Kiel Declarative Programming Days 2013

20th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2013)
22nd International Workshop on Functional and (Constraint) Logic Programming (WFLP 2013)
27th Workshop on Logic Programming (WLP 2013)

Michael Hanus, Ricardo Rocha (Eds.)
Bericht Nr. 1306
September 2013
ISSN 2192-6247
Kiel Declarative Programming Days 2013

20th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2013)
22nd International Workshop on Functional and (Constraint) Logic Programming (WFLP 2013)
27th Workshop on Logic Programming (WLP 2013)

Michael Hanus, Ricardo Rocha (Eds.)

Bericht Nr. 1306
September 2013
ISSN 2192-6247

e-mail: mh@informatik.uni-kiel.de, ricroc@dcc.fc.up.pt

This technical report contains the papers presented at the Kiel Declarative Programming Days held during September 11-13, 2013 in Kiel, Germany.
Preface

This report contains the papers presented at the *Kiel Declarative Programming Days 2013*, held in Kiel (Germany) during September 11-13, 2013. The Kiel Declarative Programming Days 2013 unified the following events:

- 20th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2013)
- 22nd International Workshop on Functional and (Constraint) Logic Programming (WFLP 2013)
- 27th Workshop on Logic Programming (WLP 2013)

All these events are centered around *declarative programming*, an advanced paradigm for the modeling and solving of complex problems. These specification and implementation methods attracted increasing attention over the last decades, e.g., in the domains of databases and natural language processing, for modeling and processing combinatorial problems, and for high-level programming of complex, in particular, knowledge-based systems.

The INAP conferences provide a communicative forum for intensive discussion of applications of important technologies around logic programming, constraint problem solving, and closely related computing paradigms. It comprehensively covers the impact of programmable logic solvers in the internet society, its underlying technologies, and leading edge applications in industry, commerce, government, and societal services. Previous INAP editions have been held in Japan, Germany, Portugal, and Austria.

The international workshops on functional and logic programming (WFLP) aim at bringing together researchers interested in functional programming, logic programming, as well as the integration of these paradigms. Previous WFLP editions have been held in Germany, France, Spain, Italy, Estonia, Brazil, Denmark, and Japan.

The workshops on (constraint) logic programming (WLP) serve as the scientific forum of the annual meeting of the Society of Logic Programming (GLP e.V.) and bring together researchers interested in logic programming, constraint programming, and related areas like databases, artificial intelligence, and operations research. Previous WLP editions have been held in Germany, Austria, Switzerland, and Egypt.

In this year these events were jointly organized under the umbrella of the Kiel Declarative Programming Days in order to promote the cross-fertilizing exchange of ideas and experiences among researchers and students from the different communities interested in the foundations, applications, and combinations of high-level, declarative programming languages and related areas. The technical program of the event included presentations of refereed technical papers and system descriptions.

The program committees collected for each paper at least three reviews. The meetings of the program committees were conducted electronically during July 2013 with the help of the conference management system EasyChair. After careful discussions, the program committees selected 21 technical papers and two system descriptions for presentation. In addition to the selected papers, the scientific program included an invited lecture by Tom Schrijvers (University of Ghent, Belgium).
We would like to thank all authors who submitted papers to this event. We are grateful to the members of the program committees and all the additional reviewers for their careful and efficient work in the review process. Finally, we express our gratitude to all members of the local organizing committee for their help in organizing a successful event.

September 2013

Michael Hanus
Ricardo Rocha
Conference Organization

Conference Chairs

**WFLP/WLP**
Michael Hanus  
Institut für Informatik  
Christian-Albrechts-Universität Kiel  
24098 Kiel, Germany  
E-mail: mh@informatik.uni-kiel.de

**INAP**
Ricardo Rocha  
Departamento de Ciência de Computadores  
Universidade do Porto  
4169-007 Porto, Portugal  
E-mail: ricroc@dcc.fc.up.pt

WFLP/WLP Program Committee

- Elvira Albert  
  Universidad Complutense de Madrid, Spain
- Sergio Antoy  
  Portland State University, USA
- François Bry  
  University of Munich, Germany
- Jürgen Dix  
  Clausthal University of Technology, Germany
- Rachid Echahed  
  CNRS, University of Grenoble, France
- Moreno Falaschi  
  Università di Siena, Italy
- Sebastian Fischer  
  Kiel, Germany
- Thom Frühwirth  
  University of Ulm, Germany
- Michael Hanus  
  University of Kiel, Germany (Chair)
- Oleg Kiselyov  
  Monterey (CA), USA
- Herbert Kuchen  
  University of Münster, Germany
- Francisco Javier López-Fraguas  
  Universidad Complutense de Madrid, Spain
- Torsten Schaub  
  University of Potsdam, Germany
- Peter Schneider-Kamp  
  University of Southern Denmark, Denmark
- Dietmar Seipel  
  University of Würzburg, Germany
- Hans Tompits  
  Vienna University of Technology, Austria
- German Vidal  
  Universidad Politécnica de Valencia, Spain
- Janis Voigtländer  
  University of Bonn, Germany
INAP Committees

Track Chairs

Nonmonotonic Reasoning
Hans Tompits
Vienna University of Technology, Austria

Applications and System Implementations
Masanobu Umeda
Kyushu Institute of Technology, Japan

Extensions of Logic Programming
Salvador Abreu
University of Évora, Portugal

Databases, Deductive Databases and Data Mining
Dietmar Seipel
University of Würzburg, Germany

Program Committee

Salvador Abreu
University of Évora, Portugal
Sergio Alvarez
Boston College, USA
Christoph Beierle
FernUniversität in Hagen, Germany
Philippe Codognet
JFLI/CNRS at University of Tokyo, Japan
Daniel Diaz
University of Paris I, France
Ulrich Geske
University of Potsdam, Germany
Petra Hofstedt
Brandenburg University of Technology at Cottbus, Germany
Katsumi Inoue
National Institute of Informatics, Japan
Gabriele Kern-Isberner
University of Dortmund, Germany
Ulrich Neumerkel
Vienna University of Technology, Austria
Vitor Nogueira
University of Évora, Portugal
Enrico Pontelli
New Mexico State University, USA
Ricardo Rocha
University of Porto, Portugal (Chair)
Irene Rodrigues
University of Évora, Portugal
Carolina Ruiz
Worcester Polytechnic Institute, USA
Vítor Santos Costa
University of Porto, Portugal
Dietmar Seipel
University of Würzburg, Germany
Terrance Swift
Universidade Nova de Lisboa, Portugal
Hans Tompits
Vienna University of Technology, Austria
Masanobu Umeda
Kyushu Institute of Technology, Japan
Marina De Vos
University of Bath, United Kingdom
Armin Wolf
Fraunhofer FIRST, Berlin, Germany
Osamu Yoshie
Waseda University, Japan
Local Organization

Linda Haberland
Michael Hanus
Björn Peemöller
Fabian Reck
Jan Rasmus Tikovsky

University of Kiel, Germany
University of Kiel, Germany
University of Kiel, Germany
University of Kiel, Germany
University of Kiel, Germany

External Reviewers

Demis Ballis
Steffen Ernsting
Raúl Gutiérrez
Benny Höckner
Andy Jost
Arne König
Ludwig Ostermayer
Tony Ribeiro
José Miguel Rojas
Javier Romero
Miguel A. Salido
Peter Sauer
Luca Torella
Amira Zaki
# Table of Contents

Delimited Continuations for Prolog: An Overview (Invited Talk)  
*Tom Schrijvers*  

Extension of Gelfond-Lifschitz Reduction for Preferred Answer Sets: Preliminary Report  
*Alexander Šimko*  

Construction of Explanation Graphs from Extended Dependency Graphs for Answer Set Programs  
*Ella Albrecht, Patrick Krümpelmann and Gabriele Kern-Isberner*  

Some Experiments on Light-Weight Object-Functional-Logic Programming in Java with Paisley  
*Baltasar Trancón y Widemann and Markus Lepper*  

On an Approach to Implementing Exact Real Arithmetic in Curry  
*Christoph Beierle and Udo Lelitko*  

And... action! – Monoid Acts and (Pre)orders  
*Nikita Danilenko*  

Sharing and Exchanging Data  
*Rana Awada, Pablo Barceló and Iluju Kiringa*  

On Axiomatic Rejection for the Description Logic $\mathcal{ALC}$  
*Gerald Berger and Hans Tompits*  

Introducing Real Variables and Integer Objective Functions to Answer Set Programming  
*Guohua Liu, Tomi Janhunen and Ilkka Niemelä*  

Propositional Encoding of Constraints over Tree-Shaped Data  
*Alexander Bau and Johannes Waldmann*  

Coverage Driven Test Generation and Consistency Algorithm  
*Jomu George Mani Paret and Otmane Ait Mohamed*  

Towards a Verification Framework for Haskell by Combining Graph Transformation Units and SAT Solving  
*Marcus Ermler*  

FOBS-X: An Extensible Hybrid Functional-Object-Oriented Scripting Language  
*James Gil de Lamadrid*
Towards Parallel Constraint-Based Local Search with the X10 Language
Danny Munera, Daniel Diaz and Salvador Abreu ........................................ 168

Constraint-based Approach for an Early Inspection of the Feasibility of Cyber Physical Systems
Benny Höckner, Petra Hofstedt, Thilo Vörtlter, Peter Sauer and Thomas Hinze .......... 183

Simplifying the Development of Rules Using Domain Specific Languages in DROOLS
Ludwig Ostermayer, Geng Sun and Dietmar Seipel ........................................ 198

A SAT-Based Graph Rewriting and Verification Tool Implemented in Haskell
Marcus Ermler ......................................................................................... 213

Euler/X: A Toolkit for Logic-based Taxonomy Integration
Mingmin Chen, Shizhuo Yu, Nico Franz, Shawn Bowers and Bertram Ludäscher ....... 219

Debate Games in Logic Programming
Chiaki Sakama .............................................................................. 225

A Datalog Engine for GPUs
Carlos Alberto Martínez-Angeles, Inês Dutra, Vítor Santos Costa and Jorge Buenabad-Chávez ................................................................. 239

Heuristic Search Over Program Transformations
Claus Zinn ...................................................................................... 254

HEX-Programs with Existential Quantification
Thomas Eiter, Michael Fink, Thomas Krennwallner and Christoph Redl .................. 269

An Abductive Paraconsistent Semantics – $MHP$
Mário Abrantes and Luís Moniz Pereira ............................................ 284

Meta-set calculus as mathematical basis for creating abstract, structured data store querying technology
Mikus Vanags, Arturs Licis and Janis Justs ............................................. 299

Author Index ..................................................................................... 315
An Abductive Paraconsistent Semantics – MH_P

Mário Abrantes\textsuperscript{1}* and Luís Moniz Pereira\textsuperscript{2}

\textsuperscript{1} Escola Superior de Tecnologia e de Gestão
Instituto Politécnico de Bragança
Campus de Santa Apolónia, 5300-253 Bragança, Portugal
mar@ipb.pt

\textsuperscript{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 extended normal logic programs, the paraconsistent minimal hypotheses semantics MH_P. The MH_P 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 \cite{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 \cite{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 abductive based paraconsistent semantics, for extended logic programs, the MH_P, which combines the merits of two already existing semantics: an abductive semantics, the minimal hypotheses semantics MH \cite{1}, and a paraconsistent semantics, the well-founded semantics with explicit negation WFSX_P \cite{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 \cite{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 Fundaço\~ao para a Ci\~encia e Tecnologia and Instituto Politécnico de Bragança grant PROTEC : SFHR/49747/2009.
Example 1. In order for program \( P = \{ a \leftarrow \neg b, b \leftarrow \neg c, c \leftarrow \neg a \} \), 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 \( P \), which is a transformed of the original program \( P \). In the example above, as \( P = \hat{P} \), 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 \( \text{no	extunderscore fly}(X) \leftarrow \text{penguin}(X) \). Meanwhile, if additionally we wish to represent the statement "Birds fly", \( \text{fly}(X) \leftarrow \text{birds}(X) \), no connection results between the predicates \( \text{no	extunderscore fly}(X) \) and \( \text{fly}(X) \), although the intention of the programmer is to set them as contradictory. In this case it is suitable to have the rule \( \neg \text{fly}(X) \leftarrow \text{penguin}(X) \) instead of \( \text{no	extunderscore fly}(X) \leftarrow \text{penguin}(X) \), since a semantics that deals with the operator \( \neg \) will by definition consider predicates \( \text{fly}(X) \) and \( \neg \text{fly}(X) \) 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 $l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$, 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 $r = l_0 \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$, 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.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 ’←’.

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 \( \text{Head}(r_1) \in \text{Atoms}(<\text{Body}(r_2)), \) or \( \text{Head}(r_1) = \neg \text{Head}(r_2), \) or \( \text{Head}(r_2) = \neg \text{Head}(r_1) \).

**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 \( \text{Heads}(s) = l \), 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 \( \text{Rel}_P(l) \), 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 \( \{ h_1 \leftarrow l_2, B_1, h_2 \leftarrow l_3, B_2, \ldots, h_n \leftarrow l_1, B_n \} \), where \( l_i = h_i \), or \( l_i = \text{not} \ h_i \), 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 \( I = (I^+, I^u, I^-) \).

The following operator shall be used in the sequel.

**Definition 1. \( \Delta \) operator.** Given a normal logic program \( Q \), we denote by \( \Delta Q \) the 3-valued interpretation that can be read from \( Q \) in the following way: \( b \in (\Delta Q)^+ \) iff \( (b \leftarrow) \in Q; b \in (\Delta Q)^u \) iff \( (b \leftarrow) \notin Q \) and there is a rule \( r \in Q \) such that \( \text{Head}(r) = b; b \in (\Delta Q)^- \) 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*, \( \mapsto_L \). We here represent this reduction system by \( \mapsto_{\text{WFS}} := \mapsto_P \cup \mapsto_N \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L \). Given a normal logic program \( P \), the transformation \( P \mapsto_{\text{WFS}} \tilde{P} \) is such that \( \text{WFM}(\tilde{P}) = \text{WFM}(P) \). Program \( \tilde{P} \) will be here called the WFS remainder of \( P \) or remainder of \( P \) \( ^7 \). The system \( \mapsto_{\text{WFS}} \) is both terminating and confluent, meaning that for any finite ground normal logic program the number of operations needed to reach \( \tilde{P} \) 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 \( \mapsto_{\text{WFS}} \) means the nondeterministic performing of operations of the system \( \mapsto_{\text{WFS}} \), 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 $\rightarrow_{LWFS}$ of the
reduction system $\rightarrow_{WFS}$, which results from $\rightarrow_{WFS}$ by substituting the negative
reduction operator, $\rightarrow_N$, by the layered negative reduction operator, $\rightarrow_{LN}$ – see
[1], def. 10. Given a normal logic program $P$, the transformation $P \rightarrow_{LWFS}^\ast \hat{P}$
8 is such that $\triangle \hat{P} = LWFS(P)$, where $LWFS(P)$ [4] stands for the layered well-
founded model of $P$ – the transformed program $\hat{P}$ is called $LWFS$ remainder or
layered remainder of $P$. The system $\rightarrow_{LWFS}$ is terminating and confluent when
applied to finite ground normal logic programs.

Example 3. The layered remainder of program $P$ (left column) is program $\hat{P}$
(center column), and the remainder of $P$ is program $\tilde{P}$ (right column) – rules
and literals stripped out, are eliminated during $\hat{P}$ and $\tilde{P}$ computations.\footnote{Rule $d \leftarrow \text{not } b$ in $P$ is eliminated, both in $\hat{P}$ and $\tilde{P}$, by layered negative reduction in the first case and by negative reduction in the second – the two operations coincide here, since the rule is not in loop via literal $\text{not } b$; the body of rule $a \leftarrow \text{not } c, b$ in $P$, becomes empty by success (which eliminates $b$) and by positive reduction (which eliminates $\text{not } c$), both in $\hat{P}$ and $\tilde{P}$; rule $p \leftarrow \text{not } b$ in is eliminated in $\hat{P}$ by negative reduction, but not in $\tilde{P}$, since the rule is in loop through literal $\text{not } b$ and layered negative reduction does nothing is such cases; $b$ is eliminated from the body of rule $h \leftarrow \text{not } p, b$ by success, both in $\hat{P}$ and $\tilde{P}$, whilst positive reduction eliminates also the literal $\text{not } p$ in $\tilde{P}$, because $p$ is an atom without rule due to the elimination of rule $p \leftarrow \text{not } b$, turning $h$ into a fact; $h$ is eliminated from the body of rule $b \leftarrow h$, by success, in $\tilde{P}$.}

\begin{align*}
b &\leftarrow h & b &\leftarrow h & b &\leftarrow h \\
h &\leftarrow \text{not } p, b & h &\leftarrow \text{not } p, b & h &\leftarrow \text{not } p, b \\
p &\leftarrow \text{not } b & p &\leftarrow \text{not } b & p &\leftarrow \text{not } b \\
a &\leftarrow \text{not } c, b & a &\leftarrow \text{not } c, b & a &\leftarrow \text{not } c, b \\
d &\leftarrow \text{not } b & d &\leftarrow \text{not } b & d &\leftarrow \text{not } b \\
b &\leftarrow & b &\leftarrow & b &\leftarrow
\end{align*}

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

$MH$ being an abductive semantics, we shall now define the\footnote{\textit{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 \textit{assumable hypotheses set} of $P$: those atoms that appear default negated in the bodies of rules of $\hat{P}$ and which are not facts in $\hat{P}$, i.e. they belong to $\triangle \hat{P}$\textsuperscript{u}.} assumable hypotheses set and the\footnote{\textit{Minimal Hypotheses Model of a Program.}} minimal hypotheses model of a normal logic program.

Definition 2. Assumable Hypotheses Set of a Program. \textit{(adapted from [1])} Let $P$ be a finite ground normal logic program. We write $Hyps(P)$ to denote the \textit{assumable hypotheses set} of $P$: those atoms that appear default negated in the bodies of rules of $\hat{P}$ and which are not facts in $\hat{P}$, i.e. they belong to $\triangle \hat{P}$\textsuperscript{u}.\footnote{\textit{The layered well-founded model of $P$ is thus $LWFM(P) = \triangle \hat{P} = \langle \{a, b\}^+, \{h, p\}^u, \{c, d\}^- \rangle$, and the well-founded model of $P$ is $WFM(P) = \triangle \tilde{P} = \langle \{a, b, h\}^+, \{\}^u, \{c, d, p\}^- \rangle$.}
The purpose of computing \( \hat{P} \) is to find the set of assumable hypotheses of \( P \), which are then used to compute the minimal hypotheses models of the program.

**Definition 3. Minimal Hypotheses Model.** (adapted from [1]) Let \( P \) be a finite ground normal logic program. Let \( \text{Hyps}(P) \) be the assumable hypotheses set of \( P \) (cf. def. 2), and \( H \) a subset of \( \text{Hyps}(P) \). A 2-valued interpretation \( M \) of \( P \) is a minimal hypotheses model of \( P \), iff \( \text{WFM}^u(P \cup H) = \emptyset \), 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 \( \text{Hyps}(P) = \{p\} \), since \( \text{not } p \) appears in \( \hat{P} \) and \( p \) is not a fact of this program.\(^{10}\) The MH models of \( P \) are \( \{a, b, \text{not } c, h, \text{not } p\} \) with hypotheses set \( \emptyset \), and \( \{a, b, \text{not } c, \text{not } h, p\} \) 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 \( \{a, b, \text{not } c, h, \text{not } p\} \).

**Example 4.** Consider the following variation of the vacation problem\(^{11}\) [1], \( P = \{a \leftarrow \text{not } b, b \leftarrow \text{not } a, c \leftarrow \text{not } d, d \leftarrow \text{not } e, \text{not } a, e \leftarrow \text{not } a, \text{not } c\} \), where \( P = \hat{P} \). The hypotheses set of \( P \) is \( \text{Hyps}(P) = \{a, b, c, d, e\} \). The MH models of \( P \) are: \( \{a, \text{not } b, c, \text{not } d, \text{not } e\} \) with hypotheses set \( \{a\} \), \( \{\text{not } a, b, \text{not } c, d, e\} \) with hypotheses set \( \{b, d\} \)\(^{12}\) and \( \{a, \text{not } b, c, \text{not } d, e\} \) 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. \( \{a, \text{not } b, c, \text{not } d, e\} \) above). Should there be a transformation on normal logic programs, let it be \( \Rightarrow_Y \), such that \( P \Rightarrow_Y P^* \), 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 \( \text{not } b \) appears in \( \hat{P} \), \( b \) is not an assumable hypothesis of \( P \) since it is a fact of \( \hat{P} \).

\(^{11}\) 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 \( a \leftarrow \text{not } b \); 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 \( \text{WFM}^u(P \cup H) \neq \emptyset \).
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.\textsuperscript{13}

4 The WFSX\textsubscript{P} Semantics

The WFSX\textsubscript{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\textsubscript{P} by means of a program transformation for extended normal logic programs, dubbed \textit{t − o transformation}\textsuperscript{14} [20], which embeds the WFSX\textsubscript{P} into the WFS. This means that the WFSX\textsubscript{P} model of an extended normal logic program \(P\), denoted by \(WFM_{\text{P}}(P)\), may be extracted from the well-founded model of the transformed program \(P^{t−o}\).

\textbf{Definition 4. t − o Transformation.} (adapted from [20]) The \textit{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 \(r = (\text{Head}(r) \leftarrow \text{Body}(r))\) 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 \(\text{Body}(r)\); (ii) A rule \(r^{o}\), obtained from \(r \in P^*\) by adding to \(\text{Body}(r)\) the literal \(\text{not} \neg \text{Head}(r)\) (where \(\neg \neg l = l\))\textsuperscript{15}, and by placing the superscript ‘\(o\)’ in \(\text{Head}(r)\) and in every positive literal of \(\text{Body}(r)\).

We call \textit{co-rules} to each pair \(r, r^{o}\) of rules in \(P^{t−o}\) (each rule is the \textit{co-rule} of the other one), and \textit{co-atoms} to each pair \(b, b^{o}\) of atoms in \(P^{t−o}\) (each atom is the \textit{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.\textsuperscript{16}

\textsuperscript{13}The solution proposed by MH semantics for the vacation problem has the following rationale: rule \(a \leftarrow \text{not} b\), 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, retrieves models that satisfy 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).

\textsuperscript{14}The original designation is \textit{T − TU transformation}. Our definition alters the notation, for simplicity purposes, while keeping the meaning of the original one.

\textsuperscript{15}The literal \(\text{not} \neg \text{Head}(r)\) is added to enforce the coherence principle.

\textsuperscript{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):

\[
\begin{align*}
  a & \leftarrow a \\
\neg a & \leftarrow \neg a \\
b & \leftarrow a \\
c & \leftarrow \neg b \\
d & \leftarrow \neg d
\end{align*}
\]

\[
\begin{align*}
  a & \leftarrow a \leftarrow \neg a \\
b & \leftarrow a \\
c & \leftarrow \neg b \\
d & \leftarrow \neg d
\end{align*}
\]

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

**Theorem 2.** $WFSP$ 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 $b \in WFMP(P^{t-o})$; $\neg b \in WFMP(P)$, iff not $b^{o} \in WFMP(P^{t-o})$; $\neg \neg b \in WFMP(P)$, iff $\neg b \in WFMP(P^{t-o})$, where the symbols $\neg b, \neg b^{o}$ on the right sides of the 'if's' must be taken as names of atoms (they are not explicitly negated literals), in accordance with definition 4.

**Definition 5.** $\triangledown$ 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 $\triangledown I$ the 3-valued interpretation obtained by means of the lexical correspondences stated in theorem 2.

Using this operator we may write, for example, $WFMP(P) = \triangledown WFMP(P^{t-o})$. 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 $WFMP(P^{t-o})$: 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 $WFMP(P^{t-o})$, where $\leq t$ is the truth ordering\(^\text{17}\).

**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 $\Gamma$ 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 $WFMP(P^{t-o})$: $(b,b^{o}) = (-, +)$, $(b,b^{o}) = (-, u)$, $(b,b^{o}) = (u, +)$. The only possible valuations are $(b,b^{o}) = (-, -)$, $(b,b^{o}) = (u, -)$, $(b,b^{o}) = (+, -)$, $(b,b^{o}) = (u, u)$, $(b,b^{o}) = (+, u)$, $(b,b^{o}) = (+, +)$.

\(^\text{17}\)Given the logic values $f$ (false), $u$ (undefined) and $t$ (true), their truth ordering is defined by: $f \leq t \leq u$, [11].
The $WFM(P_{t-o})$ in example 5 is
\[
\langle \{a, \neg a, b, c\}^+, \{d, d^o\}^u, \{a^o, \neg a^o, b^o, \neg b^o, c^o, \neg c^o, \neg d, \neg d^o\}^\ominus \rangle.
\]
Using theorem 2 we obtain the $WFSX_P$ model $\langle \{a, \neg a, b, c\}^+, \{d\}^u, \{a, \neg a, b, c, \neg c, \neg d\}^\ominus \rangle$. 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\textsuperscript{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 outcome of the balanced layered reduction transformation of $P$, $P \rightarrow_{bLWFM}$

\textsuperscript{18} A logic dubbed NINE [20], presented in the appendix, provides a truth-functional model theory for the $WFSX_P$ semantics.
bP^{t-o}$, where (i) $\rightarrow_{bLWFM}$ is obtained from $\rightarrow_{LWFM}$ replacing the layered negative reduction operator, $\rightarrow_{LN}$, by the balanced layered negative reduction operator, $\rightarrow_{bLN}$, defined below; (ii) $\rightarrow_{bLWFM}$ means the nondeterministic performing of operations of the system $\rightarrow_{bLWFM}$ until the resulting program becomes invariant under any further operation.

**Definition 7. Balanced\textsuperscript{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, $P_1 \rightarrow_{bLN} P_2$, 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 $\neg b^o$, where neither $r$ is involved in a loop through the literal $\neg b^o$ nor is its co-rule $r^o$ involved in a loop through the literal $\neg b^o$, 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 $\neg b^o$, where neither $r^o$ is involved in a loop through the literal $\neg b^o$, nor is its co-rule $r$ involved in a loop through the literal $\neg b^o$, and $P_2 = P_1 \setminus \{r^o\}$\textsuperscript{20}.

**Definition 8. Balanced Layered Reduction.** The balanced layered reduction system is the system $\rightarrow_{bLWFM} := \rightarrow \cup \rightarrow_{bLN} \cup \rightarrow_S \cup \rightarrow_F \cup \rightarrow_L$.

**Theorem 4. Termination and Confluence.** The system $\rightarrow_{bLWFM}$, 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 $P^{t-o} \rightarrow_{bLWFM}^* bP^{t-o}$.

**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.$^\textsuperscript{21}$

$$
\begin{align*}
\text{b} & \leftarrow \text{h} & \text{b} & \leftarrow \text{h} & \text{b}^o & \leftarrow \text{h}^o, \neg \text{b}^o \\
\text{h} & \leftarrow \neg \text{p} & \text{h} & \leftarrow \neg \text{p}^o & \text{h}^o & \leftarrow \neg \text{p}^o, \neg \text{b}^o \\
\text{p} & \leftarrow \neg \text{b} & \text{p} & \leftarrow \neg \text{b}^o & \text{p}^o & \leftarrow \neg \text{b}^o, \neg \text{p} \\
\text{b} & \leftarrow & \text{b} & \leftarrow & \text{b}^o & \leftarrow \neg \text{b}^o \\
\neg \text{h} & \leftarrow & \neg \text{h} & \leftarrow & \neg \text{h}^o & \leftarrow \neg \text{h}
\end{align*}
$$

\textsuperscript{19} The expression ‘balanced’ refers to the consideration of pairs of co-rules in this definition.

\textsuperscript{20} This operation is weaker than layered negative reduction, meaning that where the former is applicable so is the latter.

\textsuperscript{21} Notice that rule $h^o \leftarrow \neg p$, $\neg \text{h}$ is eliminated by balanced negative reduction, because $\neg \text{h}$ is a fact of $bP^{t-o}$, and neither is the rule involved in a loop through literal $\neg \text{h}$, nor is its co-rule $h \leftarrow \neg p^o$ involved in a loop through literal $\neg h^o$ – balanced layered negative reduction has, in this case, the same effect as negative reduction; rule $b^o \leftarrow h^o$, $\neg \text{b}$ is eliminated by failure, because $h^o$ does not have a rule after elimination of $h^o \leftarrow \neg p$, $\neg \text{h}$; literals $\neg \text{b}$, $\neg \text{p}$ are eliminated by positive reduction, since $\neg \text{b}$, $\neg \text{p}$ do not have a rule in $bP^{t-o}$.
**Proposition 3.** Let $P$ be an extended normal logic program and $M = \triangle bP^{l-o}$. Then the valuation with respect to $M$ of any pair of co-atoms, say $l, l'$, of the Herbrand base of $P^{l-o}$, 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 $bP^{l-o}$ its balanced layered remainder. We say that $Hyp(P) \subseteq H_P$ 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 $bP^{l-o}$, and $h$ is not a fact in $bP^{l-o}$, i.e. $h \in (\triangle bP^{l-o})^u$.22

**Definition 11. Paraconsistent Minimal Hypotheses Semantics, MH$_P$.** Let $P$ be a finite ground extended normal logic program, $bP^{l-o}$ its balanced layered remainder and $Hyps$ 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. $M = WFM_P(P \cup H) = \nabla WFM(P^{l-o} \cup H \cup \{ h^o \leftarrow \text{not } \neg h : h \in H \})$, for all $H \subseteq Hyps$, $H \neq \emptyset$, $H$ is inclusion-minimal and $WFM^u_P(P \cup H) = \emptyset$.
2. $M = WFM_P(P) = \nabla WFM(P^{l-o})$, where $WFM^u_P(P) = \emptyset$.

Each hypothesis, say $h$, is added to $bP^{l-o}$ as a pair of rules $\{ h \leftarrow, h^o \leftarrow \text{not } \neg h \}$. 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 $bP^{l-o}$, $p$ is not a fact in the program $bP^{l-o}$, whilst $b$ is. The $MH_P$ models are:24

$$M_1 = \nabla WFM(P^{l-o})$$
$$= \nabla \langle \{ b, b^o, h, \neg h \}^+, \{ \}^u, \{ \neg b, \neg b^o, h^o, \neg h^o, p, p^o, \neg p, \neg p^o \}^\ominus \rangle$$
$$= \langle \{ b, h, \neg h \}^+, \{ \}^u, \{ \neg b, h, \neg h, p, \neg p \}^\ominus \rangle,$$ with hypotheses set $\emptyset$

$$M_2 = \nabla WFM(P^{l-o} \cup \{ p \} \cup \{ p^o \leftarrow \text{not } \neg p \})$$
$$= \nabla \langle \{ b, b^o, \neg h, \neg h^o, p, p^o \}^+, \{ \}^u, \{ \neg b, \neg b^o, h, h^o, \neg p, \neg p^o \}^\ominus \rangle$$
$$= \langle \{ b, \neg h, p \}^+, \{ \}^u, \{ \neg b, h, \neg p \}^\ominus \rangle,$$ with hypotheses set $\{ p \}$

---

22 This is equivalent to saying that $h \in (\nabla \triangle bP^{l-o})^u$. Notice that the purpose of computing $bP^{l-o}$, 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 ← h, h ← not p, p ← 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 $MH_P$ 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.** 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 $P$.

### 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 $\Sigma_2^P$.

**Theorem 6.** Brave reasoning with $MH_P$ semantics is in $\Sigma_2^P$.

**Proof.** Let us show that finding a $MH_P$ model of an extended normal logic program $P$ is in $\Sigma_2^P$. 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 known to be of polynomial time complexity [17]; the difference between $\rightarrow_{WFS}$ and $\rightarrow_{bLWFS}$ lies on the operator $\rightarrow_N$ of the former being replaced by the operator $\rightarrow_{bLN}$ of the latter; to perform $\rightarrow_{bLN}$ 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 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 \( bP^{l-o} \) is computed, nondeterministically guess a set \( H \) of hypotheses – the assumable hypotheses set, is the set of all atoms involved in default negations in \( bP^{l-o} \), that are not facts. Check if \( WF_{PM}(P \cup H) = \emptyset \) – 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 \( WF_{PM}(P \cup H') = \emptyset \).

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 7.** 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 \( \Pi^P_2 \). \( \square \)

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


297
An Abductive Paraconsistent Semantics – MH
20. Damásio, C.V., Pereira, L.M.: A survey of paraconsistent semantics for logic pro-
grams. Handbook of defeasible reasoning and uncertainty management systems:
volume 2: reasoning with actual and potential contradictions (1998) 241–320
22. Alferes, J.J., Damásio, C.V., Pereira, L.M.: Top-down query evaluation for well-
(1972) 146–160
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; ‘T’,
is the contradictory truth value; ‘III’, is understood as truth with contradictory
belief; ‘IV’, is understood as contradictory belief; ‘⊥’, 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, l, ¬l, ¬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>
<thead>
<tr>
<th>l</th>
<th>l</th>
<th>¬l</th>
<th>¬l</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
</tbody>
</table>

Table 1. NINE and SIX valuations