $(b, b^o) = (-, -)$, $(b, b^o) = (u, -)$, $(b, b^o) = (+, -)$, $(b, b^o) = (u, u)$, $(b, b^o) = (+, u)$, $(b, b^o) = (+, +)$.

[17] Given the logic values f (false), u (undefined) and t (true), their *truth ordering* is defined by: $f \leq_t u \leq_t t$, [11].

The $WFM(P^{t-o})$ in example 5 is
$\langle \{a, \neg a, b, c\}^+, \{d, d^o\}^u, \{a^o, \neg a^o, b^o, \neg b, \neg b^o, c^o, \neg c, \neg c^o, \neg d, \neg d^o\}^- \rangle$. Using theorem 2 we obtain the $WFSX_P$ model $\langle \{a, \neg a, b, c\}^+, \{d\}^u, \{a, \neg a, b, \neg b, c, \neg c, \neg d\}^- \rangle$. To get the meaning of this model, notice that $WFSX_P$ does not enforce default consistency, i.e. l and *not* l can be simultaneously true in a paraconsistent model, in contradistinction to all other paraconsistent semantics [20], which allows to detect dependence on contradictory information. This is a consequence of adopting the *coherence principle* [22]. In the above example, for any $l \in \{a, \neg a, b, c\}$, both l and *not* l belong to the $WFM_P(P)$, thus revealing the valuations of $\{a, b, c\}$ as inconsistent (the valuation of a is also contradictory with respect to explicit negation, since a and $\neg a$ are both true; is due to this contradictory valuation that the inconsistency of a, b, c occurs). The valuation of atom d is in turn consistent. Table 1 (see appendix) presents in column $NINE$[18] the correspondence between each possible 4-tuple of $WFSX_P$ valuations of a literal $(l^o, l, \neg l^o, \neg l)$ and the nine possible logic values literal l may assume with respect to logic $NINE$. According to table 1, the logic values of the atoms in the $NINE$ model corresponding to the $WFM_P(P)$ above, are as follow: a is contradictory true (logic value I); b, c are true with contradictory belief (logic value II); d is default true (logic value dt).

Definition 6. Total/Partial Paraconsistent Model. (adapted from [23]) Let P be a finite ground extended normal logic program and SEM a semantics for extended normal logic programs. A SEM model $M = \langle M^+, M^u, M^- \rangle$ of P is a *total paraconsistent* model, iff $M^u = \emptyset$. M is called a *partial paraconsistent model*, iff $M^u \neq \emptyset$.

Proposition 2. The $WFSX_P$ semantics is a partial paraconsistent models semantics, meaning that the $WFSX_P$ models of some extended normal logic programs contain undefined literals.

Theorem 3. (adapted from [2]) If P is a normal logic program, then the models $WFM(P)$ and $WFM_P(P)$ are equal, if we neglect the explicitly negated literals in $WFM_P(P)$.

5 The MH_P Semantics

The MH_P semantics of an extended normal logic program P is computed via the following steps: 1) Compute the *balanced layered remainder* bP^{t-o} of P (see definition 9); 2) Compute the *assumable hypotheses set* of P (see definition 10); 3) Compute the MH_P models of P (see definition 11).

The balanced layered remainder bP^{t-o} of an extended normal logic program P, is the outcame of the *balanced layered reduction* transformation of P, $P \mapsto^*_{bLWFM}$

[18] A logic dubbed $NINE$ [20], presented in the appendix, provides a truth-functional model theory for the $WFSXp$ semantics.

bP^{t-o}, where (i) $\mapsto_{bLWFM}$ is obtained from $\mapsto_{LWFM}$ replacing the layered negative reduction operator, $\mapsto_{LN}$, by the *balanced layered negative reduction* operator, $\mapsto_{bLN}$, defined below; (ii) $\mapsto^*_{bLWFM}$ means the nondeterministic performing of operations of the system $\mapsto_{bLWFM}$ until the resulting program becomes invariant under any further operation.

Definition 7. Balanced[19] Layered Negative Reduction. Let P_1 and P_2 be two ground normal logic programs, whose Herbrand bases contain 'o' superscript and nonsuperscript atoms. We say that P_2 results from P_1 by a *balanced layered negative reduction* operation, $P_1 \mapsto_{bLN} P_2$, iff one of the next two cases occur: (1) There is a fact b^o in P_1 and a rule r in P_1 whose body contains the literal $not\ b^o$, where neither r is involved in a loop through the literal $not\ b^o$ nor is its co-rule r^o involved in a loop through the literal $not\ b$, and $P_2 = P_1 \setminus \{r\}$; (2) There is a fact b in P_1 and a rule r^o in P_1 whose body contains the literal $not\ b$, where neither r^o is involved in a loop through the literal $not\ b$, nor is its co-rule r involved in a loop through the literal $not\ b^o$, and $P_2 = P_1 \setminus \{r^o\}$.[20]

Definition 8. Balanced Layered Reduction. The *balanced layered reduction* system is the system $\mapsto_{bLWFM} := \mapsto_P \cup \mapsto_{bLN} \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L$.

Theorem 4. Termination and Confluency. The system $\mapsto_{bLWFM}$, when applied to finite ground programs, is both terminating and confluent.

Definition 9. Balanced Layered Remainder. Let P be a finite ground extended normal logic program and P^{t-o} its $t-o$ transformed. We call *balanced layered remainder* of P to the program bP^{t-o} such that $P^{t-o} \mapsto^*_{bLWFM} bP^{t-o}$.

Example 6. The balanced layered remainder of the program P (left column) is bP^{t-o} (two columns on the right) – rules and literals stripped out, are eliminated during bP^{t-o} computation.[21]

$b \leftarrow h$	$b \leftarrow h$	~~$b^o \leftarrow h^o, not\ \neg b$~~
$h \leftarrow not\ p$	$h \leftarrow not\ p^o$	~~$h^o \leftarrow not\ p, not\ \neg h$~~
$p \leftarrow not\ b$	$p \leftarrow not\ b^o$	$p^o \leftarrow not\ b,$ ~~$not\ \neg p$~~
$b \leftarrow$	$b \leftarrow$	$b^o \leftarrow$ ~~$not\ \neg b$~~
$\neg h \leftarrow$	$\neg h \leftarrow$	$\neg h^o \leftarrow not\ h$

[19] The expression 'balanced' refers to the consideration of pairs of co-rules in this definition.

[20] This operation is weaker than layered negative reduction, meaning that where the former is applicable so is the latter.

[21] Notice that rule $h^o \leftarrow not\ p, not\ \neg h$ is eliminated by balanced negative reduction, because $\neg h$ is a fact of bP^{t-o}, and neither is the rule involved in a loop through literal $\neg h$, nor is its co-rule $h \leftarrow not\ p^o$ involved in a loop through literal $\neg h^o$ – balanced layered negative reduction has, in this case, the same effect as negative reduction; rule $b^o \leftarrow h^o, not\ \neg b$ is eliminated by failure, because h^o does not have a rule after elimination of $h^o \leftarrow not\ p, not\ \neg h$; literals $not\ \neg b, not\ \neg p$ are eliminated by positive reduction, since $\neg b, \neg p$ do not have a rule in bP^{t-o}.

Proposition 3. Let P be an extended normal logic program and $M = \triangle bP^{t-o}$. Then the valuation with respect to M of any pair of co-atoms, say l, l^o, of the Herbrand base of P^{t-o}, agrees with the statement of corollary 1.

Definition 10. Assumable Hypotheses Set of a Program. Let P be a finite ground extended normal logic program and bP^{t-o} its balanced layered remainder. We say that $Hyps(P) \subseteq \mathcal{H}_P$ is the *assumable hypotheses set* of P, iff for all $h \in Hyps$ it is the case that the default literal *not* h^o appears in program bP^{t-o}, and h is not a fact in bP^{t-o}, i.e. $h \in (\triangle bP^{t-o})^u$.[22]

Definition 11. Paraconsistent Minimal Hypotheses Semantics, $\mathbf{MH_P}$. Let P be a finite ground extended normal logic program, bP^{t-o} its balanced layered remainder and $Hyps$ the set of assumable hypotheses of P. Then the *paraconsistent minimal hypotheses* semantics of P, denoted $MHp(P)$, is defined by the paraconsistent minimal hypotheses models M of P, which are computed as follows:
1. $M = WFM_P(P \cup H) = \bigtriangledown WFM(P^{t-o} \cup H \cup \{h^o \leftarrow not\ \neg h : h \in H\})$,
   for all $H \subseteq Hyps$, $H \neq \emptyset$, H is inclusion-minimal and $WFM^u_P(P \cup H) = \emptyset$.
2. $M = WFM_P(P) = \bigtriangledown WFM(P^{t-o})$, where $WFM^u_P(P) = \emptyset$.

Each hypothesis, say h, is added to bP^{t-o} as a pair of rules $\{h \leftarrow,\ h^o \leftarrow not\ \neg h\}$. No MH_P model of an extended normal logic program has undefined literals. Table 1 (see appendix) presents in column SIX[23] the correspondence between each possible 4-tuple of MH_P valuations of a literal $(l^o, l, \neg l^o, \neg l)$ and the six possible logic values literal l may assume with respect to SIX.

Example 7. Consider the program in example 6. The assumable hypotheses set of P is $\{p\}$, since although both *not* b^o, *not* p^o appear in bP^{t-o}, p is not a fact in the program bP^{t-o}, whilst b is. The MH_P models are:[24]

$$\begin{aligned}
M_1 &= \bigtriangledown WFM(P^{t-o}) \\
&= \bigtriangledown\langle \{b, b^o, h, \neg h\}^+, \{\}^u, \{\neg b, \neg b^o, h^o, \neg h^o, p, p^o, \neg p, \neg p^o\}^- \rangle \\
&= \langle \{b, h, \neg h\}^+, \{\}^u, \{\neg b, h, \neg h, p, \neg p\}^- \rangle,\ \text{with hypotheses set } \emptyset \\
M_2 &= \bigtriangledown WFM(P^{t-o} \cup \{p\} \cup \{p^o \leftarrow not\ \neg p\}) \\
&= \bigtriangledown\langle \{b, b^o, \neg h, \neg h^o, p, p^o\}^+, \{\}^u, \{\neg b, \neg b^o, h, h^o, \neg p, \neg p^o, \}^- \rangle \\
&= \langle \{b, \neg h, p\}^+, \{\}^u, \{\neg b, h, \neg p\}^- \rangle,\ \text{with hypotheses set } \{p\}
\end{aligned}$$

[22] This is equivalent to saying that $h \in (\bigtriangledown \triangle\ bP^{t-o})^u$. Notice that the purpose of computing bP^{t-o}, is to find the set of assumable hypotheses of P.

[23] Logic SIX provides a truth-functional model theory for the MH_P semantics.

[24] According to column SIX of table 1, the interpretations of the valuations of the literals in the two MH_P models above, are as follows: with respect to model M_1, b is true (logic value t), h is contradictory true (logic value I), p has contradictory belief (logic value IV); with respect to model M_2, b, p are true (logic value t) and h is false (logic value f).

We see that the first model is inconsistent while the second is not. The first model coincides with the $WFM_P(P)$. The second model arises because the MH_P uses the loop $\{b \leftarrow h,\ h \leftarrow \mathit{not}\ p,\ p \leftarrow \mathit{not}\ b\}$ of program P as a choice device.

The following results show that MH_P is a *total paraconsistent models* semantics (cf. def. 6) and that MH and MH_P coincide for normal logic programs, if we discard the default literals involving explicitly negated atoms from the MH_P models.

Proposition 4. The MHp semantics is a total paraconsistent models semantics, meaning that for any extended normal logic program P all the MH_P models of P are total paraconsistent models.

Theorem 5. MHp and MH coincide on Normal Logic Programs. Let P be a normal logic program and P^{t-o} the $t-o$ transformed of P. Let $MH(P)$ be the set of minimal hypotheses models of P and $MH_P(P)$ the set of paraconsistent minimal hypotheses models of P. Then $M \in MH(P)$, iff there is a model $M_p \in MH_P(P)$, such that $M^+ = M_p^+$ and $M^- = (M_p^- \cap \mathcal{H}_p)$, where $\mathcal{H}_p$ is the Herbrand base of P.

5.1 Formal Properties

The MH_P semantics enjoys the properties of *existence* and *simple relevance*[25].

5.2 Complexity

[26]The theorem below shows that a *brave reasoning* task with MH_P semantics, i.e. finding an MH_P model satisfying some particular set of literals (a query), is in Σ_2^P.

Theorem 6. Brave reasoning with MH_P semantics is in Σ_2^P.

Proof. Let us show that finding a MH_P model of an extended normal logic program P is in Σ_2^P. Computing P^{t-o} is fulfilled in linear time. The balanced layered remainder bP^{t-o} is computed in polynomial time, by the following reasoning: the calculus of the remainder of a normal logic program is kown to be of polynomial time complexity [17]; the difference between $\mapsto_{WFS}$ and $\mapsto_{bLWFS}$ lies on the operator $\mapsto_N$ of the former being replaced by the operator $\mapsto_{bLN}$ of the latter; to perform $\mapsto_{bLN}$ the rule layering must be computed; the rule layering can be calculated in polynomial time since it is equivalent to identifying the *strongly connected components* [24] in a graph, $SCCs$, in this case in the complete rule graph of P^{t-o}; once the $SCCs$ are found, one collects their heads in sets, one set for each SCC, and searches the bodies of each SCC rules for

[25] We say that SEM has the property of *simple relevance*, iff for any extended normal logic program P, whenever there is a SEM model of $Rel_P(l)$ such that $l \in M_l$, there is also a SEM model M of P such that $l \in M$.

[26] This subsection follows closely subsection 4.6 of [1].

default literals, collecting in a set per SCC the corresponding atoms that also pertain to the set of heads of that SCC – this is all linear time; when verifying the preconditions to perform a negative reduction operation (existence of a fact, say b, and a rule with *not* b in the body), it is linear time to check if a rule is in loop (check if it belongs to a SCC) and if it is in loop through literal *not* b (check if b belongs to the heads of the SCC) – the same for cheking if the co-rule is in loop through *not* b^o; therefore, balanced layered negative reduction adds only polynomial time complexity operations over negative reduction. Once bP^{t-o} is computed, nondeterministically guess a set H of hypotheses – the assumable hypotheses set, is the set of all atoms involved in default negations in bP^{t-o}, that are not facts. Check if $WFM_P^u(P \cup H) = \emptyset$ – this is polynomial time. Checking that H is empty or non-empty minimal, requires another nondeterministic guess of a strict subset H' of H and then a polynomial check if $WFM_P^u(P \cup H') = \emptyset$.□

The theorem below shows that a *cautious reasoning* task with MH_P semantics, i.e. guaranteeing that every MH_P model satisfies some particular set of literals (a query), is in Π_2^P.

Theorem 7. Cautious reasoning with MH_P semantics is in Π_2^P.

Proof. Cautious reasoning is the complement of brave reasoning, and since the latter is in Σ_2^P the former must necessarily be in Π_2^P. □

6 Conclusions and Future Work

We have presented an abductive paraconsistent semantics, MH_P, that inherits the advantages of the abductive semantics MH, which renders the MH_P existential and simple relevant, and of the paraconsistent semantics $WFSX_P$, due to which the MH_P is endowed with the ability to detect support based on contradiction through non-trivial inconsistent models. MH_P explores a new way to envisage logic programming semantics through abduction, by dealing with paraconsistency using total paraconsistent models. The MH_P semantics may be used to perform reasoning in the following ways (let Q be a query and P an extended normal logic program in the role of a database): **Skeptical Consistent Reasoning:** Query Q succeeds, iff it succeeds for all consistent MH_P models of P (it fails if there are no consistent models); **Brave Consistent Reasoning:** Query Q succeeds, iff it succeeds for at least one non-contradictory MH_P model of P; **Skeptical Paraconsistent Reasoning:** Query Q succeeds, iff it succeeds for all MH_P models of P, such that none of the atoms in $Atoms(Q)$ has support on contradiction for any of these models; (it fails if there are no such models); **Brave Paraconsistent Reasoning:** Query Q succeeds, iff it succeeds for at least one MH_P model of P, such that none of the atoms in $Atoms(Q)$ has support on contradiction in this model; **Skeptical Liberal Reasoning:** Query Q succeeds, iff it succeeds for all MH_P models of P; **Brave Liberal Reasoning:** Query Q succeeds, iff it succeeds for at least one MH_P model of P. As future work, the MH_P may be extended in order to obtain a framework for representing and integrating knowledge updates from external sources and also inner source knowledge updates (or self updates), in line with the proposal in [25].

Acknowledgments. We are grateful to three anonymous referees whose observations were useful to improve the paper.

References

1. Pinto, A.M., Pereira, L.M.: Each normal logic program has a 2-valued minimal hypotheses semantics. INAP 2011, CoRR **abs/1108.5766** (2011)
2. Alferes, J.J., Damásio, C.V., Pereira, L.M.: A logic programming system for nonmonotonic reasoning. J. Autom. Reasoning **14**(1) (1995) 93–147
3. Denecker, M., Kakas, A.C.: Abduction in logic programming. In: Computational Logic: Logic Programming and Beyond'02. (2002) 402–436
4. Pinto, A.M.: Every normal logic program has a 2-valued semantics: theory, extensions, applications, implementations. PhD thesis, Universidade Nova de Lisboa (July 2011) Published in the December issue of the ALP newsletter:http://www.cs.nmsu.edu/ALP/2012/12/doctoral-dissertation-every-normal-logic-program-has-a-2-valued-semantics-theory-extensions-applications-implementations/.
5. Gelfond, M., Lifschitz, V.: Logic programs with classical negation. In: ICLP. (1990) 579–597
6. Kowalski, R.A., Sadri, F.: Logic programs with exceptions. In: ICLP'90. (1990) 598–613
7. Pereira, L.M., Alferes, J.J.: Well founded semantics for logic programs with explicit negation. In: ECAI'92. (1992) 102–106
8. Przymusinski, T.: Well-founded Semantics Coincides With Three-Valued Stable Semantics. Fundamenta Informaticae **XIII** (1990) 445–463
9. Przymusinski, T.C.: Extended stable semantics for normal and disjunctive programs. In: ICLP'90. (1990) 459–477
10. Wagner, G.: A database needs two kinds of negation. In: MFDBS. (1991) 357–371
11. Alferes, J.J., Pereira, L.M.: Reasoning with Logic Programming. Volume 1111 of Lecture Notes in Computer Science. Springer (1996)
12. Arieli, O.: Paraconsistent declarative semantics for extended logic programs. Ann. Math. Artif. Intell **36**(4) (2002) 381–417
13. Blair, H.A., Subrahmanian, V.S.: Paraconsistent logic programming. Theor. Comput. Sci. (1989) 135–154
14. Pearce, D., Wagner, G.: Logic programming with strong negation. In Schroeder-Heister, P., ed.: ELP. Volume 475 of Lecture Notes in Computer Science., Springer (1989) 311–326
15. Sakama, C.: Extended well-founded semantics for paraconsistent logic programs. In: FGCS. (1992) 592–599
16. Costantini, S.: Contributions to the stable model semantics of logic programs with negation. Theoretical Computer Science **149**(2) (2 October 1995) 231–255
17. Brass, S., Dix, J., Freitag, B., Zukowski, U.: Transformation-based bottom-up computation of the well-founded model. TPLP (2001) 497–538
18. Gelder, A.V.: The alternating fixpoint of logic programs with negation. J. of Comp. System Sciences **47**(1) (1993) 185–221

19. Lifschitz, V.: What is answer set programming? In: AAAI'08. (2008) 1594–1597
20. Damásio, C.V., Pereira, L.M.: A survey of paraconsistent semantics for logic programs. Handbook of defeasible reasoning and uncertainty management systems: volume 2: reasoning with actual and potential contradictions (1998) 241–320
21. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, MIT Press (1988) 1070–1080
22. Alferes, J.J., Damásio, C.V., Pereira, L.M.: Top-down query evaluation for well-founded semantics with explicit negation. In: ECAI'94. (1994) 140–144
23. Saccà, D., Zaniolo, C.: Partial models and three-valued models in logic programs with negation. In: LPNMR'91. (1991) 87–101
24. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. (1972) 146–160
25. Alferes, J.J., Brogi, A., Leite, J.A., Pereira, L.M.: Evolving logic programs. In: JELIA'02. (2002) 50–61

Appendix

Logics *NINE* and *SIX*

In [20] the author presents a logic, there dubbed $NINE$, that provides a truth-functional model theory for the $WFSXp$, based on Ginsberg's bilattices concept. The truth-space of $NINE$ comprises nine logic values, here presented together with their meanings: '**t**' and '**f**' are the classical values for truth and falsity; '**I**', is the contradictory truth value; '**II**', is understood as truth with contradictory belief; '**III**', is understood as falsity with contradictory belief; '**IV**', is understood as contradictory belief; '$\perp$', is understood as undefinedness; '**df**', is understood as default falsity; '**dt**', is understood as default truth. Using the ideas seth forth in the definition of $NINE$ we define SIX, a truth-functional model theory for the MHp, whose truth-space is $\{t, f, I, II, III, IV\}$. Table 1 represents all the possible 4-tuple valuations of a literal $(l^o, l, \neg l^o, \neg l)$ with the corresponding $NINE$ and SIX logic values. There are 20 possible 4-tuple literal valuations, as a consequence of corollary 1 and the coherence principle.

Table 1. NINE and SIX valuations

l^o	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	u	u	u	$-$	$-$	$-$	u	u	$+$
l	$-$	$-$	$-$	$-$	$-$	$-$	u	u	u	u	u	u	u	u	$+$	$+$	$+$	$+$	$+$	$+$
$\neg l^0$	$-$	$-$	u	$-$	u	$+$	$-$	$-$	u	$-$	u	$-$	$-$	u	$-$	$-$	$-$	$-$	$-$	$-$
$\neg l$	$-$	u	u	$+$	$+$	$+$	$-$	u	u	$+$	$+$	$-$	u	u	$-$	u	$+$	$-$	u	$-$
$NINE$	IV	IV	df	III	f	f	IV	IV	df	III	f	dt	dt	$\perp$	II	II	I	t	t	t
SIX	IV	IV		III	f	f	IV	IV		III	f				II	II	I	t	t	t