The Abductive Paraconsistent Semantics MHp

Mário Abrantes¹* and Luís Moniz Pereira²

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 abductive semantics for ELP, the paraconsistent minimal hypotheses semantics MHp. The MHp is a semantics of total paraconsistent models which combines the merits of two already existing semantics: it inherits the existence property of the abductive minimal hypotheses semantics MH[, 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 WFSXp, which is a semantics of partial paraconsistent models. The MHp enjoys also the property of simple relevance, which permits top-down query answering for brave reasoning purposes. Besides, the MHp lends itself to various types of skeptical and brave reasoning, which include the possibility of drawing conclusions from inconsistent nontrivial models. The MHp coincides with the MH on normal logic programs, and with the WFSXp 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 paraconsistent minimal hypotheses semantics MHp. Extended logic programs [?] are logic programs whose language contains two types of negation operators: the explicit negation operator '¬', for representing contradictory knowledge, in addition to the negation-as-failure (or default negation) operator 'not', used for representing incomplete information. The MHp semantics combines the merits of two already existing semantics: the abductive minimal hypotheses semantics MH [?] for normal logic programs, and the

* Partially supported by Fundação para a Ciência e Tecnologia and Instituto Politécnico de Bragança grant PROTEC : SFHR/49747/2009.
paraconsistent well-founded semantics with explicit negation $WFSX_p$ [?] for extended logic programs.

From the $MH$ semantics (see section ??), the $MH_p$ inherits the property of existence, which guarantees every extended logic program has at least one $MH_p$ model. The $MH$ semantics is a conservative extension of the stable model semantics $SM$ [?], i.e., the $MH$ semantics retrieves a superset of the set of stable models, for every normal logic program $P$. As a consequence of the existence property, the $MH$ enjoys the simple relevance property (see proposition ??), which the $SM$ does not. This property allows query answering by resorting only to the subset of rules that are relevant to the literals that appear in the query. The $MH_p$ inherits also this property from the $MH$.

From the $WFSX_p$ semantics, the $MH_p$ inherits the capability of support on contradiction detection. Contradictions may arise in semantics for extended logic programs, since explicitly negated literals $\neg L$ may be asserted in the models. $WFSX_p$ is a paraconsistent semantics. Paraconsistent semantics [?, ?, ?, ?, ?], are semantics that admit non trivial inconsistent models, i.e., models containing $L$ and $\neg L$ for some non default negated literal $L$, and at same time not containing both $K$ and $\neg K$, for some other non default negated literal $K$. This has been shown an important feature of frameworks for knowledge and reasoning representation. The $WFSX_p$ envisages default negation and explicit negation necessarily related through the coherence principle [?): if $\neg L$ holds, then not $L$ should also hold (similarly, if $L$ then not $\neg L$). The $WFSX_p$ does not enforce consistency with respect to default negation. This particularity allows support on contradiction detection: a non default negated literal $L$ has support on contradiction in a model $M$ iff both $L$ and not $L$ belong to $M$. Hence $M$ is an inconsistent $WFSX_p$ model iff both $L$ and not $L$ belong to $M$, for some non default negated literal $L$. The same characterization of inconsistency follows for $MH_p$ models.

The results presented in this paper are enounced for the universe of finite ground normal logic programs. The proofs not presented in the paper may be found in [?].

The rest of the paper proceeds as follows. In section ?? we define the language of extended logic programs and the terminology to be used in the sequel. For self-containment we present in sections ?? and ?? the definitions of the $MH$ semantics and $WFSX_p$ semantics. In section ?? we exhibit in technical detail the definition and characterization of the $MH_p$ semantics. Section ?? is dedicated to conclusions and future work.
2 Language and Terminology of Logic Programs

An extended logic program defined over a language $L$, is a finite set of ground extended rules (or simply rules, for short), each one of the form

$$l_0 ← l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n,$$

where each $l_i$, $0 \leq i \leq n$, is an objective literal (either an atom $b$ or its explicit negation, $\neg b$, where $\neg b = b$); $m, n$ are integer non negative numbers; the operator ‘$\cdot$’ stands for the conjunctive connective and the operator ‘not’ stands for default negation (not $l$ is called a default literal). A literal (program) is ground if it does not contain variables. The set of all ground objective literals involved in an extended logic program, plus their explicit negations, is named the (extended) Herbrand base of $P$, denoted $\mathcal{H}_P$. If $m = n = 0$ a rule is called fact. A program containing no explicitly negated literals is called normal program; a rule with no explicitly negated literals is called normal rule. Given a rule $r = l_0 ← l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$, the objective literal $l_0$ is the head of the rule and $l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$ is the body of the rule.

For ease of exposition, we henceforth use the following abbreviations: $\text{Atom}(L)$, is the unitary set containing the atom involved in the ground literal $L$; $\text{Body}(r)$, is the set of literals in the body of the ground rule $r$; $\text{Heads}(E)$, is the set of all objective literals that appear in the heads of the set of ground rules $E$; if $E$ is unitary, we may use ‘Head’ instead of ‘Heads’; $\text{Facts}(E)$, is the set of all facts that appear in the set of ground rules $E$; $\text{ObLiterals}(E)$, is the set of all objective literals that appear in the conjunction of literals $E$. We may compound some of these abbreviations, as for instance $\text{ObLiterals}(\text{Body}(r))$ whose meaning is immediate. Each of the abbreviations may also be taken as the conjunction of the elements contained in the respective sets.

Some terminology used in the sequel is now established, concerning the dependencies among the elements (objective literals and rules) of ground extended logic programs, triggered by the dependency operator ‘$\leftarrow$’.

**Complete rule graph.** (adapted from [??]) The complete rule graph of an extended logic program $P$, denoted by $\text{CRG}(P)$, is the directed graph whose vertices are the rules of $P$. Two vertices representing rules $r_1$ and $r_2$ define an arc from $r_1$ to $r_2$ iff $\text{Head}(r_1) \subseteq \text{ObLiterals}(\text{Body}(r_2))$, or $\text{Head}(r_1) = \neg \text{Head}(r_2)$.

**Rule depending on a rule.** (adapted from [??]) Let $P$ be an extended logic program. We say that a rule $r$ depends on a rule $s$ iff there is a direct path from $s$ to $r$ in the complete rule graph of $P$.

**Subprogram relevant to an objective literal.** (adapted from [??]) Let $P$ be an extended 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 $\text{Head}(s) = L$, and $s$ depends on $r$. In particular, rule $s$ is relevant to $L$. The set of all rules of $P$ relevant to $L$ is represented by $\text{Rel}_P(L)$, and is named subprogram relevant to $L$.

**Loop.** (adapted from [??]) A set of normal rules $R$ forms a loop (or the rules of
set \( R \) are in loop) iff for any two rules \( r, s \in R \), \( r \) depends on \( s \) and \( s \) depends on \( r \). We say that rule \( r \in R \) is in loop through literal \( L \in \text{Body}(r) \) iff there is a rule \( s \in R \) such that \( \text{Head}(s) = \text{Atom}(L) \).\(^3\)

Given a 3-valued interpretation \( I \) of an extended logic program, we represent by \( I^+ \) (resp. \( I^- \)) the set of its 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 \( (b \leftarrow) \in Q; b \in \triangle Q \) iff \( (b \leftarrow) \not\in Q \) and there is a rule \( r \) in \( Q \) such that \( \text{Head}(r) = b; b \in \triangle Q \) iff \( b \) has no rule in \( Q \).

### 3 The \( MH \) Semantics

In this section we define the *minimal hypotheses semantics* for normal logic programs, \( MH \). \( MH \) is an abductive 2-valued semantics,\(^5\) whose set of models for a normal logic program contains all the stable models of the program. For this reason we say \( MH \) is a a *conservative extension* of \( SM \). We start by introducing the concept of reduction system [\[?\].

#### 3.1 Reduction System

In [\[?] the authors propose a set of five operations to reduce a normal logic program (i.e., eliminate rules or literals), while maintaining the well-founded semantics [\[?] of the resulting programs – positive reduction, PR, negative reduction, NR, success, S, failure, F, and loop detection, L – whose definitions are as follows (\( P_1 \) and \( P_2 \) being two ground normal logic programs):

1. **Positive reduction, PR.** Program \( P_2 \) results from \( P_1 \) by *positive reduction* iff there is a rule \( r \in P_1 \) and a default literal *not* \( b \in \text{Body}(r) \) such that \( b \not\in \text{Heads}(P_1) \), and \( P_2 = (P_1 \setminus \{r\}) \cup \{\text{Head}(r) \leftarrow (\text{Body}(r) \setminus \{\text{not } b\})\} \).

2. **Negative reduction, NR.** Program \( P_2 \) results from \( P_1 \) by *negative reduction* iff there is a rule \( r \in P_1 \) and a default literal *not* \( b \in \text{Body}(r) \) such that \( b \in \text{Facts}(P_1) \), and \( P_2 = P_1 \setminus \{r\} \).

\(^3\) A definition of loop can be stated for extended logic programs, but only the normal logic programs version is used in this paper.

\(^4\) We also write \( b = +, b = u, b = - \), to mean respectively, \( b \in I^+, b \in I^u, b \in I^- \).

\(^5\) See [\[?] for abductive semantics.
3. Success, S. Program $P_2$ results from $P_1$ by success if there is a rule $r \in P_1$ and a fact $b \in \text{Facts}(P_1)$ such that $b \in \text{Body}(r)$, and $P_2 = (P_1 \setminus \{r\}) \cup \{\text{Head}(r) \leftarrow (\text{Body}(r) \setminus \{b\})\}$.

4. Failure, F. Program $P_2$ results from $P_1$ by failure if there is a rule $r \in P_1$ and a positive literal $b \in \text{Body}(r)$ such that $b \not\in \text{Heads}(P_1)$, and $P_2 = P_1 \cap \{r\}$.

5. Loop Detection, L. Program $P_2$ results from $P_1$ by loop detection if there is a set $A$ of ground atoms such that:
   (a) For each rule $r \in P_1$, if $\text{Head}(r) \in A$, then $\text{Body}(r) \cap A \neq \emptyset$;
   (b) $P_2 := \{r \in P_1 | \text{Body}(r) \cap A = \emptyset\}$.

We represent this set of operations by $\mapsto_{WFS} := \{\text{PR, NR, S, F, L}\}$. By applying non-deterministically this set of operations on a program $P$, until the resulting program becomes invariant under any further operation of $\mapsto_{WFS}$, we obtain the program $\hat{P}$, named remainder of $P$. This transformation is terminating and confluent, meaning that for any finite ground normal logic program $P$, the number of operations needed to reach $\hat{P}$ is finite, and the order in which the operations are performed is irrelevant. We denote the transformation of $P$ into $\hat{P}$ as $P \mapsto_{WFS} \hat{P}$. It is shown in [1] that $\text{WFM}(P) = \text{WFM}(\hat{P})$, where $\text{WFM}$ stands for well-founded model [2]. It is also the case that $\text{SM}(P) = \text{SM}(\hat{P})$.

The next example shows this transformation at work.

Example 1. Computing the remainder of a program. Let $P$ be the set of rules below. The remainder $\hat{P}$ is the non shadowed part of the program (the labels (i)–(v) indicate the operations used in the corresponding reductions, as pinpointed in the legend).

\[
\begin{align*}
 a & \leftarrow \text{not } f \quad \text{(i)} \\
 a & \leftarrow \text{not } b \quad \text{(i)} \\
 b & \leftarrow \text{not } a \quad \text{(ii)} \\
 c & \leftarrow a \quad \text{(iii)} \\
 d & \leftarrow f \quad \text{(iv)} \\
 c & \leftarrow d \quad \text{(v)} \\
 d & \leftarrow e \quad \text{(v)}
\end{align*}
\]

Legend: (i) PR, (ii) NR, (iii) S, (iv) F, (v) L.

One way to obtain conservative extensions of the $\text{SM}$ semantics, is to relax some operations of the reduction system $\mapsto_{WFS}$, which yields weaker reduction systems, that is, systems that erase less rules or literals than $\mapsto_{WFS}$. This is the option taken in the next subsection to define the $\text{MH}$ models.

3.2 Minimal Hypotheses Semantics Models

The minimal hypotheses semantics $\text{MH}$ [2], is a semantics whose reduction system $\mapsto_{LWFS}$ is obtained from $\mapsto_{WFS}$ by replacing the negative reduction operation $\text{NR}$, by the layered negative reduction operation $\text{LNR}$, i.e., $\mapsto_{LWFS} := \mapsto_{WFS}$. The expression 'layered' comes from the division of a program into layers,[2] which inspired all concepts in this paper involving this word or variations on it.
LNR is a weaker version of NR that instead of eliminating any rule \( r \) containing say \( \text{not } b \) in the body, in the presence of the fact \( b \), as NR does, only eliminates rule \( r \) if this rule is not in loop through literal \( \text{not } b \).

We write \( P \mapsto_{LWFS} \hat{P} \), where \( \hat{P} \) is the layered remainder of \( P \). In the case of program \( P \) in example ?? above, the layered remainder \( \hat{P} \) is the non shadowed part of the program below.

\[
\begin{align*}
a & \leftarrow \text{not } f \\
a & \leftarrow \text{not } b \\
b & \leftarrow \text{not } a \\
c & \leftarrow a \\
d & \leftarrow f \\
c & \leftarrow d \\
d & \leftarrow c
\end{align*}
\]

Notice that rule \( r = (b \leftarrow \text{not } a) \) is no longer eliminated by the fact \( a \), since rule \( r \) and rule \( a \leftarrow \text{not } b \) are in loop, and in the case of rule \( r \) the loop is through the literal \( \text{not } a \). The \( MH \) models of a program \( P \) are then computed as follows: (1) Take as assumable hypotheses set of \( P \), \( \text{Hyps}(P) \), the set of all atoms that appear default negated in \( \hat{P} \); in the case of the previous program we have \( \text{Hyps}(P) = \{a, b\} \); (2) Form all programs \( P \cup H \), for all possible subsets \( H \subseteq \text{Hyps}(P) \), \( H \neq \emptyset \) (if \( \text{Hyps}(P) = \emptyset \), then \( H = \emptyset \) is the only set to consider); take all the interpretations for which \( WFM(P \cup H) \) is a total model (meaning a model that has no undefined literals); \( H \) is the hypotheses set of the interpretation \( WFM(P \cup H) \); (3) Take all the interpretations obtained in the previous point, and chose as \( MH \) models the ones that have minimal \( H \) sets with respect to set inclusion. The \( MH \) models of program \( P \) in the example above, and the corresponding hypotheses sets, are

\[
\begin{align*}
M_1 = \{a, \text{not } b, c, \text{not } d, \text{not } e, \text{not } f\} & \quad H = \{a\} \\
M_2 = \{a, b, c, \text{not } d, \text{not } e, \text{not } f\} & \quad H = \{b\}.
\end{align*}
\]

Notice that \( M_1 \) is the only \( SM \) model of \( P \). The \( \mapsto_{LWFS} \) reduction system keeps some loops intact and uses them as choice devices for generating \( MH \) models, allowing us to have \( MH(P) \supseteq SM(P) \). The sets \( H \) considered may be taken as abductive explanations [?] for the corresponding models. \( MH \) enjoys the properties of existence and simple relevance (see proposition ??).

4 The WFSX\(_p\) Semantics

The \( WFSX_p \) model of an extended logic program may be computed by means of a dedicated fixpoint operator [?]. In this section, however, we instead present a definition of the \( WFSX_p \) by means of a program transformation for extended logic programs, dubbed \( t-o \) transformation\(^7\) [?], which embeds the \( WFSX_p \) into

\(^7\) The original designation is \( T-TU \) transformation. Our definition alters the notation, for simplicity purposes, while keeping the meaning of the original one.
The Abductive Paraconsistent Semantics MHp

The WFS. This means that the WFSXp model of an extended logic program P, denoted by WFMp(P), may be extracted from the well-founded model of the transformed program Pt−o.

Definition 2. t−o Transformation. (adapted from [?]) The t−o transformation maps an extended logic program P into a normal logic program Pt−o, by means of the two following steps:

1. Every explicitly negated literal in P, say ¬b, appears also in the transformed program, where it must be read as a new atom (instead of an explicitly negated literal). Let P* be the program resulting from making these transformations on P.
2. Every rule r = (Head(r) ← Body(r)) in P* is substituted by the following pair of rules: (i) A rule also designated r, for simplicity, obtained from r ∈ P* by placing the superscript ‘o’ in the default negated atoms that appear in Body(r); (ii) A rule r°, obtained from r ∈ P* by adding to Body(r) the literal not ¬Head(r) (where ¬¬l = l)⁸, and by placing the superscript ‘o’ in Head(r) and in every non default negated literal of Body(r).

We call co-rules to each pair r, r° of rules in Pt−o (each rule of the pair is the co-rule of the other one), and co-atoms to each pair b, b° of atoms in Pt−o (each atom of the pair is the co-atom of the other one). To each atom b of the language of P*, there corresponds the pair of co-atoms b, b° of the language of Pt−o and vice-versa.

Example 2. (adapted from [?]) Program P (left column) has the t−o transformed Pt−o (two columns on the right):

```
<table>
<thead>
<tr>
<th>a</th>
<th>a°</th>
<th>¬a</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬a</td>
<td>¬a°</td>
<td>¬a°</td>
</tr>
<tr>
<td>b</td>
<td>a°</td>
<td>¬a°</td>
</tr>
<tr>
<td>c</td>
<td>¬b</td>
<td>¬b°</td>
</tr>
<tr>
<td>d</td>
<td>¬d</td>
<td>¬d°</td>
</tr>
</tbody>
</table>
```

Notice that program Pt−o is a normal logic program: notations ¬L and ¬L° must be taken as new atoms names, instead of explicitly negated literals.

The following theorem states how to read the WFMp(P) model from the WFM(Pt−o), where P is any extended logic program.

Theorem 1. WFSXp is embeddable into WFS. (adapted from [?]) Let P be an extended logic program. The following equivalences hold for an arbitrary objective literal L of the language of P:

- L ∈ WFMp(P) iff L ∈ WFM(Pt−o);
- not L ∈ WFMp(P) iff not L° ∈ WFM(Pt−o);

⁸ The literal not ¬Head(r) is added to enforce the coherence principle.
– \( \neg L \in WFMP(p) \) iff \( \neg L \in WFMP(P^{t-\circ}) \);
– \( \text{not } \neg L \in WFMP(p) \) iff \( \text{not } \neg L^\circ \in WFMP(P^{t-\circ}) \);

where the symbols \( \neg L, \neg L^\circ \) on the right sides of the ‘iff’s’ must be taken as new atoms names (instead of explicitly negated literals), in accordance with point 1 of definition ??

The following operator shall be used in the sequel.

**Definition 3.** \( \triangledown \) operator. Given a 3-valued interpretation \( I = (I^+, I^u, I^-) \), where \( I^+, I^u, I^- \) may contain 'o' superscript or otherwise non superscript atoms, we denote by \( \triangledown I \) the interpretation obtained by means of the equivalences stated in theorem ??, substituting \( WFMP(P^{t-\circ}) \) for \( I \) and \( WFMP(p) \) for \( \triangledown I \).

Using this operator we may write, for example, \( WFMP(p) = \triangledown WFMP(P^{t-\circ}) \).

The next proposition characterizes the valuations of the pairs of co-atoms \( b, b^\circ \) of the language of \( P^{t-\circ} \), with respect to the \( WFMP(P^{t-\circ}) \): it is the case that \( b^\circ \leq_I b \) for any such pair \((b^\circ, b)\) standing here for their valuations with respect to the \( WFMP(P^{t-\circ}) \), where \( \leq_I \) is the truth ordering\(^9\).

**Proposition 1.** Let \( b, b^\circ \) be two co-atoms of the Herbrand base of \( P^{t-\circ} \). It is not possible to have any of the following three types of valuations with respect to the \( WFMP(P^{t-\circ}) \): \( (b, b^\circ) = (-, +) \), \( (b, b^\circ) = (-, u) \), \( (b, b^\circ) = (u, +) \). The only possible valuations are \( (b, b^\circ) = (-, -) \), \( (b, b^\circ) = (+, -) \), \( (b, b^\circ) = (u, u) \), \( (b, b^\circ) = (+, u) \), \( (b, b^\circ) = (+, +) \).

**Proof.** The proposition is an immediate consequence of theorem ??.

In example ?? we have

\[
WFMP(P^{t-\circ}) = \langle \{a, \neg a, b, c\}^+, \{d, d^\circ\}^u, \\
\{a^2, \neg a^2, b^\circ, \neg b^\circ, c^2, \neg c^2, \neg d^\circ, \neg d, \neg d^\circ\}^\circ \rangle.
\]

Using theorem ?? we obtain the \( WFSXP_p \) model

\[
WFMP(p) = \langle \{a, \neg a, b, c\}^+, \{d\}^u, \{a, \neg a, b, \neg b, c, \neg c, \neg d\}^\circ \rangle.
\]

Notice that \( d \in WFMP^u_p(P) \) means that neither \( d \) nor \( \text{not } d \) belong to the \( WFMP(p) \) defined by theorem ??. To get the meaning of this model, notice that \( WFSXP_p \) does not enforce default consistency, i.e., \( L \) and \( \text{not } L \) can be simultaneously true in a paraconsistent model, in contradistinction to all other paraconsistent semantics [?]. This feature allows the detection of dependence on contradictory information, and is a consequence of adopting the **coherence principle** [?]. In the above example, for any \( L \in \{a, \neg a, b, c\} \), both \( L \) and \( \text{not } L \) belong to the \( WFMP(p) \), thus revealing the valuations of \( \{a, b, c\} \) as inconsistent (the

\[^9\] Given the logic values \( f \) (false), \( u \) (undefined) and \( t \) (true), their **truth ordering** is defined by: \( f \leq u \leq t \) [?].
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 inconsistencies of \( a, b, c \) valuations occur. The valuation of atom \( d \) is in turn consistent. Table ?? (see appendix) presents in the line \( \text{NINE}^{10} \) the correspondence between each possible 4-tuple \( (L^o, L, \neg L^o, \neg L) \) \( \text{WFM}(P^{t-o}) \) valuation and the nine possible logic values literal \( L \) may assume with respect to logic \( \text{NINE} \). According to table ??, the logic values of the atoms in the \( \text{NINE} \) model corresponding to the \( \text{WFM}_p(P) \) 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 \( dt \)).

The following definition adapts the notions of partial and total models \([?]\) to the paraconsistent case.

**Definition 4. Total/Partial Paraconsistent Model.** (adapted from \([?]\)) Let \( SEM \) be a paraconsistent semantics for extended logic programs and \( M \) a \( SEM \) model of an extended logic program \( P \). We say that \( M \) is a total paraconsistent model if for every objective literal \( L \) of the language of \( P \), either \( L \in M \) or \( \neg L \in M \). We say that \( M \) is a partial paraconsistent model iff there is an objective literal \( L \) of the language of \( P \), such that \( L \notin M \) and \( \neg L \notin M \).

The next proposition shows that \( \text{WFSX}_p \) is a partial models semantics.

**Proposition 2.** The \( \text{WFSX}_p \) semantics is a partial paraconsistent models semantics, meaning that the \( \text{WFSX}_p \) models of some extended logic programs are partial paraconsistent models.

The following theorem shows that \( \text{WFSX}_p \) collapses into \( \text{WFS} \) for normal logic programs.

**Theorem 2.** (adapted from \([?]\)) If \( P \) is a normal logic program, then the models \( \text{WFM}(P) \) and \( \text{WFM}_p(P) \) are equal, if we neglect all the literals of the type \( \neg L \) in the \( \text{WFM}_p(P) \).

### 5 The \( MH_p \) Semantics

The \( MH_p \) semantics of an extended logic program \( P \) is computed via the following steps: 1) Compute the balanced layered remainder \( bP^{t-o} \) of \( P \) (see definition ??); 2) Compute the assumable hypotheses set of \( P \) (see definition ??); 3) Use the assumable hypotheses set to obtain the \( MH_p \) models of \( P \) (see definition ??).

The balanced layered remainder \( bP^{t-o} \) of an extended logic program \( P \), is computed by resorting to a transformation system, \( \rightarrow_{bLWF_S} \), obtained from \( \rightarrow_{WF_S} \) by replacing the negative reduction operation \( NR \), by the balanced layered negative reduction operation \( bLN_R \) defined below. This operation adapts the layered negative reduction \( LN_R \) to the \( P^{t-o} \) structure.

\(^{10}\) A logic dubbed \( \text{NINE} \) \([?]\) provides a truth-functional model theory for the \( \text{WFSX}_p \) semantics.
Definition 5. Balanced\textsuperscript{11} Layered Negative Reduction. Let $P_1$ and $P_2$ be two ground normal logic programs, whose Herbrand bases contain 'o' superscript and non superscript atoms. We say that $P_2$ results from $P_1$ by a balanced layered negative reduction operation iff one of the next two cases occur: (1) There is a fact $b^\circ$ in $P_1$ and a rule $r$ in $P_1$ whose body contains the literal not $b^\circ$, where neither $r$ is involved in a loop through the literal not $b^\circ$, nor is its co-rule $r^\circ$ 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 $r^\circ$ in $P_1$ whose body contains the literal not $b$, where neither $r^\circ$ is involved in a loop through the literal not $b$, nor is its co-rule $r$ involved in a loop through the literal not $b^\circ$, and $P_2 = P_1 \setminus \{r^\circ\}$.

Definition 6. Balanced Layered Reduction. The balanced layered reduction system is the system $\rightarrow_{bLWFS} := \rightarrow_{PR} \cup \rightarrow_{bLNR} \cup \rightarrow_S \cup \rightarrow_F \cup \rightarrow_L$.

Theorem 3. Termination and Confluency. The system $\rightarrow_{bLWFS}$, when applied to finite ground programs, is both terminating and confluent.

Proof. This result is a consequence of both the validity of the same properties for the system $\rightarrow_{WFS}$, and $\rightarrow_{bLWFS}$ being a weaker reduction system than $\rightarrow_{WFS}$.

Definition 7. Balanced Layered Remainder. Let $P$ be a finite ground extended logic program and $P^\rightarrow_{t-o}$ its $t-o$ transformed. We call balanced layered remainder of $P$ to the program $bP^\rightarrow_{t-o}$ such that $P^\rightarrow_{t-o} \rightarrow_{bLWFS} bP^\rightarrow_{t-o}$.

Example 3. We represent below the balanced layered remainder $bP^\rightarrow_{t-o}$ (two columns on the right) of the program $P$ (left column) – shadowed rules and literals are eliminated during $bP^\rightarrow_{t-o}$ computation.

\[
\begin{align*}
  b & \leftarrow h & b & \leftarrow h & h^\circ & \leftarrow h^\circ, \text{not } \neg b \\
  h & \leftarrow \text{not } p & h & \leftarrow \text{not } p^\circ & h^\circ & \leftarrow \text{not } p, \text{not } \neg h \\
  p & \leftarrow \text{not } b & p & \leftarrow \text{not } b^\circ & p^\circ & \leftarrow \text{not } b, \text{not } \neg p \\
  b & \leftarrow & b & \leftarrow & b^\circ & \leftarrow \text{not } \neg b \\
  \neg h & \leftarrow & \neg h & \leftarrow & \neg h^\circ & \leftarrow \text{not } h
\end{align*}
\]

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

\textsuperscript{11} The expression ‘balanced’ refers to the consideration of pairs of co-rules in this definition.
The Abductive Paraconsistent Semantics MHp

The next result shows the remainder $bP^{t-o}$ is well defined, since the co-atoms valuations in $\Delta bP^{t-o}$ satisfy proposition ??.

**Proposition 3.** Let $P$ be an extended logic program and $M = \Delta bP^{t-o}$. Then the valuation with respect to $M$ of any pair of co-atoms, say $L, L^\circ$, of the Herbrand base of $P^{t-o}$, agrees with the statement of proposition ??.

**Proof.** The result stated in proposition ?? is valid for the well-founded model of the $t-o$ transformed $P^{t-o}$ of an extended logic program $P$. The WFM of a normal logic program can be computed by resorting to the remainder of the program, through the reduction system $\Rightarrow_{WFS}$. As the balanced layered remainder of $P$ is computed via the system $\Rightarrow_{bLWFS}$, we just need to prove that the operation $bLNR$ does not originate any violation of the possible valuations of co-atoms stated in proposition ??; That no such violation may occur stems from the very definition of $bLNR$.

Definitions ?? and ?? below are adaptations of MH analogous concepts (see subsection ??) to the $P^{t-o}$ structure dealt with in the computation of $MH_p$ models.

**Definition 8. Assumable Hypotheses Set of a Program.** Let $P$ be a finite ground extended logic program and $bP^{t-o}$ its balanced layered remainder. We say that $\text{Hyps}(P) \subseteq \mathcal{H}_P$ is the assumable hypotheses set of $P$ if for all $h \in \text{Hyps}(P)$ it is the case that the default literal $\text{not } h^o$ appears in program $bP^{t-o}$.

**Definition 9. Paraconsistent Minimal Hypotheses Semantics, $MH_p$.** Let $P$ be a finite ground extended logic program, $bP^{t-o}$ its balanced layered remainder and $\text{Hyps}(P)$ the set of assumable hypotheses of $P$. Then the paraconsistent minimal hypotheses semantics of $P$, denoted $MH_p(P)$, is defined by the paraconsistent minimal hypotheses models $M$ of $P$, which are computed as follows:

1. $M = WFMP(P \cup H) = \vee WFMP(P^{t-o} \cup H \cup \{h^o \leftarrow \text{not } \neg h : h \in H\})$, for all $H \subseteq \text{Hyps}(P)$, $H \neq \emptyset$, $H$ being inclusion-minimal and $WFMP(P \cup H) \neq \emptyset$.
2. $M = WFMP(P) = \vee WFMP(P^{t-o})$, if $\text{Hyps}(P) = \emptyset$.

Each hypothesis, say $h \in H$, is added to $bP^{t-o}$ as a pair of rules $\{h \leftarrow, h^o \leftarrow \text{not } \neg h\}$. No $MH_p$ model has undefined objective literals, meaning literals $L$ such that neither $L$ nor not $L$ belong to the model. Table ?? (see appendix) presents in line $SIX^{13}$ the correspondence between each 4-tuple $(L^o, L, \neg L, \neg L) \Delta bP^{t-o}$ valuation and the six possible logic values literal $L$ may assume with respect to $SIX$.

---

12 Notice that the purpose of computing $bP^{t-o}$ is to find the set of assumable hypotheses of $P$.

13 Logic $SIX$ provides a truth-functional model theory for the $MH_p$ semantics.
Example 4. Consider the program in example ??.. The assumable hypotheses set of \( P \) is \( \{ b, p \} \), since both \( \text{not } b^o, \text{not } p^o \) appear in \( bP^{t-o} \). The \( MH_p \) models are:

\[
M_1 = \bigvee \text{WFM}(P^{t-o} \cup \{ b \} \cup \{ b^o \leftarrow \text{not } b \}) \\
= \bigvee \{(b, b^o, h, \text{not } h)^{+}, \{ \text{not } b, \text{not } b^o, h^o, \text{not } h^o, p, p^o, \text{not } p, \text{not } p^o \}^{-} \}
\]

\[
M_2 = \bigvee \text{WFM}(P^{t-o} \cup \{ p \} \cup \{ p^o \leftarrow \text{not } p \}) \\
= \bigvee \{(b, b^o, \text{not } h, h^o, p, p^o)^{+}, \{ \text{not } b, b^o, \text{not } h, h^o, p, p^o \}^{-} \}
\]

We see that the first model is inconsistent while the second is not. The first model coincides with the \( \text{WFM}_p(P) \). The second model arises because the \( MH_p \) uses the loop \( \{ b \leftarrow h, h \leftarrow \text{not } p^o, p^o \leftarrow \text{not } b, \text{not } p \} \) of program \( bP^{t-o} \) as a choice device. According to line \( \text{SIX} \) of table ??, 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 \( f \)).

The following result shows that \( MH_p \) is a total paraconsistent models semantics (cf. definition ??).

Proposition 4. The \( MH_p \) semantics is a total paraconsistent models semantics, meaning that for any extended logic program \( P \) all the \( MH_p \) models of \( P \) are total paraconsistent models.

The next theorem shows that \( MH \) and \( MH_p \) coincide for normal logic programs, if we discard literals of the type \( \text{not } \neg L \) from the \( MH_p \) models.

Theorem 4. \( MH_p \) 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 \( P \).

Formal Properties

The \( MH_p \) semantics enjoys the properties of existence and simple relevance.

Definition 10. Existence. We say that a semantics \( SEM \) has the property of existence iff every logic program \( P \) has at least one \( SEM \) model.

Definition 11. Simple relevance. We say that a semantics \( SEM \) has the property of simple relevance iff for any logic program \( P \), whenever there is a \( SEM \) model \( M_l \) of \( \text{Rel}_p(l) \) such that \( l \in M_l \), there is also a \( SEM \) model \( M \) of \( P \) such that \( l \in M \).
Proposition 5. Both $MH$ and $MH_p$ enjoy the properties of existence and simple relevance.

Complexity

14The 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 $\Sigma^P_2$.

Theorem 5. Brave reasoning with $MH_p$ semantics is in $\Sigma^P_2$.

Proof. Let us show that finding a $MH_p$ model of an extended logic program $P$ is in $\Sigma^P_2$. Computing $P^{d-o}$ is fulfilled in linear time. The balanced layered remainder $bP^{d-o}$ is computed in polynomial time, by the following reasoning: the calculus of the remainder of a normal logic program is known to be of polynomial time complexity [?]; the difference between $\rightarrow_{WFS}$ and $\rightarrow_{MLWFS}$ lies on the operation $NR$ of the former being replaced by the operation $bLNR$ of the latter; to perform $bLNR$ the rule layering$^{15}$ must be computed; the rule layering can be calculated in polynomial time since it is equivalent to identifying the strongly connected components (SCCs) [?] in a graph, in this case in the complete rule graph of $P^{d-e}$; once the SCCs are found, one collects their heads in sets, one set for each SCC – this is all linear time; when verifying the preconditions to perform a balanced layered 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 checking if the co-rule is in loop through $not b$; therefore, balanced layered negative reduction adds only polynomial time complexity operations over negative reduction. Once $bP^{d-o}$ is computed, non deterministically guess a set $H$ of hypotheses (computing the assumable hypotheses set is linear time). Check if $WFMP_u^o(P \cup H) = \emptyset$ – this is polynomial time. Checking that $H$ is non-empty minimal, requires another non deterministic guess of a strict subset $H'$ of $H$ and then a polynomial check if $WFMP_u^o(P \cup H') = \emptyset$.

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 $\Pi^P_2$.

Theorem 6. Cautious reasoning with $MH_p$ semantics is in $\Pi^P_2$.

Proof. Cautious reasoning is the complement of brave reasoning, and since the latter is in $\Sigma^P_2$ the former must necessarily be in $\Pi^P_2$.

$^{14}$ This subsection follows closely subsection 4.6 of [?].

$^{15}$ See [?] for rule layering definition. This notion can be adapted to extended logic programs without changing the complexity of the normal logic programs layering computation algorithm.
6 Conclusions and Future Work

We have presented an abductive paraconsistent semantics $MH_p$, that inherits the $MH$ properties of existence and simple relevance, and also the support based on contradiction detection capability of the paraconsistent $WFSX_p$. $MH_p$ 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 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 consistent $MH_p$ model of $P$; **Skeptical Paraconsistent Reasoning:** Query $Q$ succeeds iff it succeeds for all $MH_p$ models of $P$, such that none of the literals in $ObLiterals(Q)$ has support on contradiction for any of these models; **Brave Paraconsistent Reasoning:** Query $Q$ succeeds iff it succeeds for at least one $MH_p$ model of $P$, such that none of the literals in $ObLiterals(Q)$ 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 $P$. Brave liberal reasoning can be performed computing $MH_p$ models of the union of relevant subprograms $Rel_P(L)$, for every objective literal $L \in ObLiterals(Q)$, due to the simple relevance property of $MH_p$. 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 [?].

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

**Appendix**

**Logics NINE and SIX**

In [?] the author presents a logic, there dubbed $NINE$, that provides a truth-functional model theory for the $WFSX_p$, based on Ginsberg’s bilattices concept.

<table>
<thead>
<tr>
<th>$L'$</th>
<th>$\bot$</th>
<th>$\top$</th>
<th>$a$</th>
<th>$u$</th>
<th>$+$</th>
<th>$-$</th>
<th>$u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
</tr>
<tr>
<td>$\neg L'$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
</tr>
<tr>
<td>$\neg L$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
<td>$u$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NINE</th>
<th>IV</th>
<th>IV</th>
<th>df</th>
<th>III</th>
<th>df</th>
<th>III</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIX</td>
<td>IV</td>
<td>IV</td>
<td>df</td>
<td>III</td>
<td>df</td>
<td>III</td>
<td>f</td>
</tr>
</tbody>
</table>

**Table 1. NINE and SIX valuations**
The truth-space of \textit{NINE} comprises nine logic values, here presented together with their meanings: ‘\textit{t}’ and ‘\textit{f}’ are the classical values for truth and falsity; ‘\textit{I}’, is the contradictory truth value; ‘\textit{II}’, is understood as truth with contradictory belief; ‘\textit{III}’, is understood as falsity with contradictory belief; ‘\textit{IV}’, is understood as contradictory belief; ‘\textit{⊥}’, is understood as undefinedness; ‘\textit{df}’, is understood as default falsity; ‘\textit{dt}’, is understood as default truth. Using the ideas set forth in the definition of \textit{NINE} we define \textit{SIX}, a truth-functional model theory for the \textit{MHp}, whose truth-space is \{\textit{t, f, I, II, III, IV}\}. Table ?? represents all the possible 4-tuple \((\textit{L}, \textit{L}, \neg\textit{L}, \neg\textit{L})\) valuations with the corresponding \textit{NINE} and \textit{SIX} logic values. There are 20 possible 4-tuple valuations, as a consequence of proposition ?? and the coherence principle.