Default negation in Normal Logic Programs considered as minimal abduction of positive hypotheses

Alexandre Miguel Pinto and Luís Moniz Pereira

Abstract

Logic Programs (LPs) are a practical tool for declarative knowledge representation and consist of normal rules and Integrity Constraints (ICs). Reasoning with LPs is parameterized by the particular semantics chosen. But the declaratively of the knowledge represented by an LP is restricted if the semantics chosen for the normal rules allows them to play the role ICs already can have. Namely because odd loops over negation, such as in rule $\{p \leftarrow not\ p\}$, entail the absence of 2-valued semantics models and complex odd loops over negation are often deliberately used as ICs. Here the authors propose a more flexible reading of default negation, such that NLPs always have a 2-valued model before ICs are evaluated. To wit, the authors do so by allowing for minimally assuming (or abducing) positive hypotheses and hence still maximising the assumption of negative hypotheses that preserve consistency. The authors show how that translates into a semantics for Normal Logic Programs (NLPs) – the Minimal Hypotheses (MH) semantics – which safeguards declarativity in this sense and moreover enjoys useful semantic properties such as cumulativity and relevancy, besides existence. Moreover, the authors introduce a program transformation which allows to compute the MH models of a program as a selection of the Stable Models of the transform.

Key words: Normal Logic Programs, Minimal Hypotheses Semantics, Stable Models Semantics, Hypothetical Reasoning, Abduction, Default Negation.

Alexandre Miguel Pinto
Outra Limited, London, United Kingdom, e-mail: apinto@outra.co.uk

Luís Moniz Pereira
NOVA-LINCS - Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, Portugal, e-mail: lmp@fct.unl.pt

Introduction

Logic Programs (LPs) are intended to be a declarative Knowledge Representation (KR) formalism. Normal Logic Program (NLPs) consist both of regular normal rules and Integrity Constraints (ICs) which are normal rules with the special $\bot$ atom as head. The intuitive meaning classically assigned to rules is that the atom in the head of a rule must be *true* in some model of the NLP if the whole body of the rule is *true* in the same model, namely all of both positive and default negated literals (DNLs). This reading of NLP rules is supported in the intuitive meaning classically assigned to default negation in a rule body: that the truth of a negative literal results from the unprovability of its positive atom, and therefore its negation by default. Such a meaning of default negation makes sense for stratified programs, i.e. with non-circular dependencies over default negation. However, when there are rules which are loop-dependent on each other via default negated literals in their bodies, then there is no assured unprovability of the positive atoms that appear negated in the loop, if there be no other non-looping rules for them.

A well-known 2-valued semantics like the Stable Models (SMs) Gelfond and Lifschitz (1988) builds on top of these intuitions and as a consequence it assigns the intuitively correct meaning to NLPs with stratified negation. But not so in all cases of NLPs with non-stratified negation. In the case of programs with default negation loops all of which are even (ELONs), such as $P = \{a \leftarrow not\ b \quad b \leftarrow not\ a\}$, the semantics can produce alternative models, depending on how each of the ELONs is broken, via opting for (or abducing) the truth of one of their default negated literals, after maximally having opted for the truth of default literals not in a loop. Several alternatives being available for breaking ELONs, the choices result thereby in distinct models. Equivalently, one could break each ELON by opting for (or abducing), the truth of the atom of one default literal in each ELON, after minimally (rather than maximally) opting for the truth of those atoms not involved in loops, since default negation takes precedence (the Closed World Assumption). In the case of programs containing unresolved odd loops over default negation (OLONs), such as $P = \{p \leftarrow not\ p\}$, the SMs semantics decides for not producing any model, and views such OLONs as unsatisfiable constraints.

Our approach regarding default negation loops is a different one, as detailed in full in this paper. In short, for both OLONs and ELONs, the authors minimally abduce (for the collection of loops) some alternative positive atoms of the default literals present in unbroken loops, in order to break them. Mark that minimality of positive abductions entails maximality of the negative ones. In summary, since abducing default literals in loops is not enough to consistently break OLONs then instead, in our approach, one will minimally abduce positive atoms in order to do so. The authors do not consider unbroken OLONs as unsatisfiable constraints, and relegate constraining to integrity constraints proper.

Furthermore, since a default negation loop may depend on another, but not both on each other for that would produce a single default negation loop, it is necessary to establish the new notion of layering of default negation loops, a notion that generalises stratification and establishes which default negation loops depend on lower

level ones. The latter have precedence in being broken by means of abduction before tackling higher level ones, as these may become broken by choices made at lower levels. Such a precedence is in tune with the Logic Programming paradigm.
To the best of our knowledge, the first work that envisaged ground default literals and their atoms as abducibles and proceeded to show how abduction can be employed to obtain the Stable Models of a Normal Logic Program without odd loops over negation was Eshghi and Kowalski (1989). What our work newly does is to show how to deal with (unbroken) odd loops over negation, so that every NLP has a semantics, thus generalising the Stable Models Semantics, which Eshghi and Kowalski (1989) do not consider. Additionally, the present authors do not restrict themselves to ground abducibles. To achieve the above the authors need to define the concept of layering, a generalisation of stratification, inasmuch odd loops over negation may be intertwined with one another. And let the semantics proceed from the bottom layers upwards. Moreover, the authors minimise abducing positive literals so that, whenever abducing them to break odd loops is not required, one falls back into regular SMs, and obtain all the SMs. If breaking odd loops is needed one still obtains all the SMs plus more models. One may imagine what the present authors do is a prioritised layered circumscription, prioritising negative default literals over their positive atoms. Furthermore, a program transformation to a new program is introduced whose SMs can be read to be our models.

Motivation

Our motivation is concerned with the KR ability of LPs. For one, one wishes to allow programs with OLONs to have a model, for that extends the scope of KR ability, as our examples will show. The use of OLONs for expressing constraints can be assigned instead to explicit integrity constraints (ICs) (whether or not they be coded with syntactic sugar).
For another, one wishes regular NLPs (those without ICs) to always have a model, in spite of any OLONs. Even when they are obtained by merging together regular NLPs from a diversity of in/out sources, possibly without OLONs, or are obtained by program updating, this may nevertheless result in OLONs in the target NLP, as shown in examples below. Such resulting OLONs might not have been foreseen as desired constraints, leading unexpectedly to the absence of a model, with the consequence of preventing the desired liveness of the KR so constructed. Instead of being represented via OLONs, desired ICs should be explicitly stated, thus adding to the transparency of the KR.
Yet for another, it will be seen that our 2-valued Minimal Hypothesis semantics (MH) for regular NLPs, as constructed in detail below, enjoys the properties of model existence, cumulativity and relevancy, not enjoyed by other 2-valued semantics, such as the SMs one. Cumulativity permits the storing of intermediate lemmas to speed up future computations. Relevancy is particularly important for the introduction of abduction in general in NLPs, since it guarantees abduction by-need, i.e.

the existence of merely top-down algorithms to produce solutions to a goal, which may restrict themselves to the dependency call-graph among atoms. As a result, literals in MH models have a proper non-circular justification when minimal abduction of positive atoms is enacted, because the semantic's abduced atoms provide extra support.
Such non-circular justifications will also be obtained when extra user-defined abducibles, attending to abductive reasoning, are introduced too into the program for KR applications, say for planning. However, it is not in the scope of this semantics foundational work to delve such abductive reasoning rich applications. That exploration is left for future work, with the knowledge that KR in a 2-valued efficiently implemented MH semantics can be granted richer possibilities, including the alternative KR rich ways OLONs can now be broken to generate solutions.
Mark that the MH semantics is related to Reiter's default logic ensuring model existence, as every default extension has a corresponding MH model. This is because positive atoms can be assumed to hold in default logic, as permitted in MH-models. For abductive LPs, the MH-semantics allows for positive hypotheses while past approaches allow only for negative ones.

It is worthy of note that when minimal positive abductions are absent, then MH models coincide with SMs, so that MH's range of applications covers that of SMs semantics. Include herein, is a program transformation for regular NLPs such that the resulting SMs are the MH models of the original program. If no positive abductions corresponding to the original program's positive atoms are present in the so-obtained SM of the transform, then there exists a corresponding SM of the original program. Otherwise, one have a strict MH model with no SM correspondence, thus showing how MH can go beyond SM's KR expressiveness all the while containing it. For all the SMs uses of OLONs as constraints can be expressed instead as regular ICs, as the authors found out in a plethora of applications. The reason being that the constraining OLONs are inserted by design and equivalent ICs can satisfy the design. Namely by constructing the ICs exactly with the rule body conditions that would bring about the OLONs, but not fully depending on them. A simple example is the program $P = \{p \leftarrow not\ p, a, b \qquad b \leftarrow p \qquad a \leftarrow\}$. In order to avoid the surging of the odd loop over p, the IC $\bot \leftarrow a$ is required; b does not play a role because it depends only on the loop itself. The concept of dependency is formalised in the sequel.

The relationship of MH semantics to SMs semantics is emphasised here. Rather than exploring other avenues to deal with inconsistency, since the authors value the widely accepted success of SMs (and of Answer Set Programs (ASP)) they have therefore aimed to build an arguably natural extension to it — generalisable to ASP but the extra complications have been avoided in an already long paper). Hence too our choice of 2-valuedness rather than considering 3-valuedness, also because the latter case was pursued by one of us already in Alferes et al. (2004).
Nowithstanding, the authors consider a comparison with the 3-valued approach of Sakama and Inoue (1995). Theirs is different from ours on the one hand, though, on the other, the two approaches share the same goal of providing a meaning to every LP, inclusively disjunctive ones in their case. In their semi-stable semantics, they

consider a modal multi-valued logic and, for program $P = \{p \leftarrow not\ p\}$, interpret p as "believed-true". One can consider the interpretation "believed-true" similar to our explicit assumption of positive abductive hypotheses.
Similarly, Eiter et al. (2010) and their subsequent work improve on the properties of the above mentioned semi-stable semantics by introducing a paracoherent one, characterised by the semi-equilibrium models of a program in a modal multi-valued logic with a belief operator. Once more, they make no use of the fixed point semantics of the transformed LP resulting from a confluent set of rewrite rules as the present authors do, nor use explicit abductive hypotheses, as is our case. But their programs' truth beliefs are notionally comparable to abduction by belief. Their aim to remove inconsistencies is similar to ours, but their approach, which includes avoiding classical contradictions too, is formally so different from our own, which aims to stay close to SMs semantics, that a comparison in detail of any bridges between the two is far removed from the scope of the present paper.

Declarativity and Knowledge Representation Role Independence

One intended meaning of ICs in LPs (those with special head $\bot$) is clear and widely agreed upon: no model of the whole LP can render *true* the body of any IC — in this sense, the part of a LP not containing ICs, can be ascribed, from a KR point of view, to be a generator of candidate models, if only the empty one, whereas the ICs part is assigned the role of pruning away undesired candidate models. Irrespective of how ICs are specifically coded and syntactically sugared in whatever implemented system.
For the sake of KR declaratively of LPs and separateness of the special IC rules from the other (regular) rules, their respective KR roles should not be mixed. Conceptually, ICs do not play the role of generating candidate models, and the regular rules should not play the role ascribed to ICs of filtering away undesired candidate models. IC rules make that filtering quite explicit and individualised, whereas such desired filtering, if coded into non-IC rules, say by means of OLONs, becomes enmeshed and diluted in them. Nevertheless, in a specific implementation, the evaluation of the two kinds of rules can be interspersed, and also for instance the goal $not\ \bot$ may be used to abduce solutions for IC satisfaction.

While the semantics of ICs in LPs is unambiguous and widely agreed upon, there are several alternative semantics for the regular part of NLPs. In order to prevent mixing the KR roles of ICs and non-ICs, the semantics for the regular normal rules must ensure they always have a model, leaving the task of discarding unwanted candidate models entirely to ICs. The current *de facto* standard in 2-valued semantics for NLPs, the Stable Models semantics, does allow a LP without explicit ICs to have no models at all, thus permitting regular rules to play in part the role of ICs. One can resort to a 3-valued semantics, like the Well-Founded Semantics (WFS Gelder et al. (1991)), which guarantees model existence for a LP, but not when staying within

2-valued setting is a requirement.
A question thus arises: what are the situations where the SMs semantics does not provide models, and what could those models be like? The answer to the question posed is not meant to find a replacement to the SMs semantics, but rather to find an alternative semantic construal that would warrant model existence, plus other desirable properties detailed below, namely cumulativity and relevancy. Delving into the reasons for the absence of models of SMs semantics, or properties it does not comply with, is not intended as a criticism of the latter but rather as a source of clues for defining a desired semantical alternative, not meant as its replacement.

It is well-known that the SM semantics does not enjoy certain properties, such as Relevancy and Cumulativity, which enable, respectively, the development of top-down proof-procedures for existential query-answering, and taking more advantage of tabling techniques to speed up computations. For practical applications, enabling such properties can mean a significant reduction in computation effort.
A second question thus arises: is there a 2-valued semantics that keeps the models SM semantics does provide, for one values its successes, and adds missing intended ones when one preserves separateness between regular rules and ICs? And which, moreover, guarantees model existence and also enjoys cumulativity and relevancy? The present work addresses these questions and answers them positively.

To start addressing these issues let us begin with a simple example.

Example 1. **A Joint Vacation Problem — Merging Logic Programs.** Three friends are planning a joint vacation. The first friend says "If we don't go to the mountains, then we should go to the beach". The second friend says "If we don't go to traveling, then we should go to the mountains". The third friend says "If we don't go to the beach, then we should go traveling". We code this information as the following NLP:

$$\begin{aligned} beach &\leftarrow not\ mountain \\ mountain &\leftarrow not\ travel \\ travel &\leftarrow not\ beach \end{aligned}$$

Each of these individual consistent rules come from a different friend. According to the SM semantics, each friend had a "solution" (a SM) for his own rule, but when one puts the three rules together the resulting merged logic program has no SM, because any atom in a purported model, or even none, leads to an inconsistency as per the semantics definition. This example aims to show the importance of having a 2-valued semantics guaranteeing model existence, in this case for the sake of arbitrary merging of individual logic programs (and for the sake of existence of a joint vacation for these three friends).

In contrast, the program

$$\begin{aligned} a &\leftarrow not\ b \\ b &\leftarrow not\ a \end{aligned}$$

has two SMs — $\{a\}$ and $\{b\}$ — despite both a and b having their rules involved in a mutual circular default negation as their unique means of support, as it was also the case with the three friends exampleabove. In both programs there is no atom whose

truth is well-founded, i.e. fully grounded on facts. This happens because all atoms are heads of rules involved in circular dependencies over default negation, their only means of support; but in the first (three friends) program there are no SMs whereas in the second (a, b even loop) there are two SMs. This asymmetry of SM semantics in assigning meaning to some *loops* over default negation and not to others stems from the classical intuitive meaning assigned to default negation: that is, of assuming the truth of the negative literal by unprovability or lack of support of the corresponding positive atom, and of not allowing the assumption of positive atoms. This, it shall be seen, is at the origin of the issues being addressed: the guarantee of model existence, and the relevancy and cumulativity properties.

Saying a regular rules NLP expresses inconsistent knowledge because it has no SMs is a semantics specific particular case of what inconsistent knowledge is. Uncontroversially, a NLP program is inconsistent iff all of its purported models would include an atom and its negation. (Trivially true when no models exist.)
Since the heads of regular rules in a NLP are always positive atoms, there is no danger of having classical contradictory conclusions since no default negated atoms can be drawn from the program; thus NLPs, absent ICs, are necessarily non-contradictory, no matter the kinds of loops over negation they contain. Loops over an odd number of DNLs, as in Example 1 above, are not intrinsically contradictory because they cannot be used to support negated conclusions. They are one other syntactical configuration of circular default negation, like even loops over default literals are.

To the effect of assigning an intuitive meaning to all configurations of circular default negation what first matters, both syntactically and semantically, is to identify all the subsets of rules in the NLP that give rise to such circularities. In the graph induced by the NLP, where one considers the NLP rules as the nodes, and a directed arc from rule r_1 to rule r_2 iff r_2 depends (indifferently via a positive atom, or a default negated literal) on r_1, then the subsets of rules of the NLP that form the circularities over default negation correspond to the Strongly Connected Components (SCCs) in such a graph. A Strongly Connected Component is a maximal strongly connected subgraph, i.e., a maximal subgraph such that there is a path from each vertex in the subgraph to every other vertex in the subgraph.

In order to assign a meaning to every regular NLP, i.e. to guarantee model existence, one needs to interpret default negation and the meaning of normal rules in a new, more general way: default negation is intuitively considered as arising from a minimal assumption (or abduction) of the truth of positive atoms appearing in default negated literals, which is sufficient to produce a 2-valued complete model; where rules are intuitively interpreted as a means to impose conditional truthfulness of atoms on the basis of the assumed ones. Mark that minimally assuming such positive atoms permits maximally assuming default literals consistent with the program.

The well-known notion of Closed-World Assumption (CWA) turns out to be a particular case of this more general intuitive meaning of default negation. And stratifiability ensures no assumptions of positive atoms are of any use at all (there being no need to break odd loops). While the CWA is usually seen as the maximisation of the assumption of negative literals, under our proposed intuitive meaning, it is

rendered "in reverse" as a particular case of the minimisation of the assumption of positive atoms. In particular, the CWA turns out to correspond to the assumption of no positive literals at all, which is possible only in programs with fully stratified default negation. We argue that the same principle of minimisation of assumed positive atoms of DNLs should be complied with in every 2-valued model of the NLP. Whenever a program contains non-stratified default negation some assumptions of positive atoms need to be taken up in every 2-valued model, i.e., one cannot demand an empty set of assumed positive atoms. In ELONs these correspond to alternatively assuming their corresponding default literals. In OLONs there is no such correspondence, so positive literal assumptions are inevitable.

Whenever an atom's unique means of support lies in circular default negation, its truthfulness in a model can only be justified if it is hypothesised — actually the SM semantics takes a similar approach but it can only deal with the cases where the circularity ranges over an even number of negations. In the even loop over a and b program above neither a nor b is necessarily *true* — their truthfulness is not fully grounded in facts, rather, they rely upon circular negation. The SMs in this program, $\{a\}$ and $\{b\}$, correspond to the particular hypothesisations of *not b* and *not a*, respectively, which happen to be *stable*, and therefore classically supported, according to the Gelfond-Lifschitz operator.

In this paper the authors take a hypothesisation approach too, which goes beyond the one taken by SMs, and uses criteria other than the Gelfond-Lifschitz stability one, for deciding whether or not to accept the hypothesised set. Namely one allows positive hypotheses, in a consistent and minimal way, as detailed in the sequel. E.g., in the three-friends example above, our approach allows for three alternative models (shown in full) in the following way: if one assumes *beach* is *true* then one cannot conclude $travel$ and therefore one concludes *mountain* is also *true* — this gives rise to the model $\{beach, mountain, not\ travel\}$, a joint and multi-place vacation solution. Symmetrically, the other two models are $\{mountain, not\ beach, travel\}$ and $\{travel, not\ mountain, beach\}$.

Arbitrary Updates and/or Merges

One of the main goals behind the conception of non-monotonic logics is the ability to deal with the changing, evolving, updating of knowledge. There are scenarios where it is possible and useful to combine several Knowledge Bases (possibly from different authors or sources) into a single one, and/or to update a given KB with new knowledge. Assuming the KBs are coded as IC-free NLPs, as well as the updates, the resulting KB is also an IC-free NLP. The lack of such a guarantee compromises the possibility of arbitrarily updating and/or merging KBs (coded as IC-free NLPs). In the case of self-updating programs, their desirable "liveness" property is put into question, even without outside intervention.

Intuitively Desired Semantics

The non-stratification of the default *not* is the fundamental ingredient which allows for the possible existence of several models for a program. The non-stratified (i.e., in a loop) DNLs of a program can thus be seen as non-deterministically assumable choices. The rules of the program impose truthfulness to some atoms of the program, thus constraining which sets of those choices are acceptable. Programs with OLONs (Ex. 1) are said to be "inconsistent" by the SMs semantics because the latter takes a negative hypotheses assumption approach, consistently maximizing them, i.e., DNLs are seen as assumable/abducible hypotheses. In Ex.1 though, assuming whichever negative hypothesis leads to the corresponding positive inconsistent conclusion via the rules. However, if the resent authors take the inverse approach, of consistent minimal *positive* hypotheses assumption (where the assumed hypotheses are the *atoms* of the DNLs), it is impossible to achieve a contradiction since no negative conclusions can be drawn from NLP rules. Minimizing positive assumptions implies the consistent maximizing of remaining negative ones but also gaining an extra degree of freedom.

Desirable Formal Properties

Only ICs (rules with $\bot$ head) should "endanger" model existence in a logic program, as has been argued above. Therefore, a semantics for NLPs with no ICs should guarantee the model existence property. Relevancy is also a useful property since it allows the development of top-down query-driven proof-procedures that allow for the sound and complete search for answers to a user's query. This is useful in the sense that in order to find an answer to a query only the relevant part of the program need be considered, the one provided by the query's call-graph, whereas in a non-relevant semantics the whole program must be considered, because model existence may be in jeopardy, with corresponding performance disadvantage compared to a relevant semantics. Moreover, since the whole program must be considered in a non-relevant semantics, complete models must be produced to warrant model existence, which in implementation terms requires the finite grounding of the program. Whereas in a relevancy obeying semantics the solution to a query may be just a partial model, guaranteed however to belong to some complete model.

Definition 1. Relevant part of P for atom a. The relevant part of NLP P for atom a is $Rel_P(a) = \{r_a \in P : head(r_a) = a\} \cup \{r \in P : \exists_{r_a \in P \wedge head(r_a)=a} r_a \twoheadleftarrow r\}$
where one writes $r_a \twoheadleftarrow r$ meaning rule r_a depends on rule r, possibly indirectly via other intermediate rules (cf. final note in Definition 5).

Definition 2. Relevancy (adapted from Dix (1995b)). A semantics *Sem* for logic programs is said Relevant iff for every program P an atom a belongs to every model M of P according to *Sem* iff a belongs to every model of the Relevant part of P to a according to *Sem*. Formally,

$$\forall_{a \in \mathscr{H}_P}(\forall_{M \in Models_{Sem}(P)} a \in M) \Leftrightarrow (\forall_{M_a \in Models_{Sem}(Rel_P(a))} a \in M_a)$$

Moreover, cumulativity also plays a role in performance enhancement in the sense that only a semantics enjoying this property can take advantage of storing intermediate lemmas to speed up future computations.

Definition 3. Cumulativity (adapted from Dix (1995a)). Let P be an NLP, and a, b two atoms of $\mathscr{H}_P$. A semantics *Sem* is Cumulative iff the semantics of P remains unchanged when any atom *true* in the semantics is added to P as a fact:

$$\forall_{a,b \in \mathscr{H}_P} \big((\forall_{M \in Models_{Sem}(P)} a \in M) \Rightarrow$$

$$(\forall_{M \in Models_{Sem}(P)} b \in M \Leftrightarrow \forall_{M_a \in Models_{Sem}(P \cup \{a\})} b \in M_a)\big)$$

Finally, each individual SM of a program, by being minimal and classically supported, should be accepted as a model according to every 2-valued semantics, and hence every 2-valued semantics should be a model conservative extension of the Stable Model semantics, which emphasises the importance of the latter.

Logic Programming Background

For self-containment the authors summarise below the basic background definitions and notations used.

Definition 4. Logic Program. By an alphabet $\mathscr{A}$ of a language $\mathscr{L}$ one means (finite or countably infinite) disjoint sets of constants, predicate symbols, and function symbols, with at least one constant. In addition, any alphabet is assumed to contain a countably infinite set of distinguished variable symbols. A term over $\mathscr{A}$ is defined recursively as either a variable, a constant or an expression of the form $f(t_1, \ldots, t_n)$ where f is a function symbol of $\mathscr{A}$, n its arity, and the t_i are terms. An atom over $\mathscr{A}$ is an expression of the form $P(t_1, \ldots, t_n)$ where P is a predicate symbol of $\mathscr{A}$, and the t_i are terms. A literal is either an atom A or its default negation *not* A. We dub default literals (or default negated literals — DNLs, for short) those of the form *not* A. A term (resp. atom, literal) is said ground if it does not contain variables. The set of all ground terms (resp. atoms) of $\mathscr{A}$ is called the Herbrand universe (resp. base) of $\mathscr{A}$. For short, $\mathscr{H}$ is used to denote the Herbrand base of $\mathscr{A}$. A Normal Logic Program (NLP) is a set of rules of the form

$$H \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m, \text{ (with } m, n \geq 0 \text{ and finite body)}$$

where H, the B_i and the C_j are atoms, and each rule stands for all its (possibly infinite) ground instances. In conformity with the standard convention, one write rules of the form $H \leftarrow$ also simply as H (known as "facts"). An NLP P is called definite if none of its rules contain default literals. H is the head of the rule r, denoted by $head(r)$, and $body(r)$ denotes the set $\{B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m\}$ of all the literals in the body of r.

When doing problem modeling with logic programs, rules of the form

$$\bot \leftarrow B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m, \text{ (with } m,n \geq 0 \text{ and finite body)}$$

with a non-empty body are known as a type of Integrity Constraints (ICs), specifically *denials*, and they are normally used to prune out unwanted candidate solutions. We abuse the '*not* ' default negation notation applying it to non-empty sets of literals too: one writes *not* S to denote $\{not\ s : s \in S\}$, and confound *not not* $a \equiv a$. When S is an arbitrary, non-empty set of literals $S = \{B_1, \ldots, B_n, not\ C_1, \ldots, not\ C_m\}$ the following notation is used:

- S^+ denotes the set $\{B_1, \ldots, B_n\}$ of positive literals in S
- S^- denotes the set $\{not\ C_1, \ldots, not\ C_m\}$ of negative literals in S
- $|S| = S^+ \cup (not\ S^-)$ denotes the set $\{B_1, \ldots, B_n, C_1, \ldots, C_m\}$ of atoms of S

As expected, a set of literals S is said consistent iff $S^+ \cap |S^-| = \emptyset$. We also write $heads(P)$ to denote the set of heads of non-IC rules of a (possibly constrained) program P, i.e., $heads(P) = \{head(r) : r \in P\} \setminus \{\bot\}$, and $facts(P)$ to denote the set of facts of P — $facts(P) = \{head(r) : r \in P \wedge body(r) = \emptyset\}$.

Definition 5. Part of body of a rule not in loop. Let P be an NLP and r a rule of P. We write $\overline{body(r)}$ to denote the subset of $body(r)$ whose atoms do not depend on r. Formally, $\overline{body(r)}$ is the largest set of literals such that

$$\overline{body(r)} \subseteq body(r) \wedge \forall_{a \in |\overline{body(r)}|} \nexists_{r_a \in P} (head(r_a) = a \wedge r_a \twoheadleftarrow r)$$

where $r_a \twoheadleftarrow r$ means rule r_a depends on rule r, i.e., either $head(r) \in |body(r_a)|$ or there is some other rule $r' \in P$ such that $r_a \twoheadleftarrow r'$ and $head(r) \in |body(r')|$.

Definition 6. Layer Supported and Classically supported interpretations. We say an interpretation I of an NLP P is layer (classically) supported iff every atom a of I is layer (classically) supported in I. a is layer (classically) supported in I iff there is some rule r in P with $head(r) = a$ such that $I \models \overline{body(r)}$ ($I \models body(r)$). Likewise, the rule r is said layer (classically) supported in I iff $I \models \overline{body(r)}$ ($I \models body(r)$).

Literals in $\overline{body(r)}$ are, by definition, not in loop with r. The notion of layered support requires that all such literals be *true* under I in order for $head(r)$ to be layer supported in I. Hence, if $\overline{body(r)}$ is empty, $head(r)$ is *ipso facto* layer supported.

Proposition 1. ***Classical Support implies Layered Support.*** *Given an NLP P, an interpretation I, and an atom a such that $a \in I$, if a is classically supported in I then a is also layer supported in I.*

Proof. Knowing that, by definition, $\overline{body(r)} \subseteq body(r)$ for every rule r, it follows trivially that a is layer supported in I if a is classically supported in I.

The intuitive semantic goal of overall minimality of assumed positive hypotheses implies, on the one hand, assuming no positive hypotheses at all in parts of the program which have stratified default negation and, on the other hand, allowing for the

assumption of non-empty minimal sets of positive hypotheses for the parts of the program with circular (non-stratified) negation. To the effect of semantically distinguishing those stratified/non-stratified cases one needs to syntactically identify such circular patterns since it is within the latter that lies the set of assumable hypotheses. We do so via the following program transformations.

Syntactic Transformations

Definite LPs (i.e., without default negation), have one 2-valued model which is its *least model*, which coincides with the Well-Founded Model (WFM Gelder et al. (1991)), because the absence of default negation leaves no room for hypotheses assumption. This is also the case for locally stratified LPs, i.e., the ones where all default negation is stratified (no circular rule dependencies via DNLs). In such cases one can use a syntactic transformation on a program to obtain that model. The authors have opted for program rewrite transformations that rest on well-known proven properties, making just slight modifications, rather than going for completely newly defined transformations, so as to facilitate the understanding of our changes by comparison with widely familiar ones.

In Brass et al. (2001) the authors defined the program Remainder (denoted by $\widehat{P}$) for calculating the WFM, of possibly infinite ground programs, which coincides with the unique perfect model for locally stratified LPs. The Remainder of P is the result of a program transformation which can be seen as a generalisation for NLPs of the least fixed point operator, $lfp(T)$, the latter utilised only for the subclass of definite LPs. We recap here the definitions necessary for the Remainder because they will be used in the definition of our Minimal Hypotheses semantics. The intuitive gist of MH semantics (formally defined in section) is as follows: an interpretation M_H is a MH model of program P iff there is some minimal set of hypotheses H such that the truth-values of all atoms of P become determined assuming the atoms in H as *true*. We resort to the program Remainder as a deterministic (and efficient, i.e., computable in polynomial time) means to find out if the truth-values of all literals become determined or not — below, one will see how the Remainder can be used to do so.

Program Remainder

For self-containment, are included here the definitions of Brass et al. (2001) upon which the Remainder relies, and adapt them where convenient to better match the syntactic conventions used throughout this paper.

Definition 7. Program transformation (def. 4.2 of Brass et al. (2001)). A *program transformation* is a relation $\mapsto$ between ground logic programs. A semantics S allows a transformation $\mapsto$ iff $Models_S(P_1) = Models_S(P_2)$ for all P_1 and P_2 with

$P_1 \mapsto P_2$. We write $\mapsto^*$ to denote the fixed point of the $\mapsto$ operation, i.e., $P \mapsto^* P'$ where $\nexists_{P'' \neq P'} P' \mapsto P''$. It follows that $P \mapsto^* P' \Rightarrow P' \mapsto P'$.

Definition 8. Positive reduction (def. 4.6 of Brass et al. (2001)). Let P_1 and P_2 be ground programs. Program P_2 results from P_1 by *positive reduction* ($P_1 \mapsto_P P_2$) iff there is a rule $r \in P_1$ and a negative literal *not* $b \in body(r)$ such that $b \notin heads(P_1)$, i.e., there is no rule for b in P_1, and $P_2 = (P_1 \setminus \{r\}) \cup \{head(r) \leftarrow (body(r) \setminus \{not\ b\})\}$.

Definition 9. Negative reduction (def. 4.7 of Brass et al. (2001)). Let P_1 and P_2 be ground programs. Program P_2 results from P_1 by *negative reduction* ($P_1 \mapsto_N P_2$) iff there is a rule $r \in P_1$ and a negative literal *not* $b \in body(r)$ such that $b \in facts(P_1)$, i.e., b appears as a fact in P_1, and $P_2 = P_1 \setminus \{r\}$.

According to our proposed meaning of rules and default negation, circular rule dependencies over DNLs are the sources of assumable hypotheses. Therefore, one needs a program transformation similar to the Negative reduction above, but one preserving the loops over negation in order to keep the sources of hypotheses; as a consequence, now is introduced a layered version of the negative reduction operation.

Definition 10. Layered negative reduction. Let P_1 and P_2 be ground programs. Program P_2 results from P_1 by *layered negative reduction* ($P_1 \mapsto_{LN} P_2$) iff there is a rule $r \in P_1$ and a negative literal *not* $b \in \overline{body(r)}$ such that $b \in facts(P_1)$, i.e., b appears as a fact in P_1, and $P_2 = P_1 \setminus \{r\}$.

The Strongly Connected Components (SCCs) of rules of a program can be calculated in polynomial time, Tarjan (1972). Once the SCCs of rules have been identified, the $\overline{body(r)}$ subset of $body(r)$, for each rule r, is identifiable in linear time — one needs to check just once for each literal in $body(r)$ if it is also in $\overline{body(r)}$. Therefore, these polynomial time complexity operations are all the added complexity Layered negative reduction adds over regular Negative reduction.

Definition 11. Success (def. 5.2 of Brass et al. (2001)). Let P_1 and P_2 be ground programs. Program P_2 results from P_1 by *success* ($P_1 \mapsto_S P_2$) iff there is a rule $r \in P_1$ and a fact $b \in facts(P_1)$ such that $b \in body(r)$, and $P_2 = (P_1 \setminus \{r\}) \cup \{head(r) \leftarrow (body(r) \setminus \{b\})\}$.

Definition 12. Failure (def. 5.3 of Brass et al. (2001)). Let P_1 and P_2 be ground programs. Program P_2 results from P_1 by *failure* ($P_1 \mapsto_F P_2$) iff there is a rule $r \in P_1$ and a positive literal $b \in body(r)$ such that $b \notin heads(P_1)$, i.e., there are no rules for b in P_1, and $P_2 = P_1 \setminus \{r\}$.

Definition 13. Loop detection (def. 5.10 of Brass et al. (2001)). Let P_1 and P_2 be ground programs. Program P_2 results from P_1 by *loop detection* ($P_1 \mapsto_L P_2$) iff there is a set $\mathscr{A}$ of ground atoms such that

1. for each rule $r \in P_1$, if $head(r) \in \mathscr{A}$, then $body(r) \cap \mathscr{A} \neq \emptyset$,

2. $P_2 := \{r \in P_1 | body(r) \cap \mathscr{A} = \emptyset\}$,
3. $P_1 \neq P_2$.

We do not enter here into the details of the *loop detection* step, but just make note that 1) such a set $\mathscr{A}$ corresponds to an unfounded set (cf. Gelder et al. (1991)); 2) loop detection is computationally equivalent to finding the SCCs, Tarjan (1972), and is known to be of polynomial time complexity; and 3) the atoms in the unfounded set $\mathscr{A}$ have all their corresponding rules involved in SCCs where all heads of rules in a loop appear positive in the bodies of the rules in the loop.

Definition 14. Reduction (def. 5.15 of Brass et al. (2001)).
Let $\mapsto_X$ denote the rewriting system: $\mapsto_X := \mapsto_P \cup \mapsto_N \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L$.

Definition 15. Layered reduction.
Let $\mapsto_{LX}$ denote the rewriting system: $\mapsto_{LX} := \mapsto_P \cup \mapsto_{LN} \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L$.

Definition 16. Remainder (def. 5.17 of Brass et al. (2001)). Let P be a program. Let $\widehat{P}$ satisfy $ground(P) \mapsto_X^* \widehat{P}$. Then $\widehat{P}$ is called the *Remainder* of P, and is guaranteed to exist and to be unique to P. Moreover, the calculus of $\mapsto_X^*$ is known to be of polynomial time complexity, Brass et al. (2001).

An important result from Brass et al. (2001) is that the WFM of P is such that $WFM^+(P) = facts(\widehat{P})$, $WFM^{+u} = heads(\widehat{P})$, and $WFM^-(P) = \mathscr{H}_P \setminus WFM^{+u}(P)$, where $WFM^+(P)$ denotes the set of atoms of P *true* in the WFM, $WFM^{+u}(P)$ denotes the set of atoms of P *true* or *undefined* in the WFM, and $WFM^-(P)$ denotes the set of atoms of P *false* in the WFM.

We define the layered counterpart of the Remainder, the Layered Remainder, as follows:

Definition 17. Layered Remainder. Let P be a program. Let the program $\mathring{P}$ satisfy $ground(P) \mapsto_{LX}^* \mathring{P}$. Then $\mathring{P}$ is called a *layered remainder* of P. Since $\mapsto_{LX}$ differs from $\mapsto_X$ only because it uses $\mapsto_{LN}$ instead of $\mapsto_N$, it follows trivially that $\mathring{P}$ is also guaranteed to exist and be unique for P. Moreover, the calculus of $\mapsto_{LX}^*$ is likewise of polynomial time complexity because $\mapsto_{LN}$ is also of polynomial time complexity.

The remainder's rewrite rules are provably confluent, i.e. independent of application order, as the layered remainder's rules differ only in the negative reduction rule and the confluence proof of the former is readily adapted to the latter.

Example 2. $\mathring{P}$ **versus** $\widehat{P}$**.** Recall the program from Example 1 but now with an additional fourth stubborn friend who insists on going to the beach no matter what.
$P =$

$$\begin{aligned} beach &\leftarrow not\ mountain \\ mountain &\leftarrow not\ travel \\ travel &\leftarrow not\ beach \\ beach & \end{aligned}$$

The single fact rule does not depend on any other, and the remaining three rules forming the loop all depend on each other and on the fact rule *beach*. $\widehat{P}$ is the fixed

point of $\mapsto_X$, i.e., the fixed point of $\mapsto_P \cup \mapsto_N \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L$. Since *beach* is a fact, the $\mapsto_N$ transformation deletes the $travel \leftarrow not\ beach$ rule; i.e., $P \mapsto_N P'$ is such that $P' = \{beach \leftarrow not\ mountain \quad mountain \leftarrow not\ travel \quad beach \leftarrow\}$. Now in P' there are no rules for *travel* and hence one can apply the $\mapsto_P$ transformation which deletes the $not\ travel$ from the body of *mountain*'s rule; i.e, $P' \mapsto_P P''$ where $P'' = \{beach \leftarrow not\ mountain \quad mountain \leftarrow \quad beach \leftarrow\}$. Finally, in P'' *mountain* is a fact and hence one can again apply the $\mapsto_N$ obtaining $P'' \mapsto_P P'''$ where $P''' = \{mountain \leftarrow \quad beach \leftarrow\}$ upon which no more transformations can be applied, so $\widehat{P} = P'''$.

Instead, the Layered negative reduction does not allow the deletion of the rule $travel \leftarrow not\ beach$ on the account of the fact *beach* because there is another rule $beach \leftarrow not\ mountain$ that (indirectly through $mountain \leftarrow not\ travel$) depends on *travel* thus forming a loop over negation. For this reason no transformations are applicable in the calculation of the Layered remainder and one obtains $\mathring{P} = P$, which is the fixed point of $\mapsto_{LX}$, i.e., the fixed point of $\mapsto_P \cup \mapsto_{LN} \cup \mapsto_S \cup \mapsto_F \cup \mapsto_L$.

Minimal Hypotheses Semantics

The Abductive Flavour of Assuming Hypotheses

The abductive perspective of Kakas et al. (1992) depicts DNLs as abducibles, i.e., assumable hypotheses. Indeed, DNLs can be considered as assumable hypotheses, but not all of them. As explained before, only those DNLs involved in SCCs are eligible. Strongly Connected Components, as in Examples 1 and 2.

Both the SMs and the approach of Kakas et al. (1992), when taking the abductive perspective, adopt negative hypotheses only. This approach works fine for some instances of non-well-founded negation such as loops (in particular, for even loops over negation like the one over *not a* and *not b* just after Example 1), but not for odd loops over negation like, e.g. $a \leftarrow not\ a$: assuming *not a* would lead to the conclusion that *a* is *true* which contradicts the initial assumption.

To overcome this problem, the authors generalise the hypotheses assumption perspective to allow the adoption, not only of negative hypotheses, but also of positive ones. Having taken this generalisation step one realises that the positive hypotheses assumption alone is sufficient to address all situations, i.e., there is no need for both positive and negative hypotheses assumptions. Indeed, because one minimises the positive hypotheses, with one stroke one maximises the negative ones, which has been the traditional way of dealing with the CWA, and also comply with the stable models stance, because the latter's requirement of support is conducive to classically minimal models.

In Example 1 one saw three solutions, each assuming as *true* one of the DNLs in the loop. Adding a fourth stubborn friend insisting on going to the beach, as in Example 2, should still permit the two full solutions $\{beach, mountain, not\ travel\}$

and $\{travel, not\ mountain, beach\}$. The only way to permit both these solutions is by resorting to the Layered Remainder, and not to the Remainder, as a means to identify the set of assumable hypotheses. Thus, all the atoms of DNLs of P that are not determined *false* in $\mathring{P}$ are candidates for the role of hypotheses one may consider to assume as *true*. Merging this perspective with the abductive one of Kakas et al. (1992) (where the DNLs are the abducibles) one comes to the following definition of the Hypotheses set of a program.

Definition 18. Hypotheses set of a program. Let P be an NLP. We write $Hyps(P)$ to denote the set of assumable hypotheses of P: the atoms that appear as DNLs in the bodies of rules of $\mathring{P}$. Formally, $Hyps(P) = \{a : \exists_{r \in \mathring{P}} not\ a \in body(r)\}$.

One can define a classical support compatible version of the Hypotheses set of a program, only using to that effect the Remainder instead of the Layered Remainder. I.e.,

Definition 19. Classical Hypotheses set of a program. Let P be an NLP. We write $CHyps(P)$ to denote the set of assumable hypotheses of P consistent with the classical notion of support: the atoms that appear as DNLs in the bodies of rules of $\widehat{P}$. Formally, $CHyps(P) = \{a : \exists_{r \in \widehat{P}} not\ a \in body(r)\}$.

Here the authors take the layered support compatible approach and, therefore, will use the Hypotheses set as in definition 18. Since $CHyps(P) \subseteq Hyps(P)$ for every NLP P, there is no loss of generality in using $Hyps(P)$ instead of $CHyps(P)$, while using $Hyps(P)$ allows for some useful semantics properties examined in the sequel.

Definition of Minimal Hypotheses Model

Intuitively, a Minimal Hypotheses model of a program is obtained from a minimal set of hypotheses which is sufficiently large to determine the truth-value of all literals via the Remainder.

Definition 20. Minimal Hypotheses model. Let P be an NLP. Let $Hyps(P)$ be the set of assumable hypotheses of P (cf. definition 18), and H some subset of $Hyps(P)$.

A 2-valued model M of P is a Minimal Hypotheses model of P iff

$$M^+ = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H})$$

where $H = \emptyset$ or H is non-empty set-inclusion minimal (i.e. the set-inclusion minimality is considered only for non-empty Hs). I.e., the hypotheses set H is minimal but sufficient to determine (via Remainder) the truth-value of all literals in the program.

We already know that $WFM^+(P) = facts(\widehat{P})$ and that $WFM^{+u}(P) = heads(\widehat{P})$. Thus, whenever $facts(\widehat{P}) = heads(\widehat{P})$ one has $WFM^+(P) = WFM^{+u}(P)$ which means $WFM^u(P) = \emptyset$. Moreover, whenever $WFM^u(P) = \emptyset$ one knows, by Corollary 5.6 of Gelder et al. (1991), that the 2-valued model M such that $M^+ = facts(\widehat{P})$

is the unique stable model of P. Also, it follows that every SM of P is a Minimal Hypotheses model of P.

In example 2 one can thus see that one has the two full models $\{beach, mountain, not\ travel\}$ and $\{travel, beach, not\ mountain\}$. This is the case because the addition of the fourth stubborn friend does not change the set of $Hyps(P)$, which is based upon the Layered Remainder and not on the Remainder.

Example 3. **Minimal Hypotheses models for the vacation with passport variation.** Consider again the vacation problem from Example 1, now with a variation including the need for valid passports for traveling $P =$

$$\begin{aligned} beach &\leftarrow not\ mountain \\ mountain &\leftarrow not\ travel \\ travel &\leftarrow not\ beach, not\ expired\_passport \\ \\ passport\_ok &\leftarrow not\ expired\_passport \\ expired\_passport &\leftarrow not\ passport\_ok \end{aligned}$$

We have $P = \mathring{P} = \widehat{P}$ and thus $Hyps(P) = \{beach, mountain, travel, passport\_ok, expired\_passport\}$. Let us see which are the MH models for this program.
$H = \emptyset$ does not yield a MH model.
Assuming $H = \{beach\}$ one has $P \cup H = P \cup \{beach\} =$

$$\begin{aligned} beach &\leftarrow not\ mountain \\ mountain &\leftarrow not\ travel \\ travel &\leftarrow not\ beach, not\ expired\_passport \\ beach & \\ passport\_ok &\leftarrow not\ expired\_passport \\ expired\_passport &\leftarrow not\ passport\_ok \end{aligned}$$

and $\widehat{P \cup H} =$

$$\begin{aligned} mountain & \\ beach & \\ passport\_ok &\leftarrow not\ expired\_passport \\ expired\_passport &\leftarrow not\ passport\_ok \end{aligned}$$

which means $H = \{beach\}$ is not sufficient to determine the truth values of all literals of P. One can easily see that the same happens for $H = \{mountain\}$ and for $H = \{travel\}$: in either case the literals *passport_ok* and *expired_passport* remain non-determined.
If assuming $H = \{expired\_passport\}$ then $P \cup H$ is

$$
\begin{aligned}
beach &\leftarrow not\ mountain \\
mountain &\leftarrow not\ travel \\
travel &\leftarrow not\ beach, not\ expired\_passport \\
\\
passport\_ok &\leftarrow not\ expired\_passport \\
expired\_passport &\leftarrow not\ passport\_ok \\
expired\_passport &
\end{aligned}
$$

and $\widehat{P \cup H} =$

$$
\begin{aligned}
mountain \\
expired\_passport
\end{aligned}
$$

which means $M^{+}_{expired\_passport} = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H}) = \{mountain, expired\_passport\}$, i.e., $M_{expired\_passport} = \{not\ beach, mountain, not\ travel, not\ passport\_ok, expired\_passport\}$, is a MH model of P. Since assuming $H = \{expired\_passport\}$ alone is sufficient to determine all literals, there is no other set of hypotheses H' of P such that $H' \supset \{expired\_passport\}$ (notice the *strict* $\supset$, not $\supseteq$), yielding a MH model of P. E.g., $H' = \{travel, expired\_passport\}$ does not lead to a MH model of P simply because H' is not minimal w.r.t. $H = \{expired\_passport\}$.
If assuming $H = \{passport\_ok\}$ then $P \cup H$ is

$$
\begin{aligned}
beach &\leftarrow not\ mountain \\
mountain &\leftarrow not\ travel \\
travel &\leftarrow not\ beach, not\ expired\_passport \\
\\
passport\_ok &\leftarrow not\ expired\_passport \\
expired\_passport &\leftarrow not\ passport\_ok \\
passport\_ok &
\end{aligned}
$$

and $\widehat{P \cup H} =$

$$
\begin{aligned}
beach &\leftarrow not\ mountain \\
mountain &\leftarrow not\ travel \\
travel &\leftarrow not\ beach \\
passport\_ok &
\end{aligned}
$$

which, apart from the fact *passport_ok*, corresponds to the original version of this example (1) and still leaves literals with non-determined truth-values. I.e., assuming the passports are OK allows for the three possibilities of Example 1 but it is not enough to entirely "solve" the vacation problem: one needs some hypotheses set containing one of *beach*, *mountain*, or *travel* if (in this case, *and only if*) it also contains *passport_ok*.

Example 4. **Minimality of Hypotheses does not guarantee minimality of model.** Let P, with no SMs, be

$$a \leftarrow not\ b, c$$
$$b \leftarrow not\ c, not\ a$$
$$c \leftarrow not\ a, b$$

In this case $P = \widehat{P} = \mathring{P}$, which makes $Hyps(P) = \{a,b,c\}$.
$H = \emptyset$ does not determine all literals of P because $facts(\widehat{P \cup \emptyset}) = facts(\widehat{P}) = \emptyset$ and $heads(\widehat{P \cup \emptyset}) = heads(\widehat{P}) = \{a,b,c\}$.
$H = \{a\}$ does determine all literals of P because $facts(\widehat{P \cup \{a\}}) = \{a\}$ and $heads(\widehat{P \cup \{a\}}) = \{a\}$, thus yielding the MH model M_a such that $M_a^+ = facts(\widehat{P \cup \{a\}})$ $= \{a\}$, i.e., $M_a = \{a, not\ b, not\ c\}$.
$H = \{c\}$ is also a minimal set of hypotheses determining all literals because $facts(\widehat{P \cup \{c\}}) = \{a,c\}$ and $heads(\widehat{P \cup \{c\}}) = \{a,c\}$, thus yielding the MH model M_c of P such that $M_c^+ = facts(\widehat{P \cup \{c\}}) = \{a,c\}$, i.e., $M_c = \{a, not\ b, c\}$. However, M_c is not a minimal model of P because $M_c^+ = \{a,c\}$ is a strict superset of $M_a^+ = \{a\}$. M_c is indeed an MH model of P, but just not a minimal model thereby being a clear example of how minimality of hypotheses does not entail minimality of consequences. Just to make this example complete, it is shown that $H = \{b\}$ also determines all literals of P because $facts(\widehat{P \cup \{b\}}) = \{b,c\}$ and $heads(\widehat{P \cup \{b\}}) = \{b,c\}$, thus yielding the MH model M_b such that $M_b^+ = facts(\widehat{P \cup \{b\}}) = \{b,c\}$, i.e., $M_b = \{not\ a, b, c\}$. Any other hypotheses set is necessarily a strict superset of either $H = \{a\}$, $H = \{b\}$, or $H = \{c\}$ and, therefore, not set-inclusion minimal; i.e., there are no more MH models of P.

Also, not all minimal models of a program are MH models, as the following example shows.

Example 5. **Some minimal models are not Minimal Hypotheses models.** Let P be

$$a \leftarrow k$$
$$k \leftarrow not\ t$$
$$t \leftarrow a, b$$
$$a \leftarrow not\ b$$
$$b \leftarrow not\ a$$

In this case $P = \widehat{P} = \mathring{P}$ and therefore $Hyps(P) = \{a,b,t\}$. Since $facts(\widehat{P}) \neq heads(\widehat{P})$, the hypotheses set $H = \emptyset$ does not yield a MH model. Assuming $H = \{a\}$ one has $\widehat{P \cup H} = \widehat{P \cup \{a\}} = \{a \leftarrow \quad , \quad k \leftarrow\}$ so, $\widehat{P \cup H}$ is the set of facts $\{a,k\}$ and, therefore, M_a such that $M_a^+ = facts(\widehat{P \cup H}) = facts(\widehat{P \cup \{a\}}) = \{a,k\}$, is a MH model of P. Assuming $H = \{b\}$ one has $\widehat{P \cup \{b\}} =$

$$a \leftarrow k$$
$$k \leftarrow not\ t$$
$$t \leftarrow a$$
$$b \leftarrow not\ a$$
$$b$$

thus $facts(\widehat{P \cup \{b\}}) = \{b\} \neq heads(\widehat{P \cup \{b\}}) = \{a,b,t,k\}$, which means the set of hypotheses $H = \{b\}$ does not yield a MH model of P. Assuming $H = \{t\}$ one has $\widehat{P \cup \{t\}} =$

$$\begin{array}{l} t \leftarrow a, b \\ b \leftarrow not\ a \\ a \leftarrow not\ b \\ t \end{array}$$

thus $facts(\widehat{P \cup \{t\}}) = \{t\} \neq heads(\widehat{P \cup \{t\}}) = \{a,b,t\}$, which means the set of hypotheses $H = \{t\}$ does not yield a MH model of P.
Since one already knows that $H = \{a\}$ yields an MH model M_a with $M_a^+ = \{a,k\}$, there is no point in trying out any subset H' of $Hyps(P) = \{a,b,t\}$ such that $a \in H'$ because any such subset would not be minimal w.r.t. $H = \{a\}$. Let us, therefore, move on to the unique subset left: $H = \{b,t\}$. Assuming $H = \{b,t\}$ one has $\widehat{P \cup \{b,t\}} = \{t \leftarrow, b \leftarrow\}$ thus $facts(\widehat{P \cup \{b,t\}}) = \{b,t\} = heads(\widehat{P \cup \{b,t\}})$, which means $M_{b,t}$ such that $M_{b,t}^+ = facts(\widehat{P \cup H}) = facts(\widehat{P \cup \{b,t\}}) = \{b,t\}$, is a MH model of P.
It is important to remark that this program has other classical models, e.g, $\{b,t\}$, and $\{a,t\}$, but $\{a,t\}$ is not an MH model because it is obtainable only via the set of hypotheses $\{a,t\}$ which is non-minimal w.r.t. $H = \{a\}$ that yields the MH model $\{a,k\}$.

There is also another class of regular rules NLPs, without any circular dependencies among rules, that have no SM either: the ones with infinite descending chains of rules. In practice, all useful programs have a finite number of dependencies between rules, but for theoretical completeness the authors show that the MH semantics can also assign a model to such programs.

A typical case of such a program (representing a whole class of programs with real theoretical interest) has an infinitely long *descending* chain of dependencies and was presented by François Fages in Fages (1994). We repeat it here for illustration and explanation.

Example 6. **Program with infinite descending chain Fages (1994).** Let P be

$$p(X) \leftarrow p(s(X)) \qquad\qquad p(X) \leftarrow not\ p(s(X))$$

The ground version of this program P, assuming there is only one constant 0 (zero), is:

$$\begin{array}{cc} p(0) \leftarrow p(s(0)) & p(0) \leftarrow not\ p(s(0)) \\ p(s(0)) \leftarrow p(s(s(0))) & p(s(0)) \leftarrow not\ p(s(s(0))) \\ p(s(s(0))) \leftarrow p(s(s(s(0)))) & p(s(s(0))) \leftarrow not\ p(s(s(s(0)))) \\ \vdots \leftarrow \vdots & \vdots \leftarrow \vdots \end{array}$$

The only model of this program is $\{p(0), p(s(0)), p(s(s(0)))\ldots\}$ or, in a non-ground form, $\{p(X)\}$. This program is of theoretical interest for two reasons: 1) it has an infinite *descending* chain of dependencies, and 2) it has no Stable Models, even though it has no loops, which shows a whole other class of NLPs to which the SMs

semantics provides no model. I.e., if a semantics for NLPs is to provide a model for each and every NLP it must cater for transfinite support chains, classically considered non-well-founded (cf. Fages (1994)). For theoretical completeness, whenever it is possible to determine if any two rules syntactically depend on each other (thus being part of an SCC), one has just one MH model. In the example above one has $\mathring{P} = P$ and thus $Hyps(P) = p(s(0)), p(s(s(0))), p(s(s(s(0)))), \ldots$ — notice $p(0)$ is not a hypothesis. The unique MH model is $\{p(0), p(s(0)), p(s(s(0))), \ldots\}$ obtained with the subset H of $Hyps(P)$ such that $H = \bigcap US_{Hyps(P)}$, where $US_{Hyps(P)}$ is an upper set of $Hyps(P)$ with $p(s^i(0)) \leq p(s^j(0))$ iff $i \leq j$. An upper set of a partially ordered set $(S, \leq)$ is a subset U with the property that, if x is in U and $x \leq y$, then y is in U. In Section the authors will present a program transformation that allows us to use currently available SM implementations to compute the MH models of the original program. While it can theoretically be done, such transformation, as it is, does not cater for programs like the one above with the infinite descending chain, by virtue of not including the computation of upper sets of $Hyps(P)$, and in that sense it is on par with the SM semantics.

Model Existence

The minimality of H is not sufficient to ensure minimality of $M^+ = facts(\widehat{P \cup H})$, making its checking explicitly necessary if that is so desired. Minimality of hypotheses is indeed the common practice in science, not the minimality of their inevitable consequences. To the contrary, the more of these the better because it signifies a greater predictive power.

In Logic Programming model minimality is a consequence of definitions: the T operator in definite programs is conducive to defining a least fixed point, a unique minimal model semantics; in SM, though there may be more than one model, minimality turns out to be a property because of the stability (and its attendant classical support) requirement; in the WFS, again the existence of a least fixed point operator affords a minimal (information) model. In abduction too, minimality of consequences is not a caveat, but minimality of hypotheses is, and is not a caveat even when the latter is not adopted. Maximal consequences from minimal hypotheses may be preferred instead. Hence our approach to LP semantics via MH-semantics is novel indeed, and insisting rather on positive hypotheses establishes an improved and more general link to abduction and argumentation Pereira and Pinto (2007a,b).

Theorem 1. ***At least one Minimal Hypotheses model of*** *P* ***complies with the Well-Founded Model.*** *Let P be an NLP. Then, there is at least one Minimal Hypotheses model M of P such that* $M^+ \supseteq WFM^+(P)$ *and* $M^+ \subseteq WFM^{+u}(P)$.

Proof. If $facts(\widehat{P}) = heads(\widehat{P})$ or equivalently, $WFM^u(P) = \emptyset$, then M_H is a MH model of P given that $H = \emptyset$ because $M_H^+ = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H}) = facts(\widehat{P \cup \emptyset}) = heads(\widehat{P \cup \emptyset}) = facts(\widehat{P}) = heads(\widehat{P})$. On the other hand, if $facts(\widehat{P}) \neq$

$heads(\widehat{P})$, then there is at least one non-empty set-inclusion minimal set of hypotheses $H \subseteq Hyps(P)$ such that $H \supseteq facts(P)$. The corresponding M_H is, by definition, a MH model of P which is guaranteed to comply with $M_H^+ \supseteq WFM^+(P) = facts(\widehat{P})$ and $M_H^- \supseteq not\ WFM^-(P) = not\ (\mathscr{H}_P \setminus M_H^+)$.

Theorem 2. ***Minimal Hypotheses semantics guarantees model existence.*** *Let P be an NLP. There is always, at least, one Minimal Hypotheses model of P.*

Proof. It is trivial to see that one can always find a set $H \subseteq Hyps(P)$ such that $M_{H'}^+ = facts(\widehat{P \cup H'}) = heads(\widehat{P \cup H'})$ — in the extreme case, $H' = Hyps(P)$. From such H' one can always select a minimal subset $H \subset H$ such that $M_H^+ = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H})$ still holds.

Relevancy

Theorem 3. ***Minimal Hypotheses semantics enjoys Relevancy.*** *Let P be an NLP. Then, by definition 2, it holds that*

$$(\forall_{M \in Models_{MH}(P)} a \in M^+) \Leftrightarrow (\forall_{M_a \in Models_{MH}(Rel_P(a))} a \in M_a^+)$$

Proof. $\Rightarrow$: Assume $\forall_{M \in Models_{MH}(P)} a \in M^+$. Now one needs to prove $\forall_{M_a \in Models_{MH}(Rel_P(a))} a \in M_a^+$. Assume some $M_a \in Models_{MH}(Rel_P(a))$; now one shows that assuming $a \notin M_a^+$ leads to an absurdity. Since M_a is a 2-valued complete model of $Rel_P(a)$ one knows that $|M_a| = \mathscr{H}_{Rel_P(a)}$ hence, if $a \notin M_a$, then necessarily $not\ a \in M_a^-$. Since $P \supseteq Rel_P(a)$, by theorem 2 one knows that there is some model M' of P such that $M' \supseteq M_a$, and thus $not\ a \in M'^-$ which contradicts the initial assumption that $\forall_{M \in Models_{MH}(P)} a \in M^+$. We conclude $a \notin M_a$ cannot hold, i.e., $a \in M_a$ must hold. Since $a \in M^+$ hold for every model M of P, then $a \in M_a$ must hold for every model M_a of $Rel_P(a)$.
$\Leftarrow$: Assume $\forall_{M_a \in Models_{MH}(Rel_P(a))} a \in M_a^+$. Now one needs to prove $\forall_{M \in Models_{MH}(P)} a \in M^+$. Let us write $P_{)a(}$ as an abbreviation of $P \setminus Rel_P(a)$. We have therefore $P = P_{)a(} \cup Rel_P(a)$. Let us now take $P_{)a(} \cup M_a$. We know that every NLP as an MH model, hence every MH model M of $P_{)a(} \cup M_a$ is such that $M \supseteq M_a$. Let H_{M_a} denote the Hypotheses set of M_a — i.e., $M_a^+ = facts(\widehat{Rel_P(a) \cup H_{M_a}}) = heads(\widehat{Rel_P(a) \cup H_{M_a}})$, with $H_{M_a} = \emptyset$ or non-empty set-inclusion minimal, as per definition 20. If $facts(\widehat{P \cup H_{M_a}}) = heads(\widehat{P \cup H_{M_a}})$ then $M^+ = facts(\widehat{P \cup H_M}) = heads(\widehat{P \cup H_M})$ is an MH model of P with $H_M = H_{M_a}$ and, necessarily, $M \supseteq M_a$.

If $facts(\widehat{P \cup H_{M_a}}) \neq heads(\widehat{P \cup H_{M_a}})$ then, knowing that every program has a MH model, one can always find an MH model M of $P_{)a(} \cup M_a$, with $H' \subseteq Hyps(P_{)a(} \cup M_a)$, where $M^+ = facts(\widehat{P \cup H'}) = heads(\widehat{P \cup H'})$. Such M is thus $M^+ = facts(\widehat{P \cup H_M}) = heads(\widehat{P \cup H_M})$ where $H_M = H_{M_a} \cup H'$, which means M

is a MH model of P with $M \supseteq M_a$. Since every model M_a of $Rel_P(a)$ is such that $a \in M_a^+$, then every model M of P must also be such that $a \in M$.

Cumulativity

MH semantics enjoys Cumulativity, thus allowing for lemma storing techniques to be used during computation of answers to queries.

Theorem 4. ***Minimal Hypotheses semantics enjoys Cumulativity.*** *Let P be an NLP. Then*

$$\forall_{a,b \in \mathscr{H}_\mathscr{P}} \big((\forall_{M \in Models_{MH}(\mathscr{P})} a \in M^+) \Rightarrow (\forall_{M \in Models_{MH}(\mathscr{P})} b \in M^+ \Leftrightarrow \forall_{M_a \in Models_{MH}(\mathscr{P} \cup \{a\})} b \in M_a^+) \big)$$

Proof. Assume $\forall_{\substack{a \in \mathscr{H}_\mathscr{P} \\ M \in Models_{MH}(\mathscr{P})}} a \in M^+$.

$\Rightarrow$: Assume $\forall_{M \in Models_{MH}(\mathscr{P})} b \in M^+$. Since every MH model M contains a it follows that all such M are also MH models of $P \cup \{a\}$. Since one assumed $b \in M$ as well, and one knows that M is a MH model of $P \cup \{a\}$ one concludes b is also in those MH models M of $P \cup \{a\}$. By adding a as a fact one has necessarily $Hyps(P \cup \{a\}) \subseteq Hyps(P)$ which means that there cannot be more MH models for $P \cup \{a\}$ than for P. Since one already knows that for every MH model M of P, M is also a MH model of $P \cup \{a\}$ one must conclude that $\forall_{M \in Models_{MH}(P)} \exists^1_{M' \in Models_{MH}(P \cup \{a\})}$ such that $M'^+ \supseteq M^+$. Since $\forall_{M \in Models_{MH}(\mathscr{P})} b \in M^+$ one necessarily concludes

$$\forall_{M_a \in Models_{MH}(\mathscr{P} \cup \{a\})} b \in M_a^+$$

$\Leftarrow$: Assume $\forall_{M_a \in Models_{MH}(\mathscr{P} \cup \{a\})} b \in M_a^+$. Since the MH semantics is relevant (theorem 3) if b does not depend on a then adding a as a fact to P or not has no impact on b's truth-value, and if $b \in M_a^+$ then $b \in M^+$ as well. If b does depend on a, which is true in every MH model M of P, then either 1) b depends positively on a, and in this case since $a \in M$ then $b \in M$ as well; or 2) b depends negatively on a, and in this case the lack of a as a fact in P can only contribute, if at all, to make b true in M as well. Then one concludes $\forall_{M \in Models_{MH}(\mathscr{P})} b \in M^+$.

Complexity

The complexity issues usually relate to a particular set of tasks, namely: 1) knowing if the program has a model; 2) if it has any model entailing some set of ground literals (a query); 3) if all models entail a set of literals. In the case of MH semantics, the answer to the first question is an immediate "yes" because MH semantics guarantees model existence for NLPs; the second and third questions correspond (respectively) to Brave and Cautious Reasoning, which the authors now analyse.

Brave Reasoning

The complexity of the Brave Reasoning task with MH semantics, i.e., finding an MH model satisfying some particular set of literals is Σ_2^P-complete.

Theorem 5. ***Brave Reasoning with MH semantics is Σ_2^P-complete.*** *Let P be an NLP, and Q a set of literals, or* query. *Finding an MH model such that $M \supseteq Q$ is a Σ_2^P-complete task.*

Proof. To show that finding a MH model $M \supseteq Q$ is Σ_2^P-complete, note that a nondeterministic Turing machine with access to an NP-complete oracle can solve the problem as follows: nondeterministically guess a set H of hypotheses (i.e., a subset of $Hyps(P)$). It remains to check if H is empty or non-empty minimal such that $M^+ = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H})$ and $M \supseteq Q$. Checking that $M^+ = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H})$ can be done in polynomial time (because computing $\widehat{P \cup H}$ can be done in polynomial time, Brass et al. (2001), for whichever $P \cup H$), and checking H is empty or non-empty minimal requires a nondeterministic guess of a strict subset H' of H and then a polynomial check if $facts(\widehat{P \cup H'}) = heads(\widehat{P \cup H'})$. For hardness note that, by definition, there is an NP-complete operation (the guessing of the subset of hypotheses H) and subsequent NP-complete check for minimality involved in the process of generating the MH models candidates for entailing Q.

Cautious Reasoning

Conversely, Cautious Reasoning, i.e., guaranteeing that every MH model satisfies some particular set of literals, is Π_2^P-complete.

Theorem 6. ***Cautious Reasoning with MH semantics is Π_2^P-complete.*** *Let P be an NLP, and Q a set of literals, or* query. *Guaranteeing that all MH models are such that $M \supseteq Q$ is a Π_2^P-complete task.*

Proof. Cautious Reasoning is the complement of Brave Reasoning, and since the latter is Σ_2^P-complete (theorem 5), the former must necessarily be Π_2^P-complete.

The set of hypotheses $Hyps(P)$ is obtained from $\mathring{P}$, which identifies rules that depend on themselves. The hypotheses are the atoms of DNLs of $\mathring{P}$, i.e., the "atoms of *not*s in loop". A Minimal Hypotheses model is then obtained from a minimal set of these hypotheses sufficient to determine the 2-valued truth-value of every literal in the program. The MH semantics imposes no ordering or preference between hypotheses — only their set-inclusion minimality. For this reason, one can think of the choosing of a set of hypotheses yielding a MH model as finding a minimal solution to a disjunction problem, where the disjuncts are the hypotheses. In this sense, it is therefore understandable that the complexity of the reasoning tasks with MH semantics is in line with that of, e.g., reasoning tasks with SM semantics with

Disjunctive Logic Programs, i.e, Σ_2^P-complete and Π_2^P-complete, Eiter and Gottlob (1993).

In abductive reasoning (as well as in Belief Revision) one does not always require minimal solutions. Likewise, taking a hypotheses assumption based semantic approach, like the one of MH, one might instead not require minimality of assumed hypotheses. In such a case, one would be under a non-Minimal Hypotheses semantics, and the complexity classes of the corresponding reasoning task would be one level down in the Polynomial Hierarchy relatively to the MH semantics, i.e., Brave Reasoning with a non-Minimal Hypotheses semantics would be NP-complete, and Cautious Reasoning would be coNP-complete. We leave the exploration of such possibilities for future work.

Comparisons

As has been seen all stable models are MH models. Since MH models are always guaranteed to exist for every regular NLP (cf. theorem 2), wheras SMs are not, it follows immediately that the Minimal Hypotheses semantics is a strict, model conservative, generalisation of the Stable Models semantics. The MH models that are stable models are exactly those in which all rules are classically supported. With this criterion one can conclude whether some program does not have any stable models. For Normal Logic Programs, the Stable Models semantics coincides with the Answer-Set semantics (which is a generalisation of SMs to Extended Logic Programs), where the latter is known (cf. Gelfond and Lifschitz (1990)) to correspond to Reiter's default logic. Hence, all Reiter's default extensions have a corresponding Minimal Hypotheses model.
Also, since Moore's expansions of an autoepistemic theory Moore (1985) are known to have a one-to-one correspondence with the stable models of the NLP version of the theory, one concludes that for every such expansion there is a matching Minimal Hypotheses model for the same NLP.

Disjunctive Logic Programs (DisjLPs — allowing for disjunctions in the heads of rules) can be syntactically transformed into NLPs by applying the Shifting Rule, presented in Dix et al. (1996), in all possible ways. By non-deterministically applying such transformation in all possible ways, several SCCs of rules may appear in the resulting NLP that were not present in the original DisjLP — assigning a meaning to every such SCC is a distinctive feature of MH semantics, unlike of other semantics. This way, the MH semantics can be defined for DisjLPs as well: the MH models of a DisjLP are the MH models of the NLP resulting from the transformation via the Shifting Rule.

There are other kinds of disjunction, like the one in logic programs with ordered disjunction (LPOD) Brewka (2002). These employ "*a new connective called ordered disjunction. The new connective allows to represent alternative, ranked options for problem solutions in the heads of rules*". As the author Brewka (2002) says, "the semantics of logic programs with ordered disjunction is based on a preference

relation on answer sets." This is different from the semantics assigned by MH since the latter includes no ordering, nor preferences, in the assumed minimal sets of hypotheses. E.g., in example 1 there is no notion of preference or ordering amongst candidate models — LPODs would not be the appropriate formalism for such cases. We leave for future work a thorough comparison of these approaches, namely comparing the semantics of LPODs against the MH models of LPODs transformed into NLPs (via the Shifting Rule).

The motivation for Witteveen (1995) is similar to our own — to assign a semantics to *every* NLP — however their approach is different from ours in the sense that the methods in Witteveen (1995) resort to contrapositive rules allowing any positive literal in the head to be shifted (by negating it) to the body, or any negative literal in the body to be shifted to the head (by making it positive). This approach considers each rule as a disjunction, making no distinction between such literals occurring in the rule, whether or not they are in loop with the head of the rule. This permits the shifting operations in Witteveen (1995) to create support for atoms that have no rules in the original program. E.g.

Example 7. **Nearly-Stable Models vs. MH models.** Take the program $P =$

$$\begin{array}{l} a \leftarrow not\ b \\ b \leftarrow not\ c \\ c \leftarrow not\ a, not\ x \end{array}$$

According to the shifting operations in Witteveen (1995) this program could be transformed into $P' =$

$$\begin{array}{l} b \leftarrow not\ a \\ b \leftarrow not\ c \\ x \leftarrow not\ a, not\ c \end{array}$$

by shifting a and *not* b in the first rule, and shifting the *not* x to the head (becoming positive x) and c to the body (becoming negative *not* c) of the third rule thus allowing for $\{b,x\}$ (which is a stable model of P') to be a *nearly stable model* of P. In this sense the approach of Witteveen (1995) allows for the violation of the Closed-World Assumption. This does not happen with our approach: $\{b,x\}$ is not a Minimal Hypotheses model simply because since x has no rules in P it cannot be true in any MH model — *not* x is not a member of $Hyps(P)$ (cf. def. 18).

As shown in theorem 1, at least one MH model of a program complies with its well-founded model, although not necessarily all MH models do. E.g., the program in Ex. 2 has the two MH full models $\{beach, mountain, not\ travel\}$ and $\{beach, not\ mountain, travel\}$, whereas the $WFM(P)$ imposes $WFM^{-}(P) = \{travel\}$, $WFM^{u}(P) = \emptyset$, and $WFM^{+}(P) = \{beach, mountain\}$. This is due to the set of hypotheses $Hyps(P)$ of P being taken from $\mathring{P}$ (based on the layered support notion) instead of being taken from $\widehat{P}$ (based on the classical notion of support).

Not all Minimal Hypotheses models are Minimal Models of a program. The rationale behind MH semantics is minimality of hypotheses, but not necessarily minimality of consequences, the latter being enforceable, if so desired, as an additional requirement, although at the expense of increased complexity.

The relation between logic programs and argumentation systems has been considered for a long time now, cf. (Dung (1995) amongst many others) and the authors have also taken steps to understand and further that relationship in Pereira and Pinto (2007a,b, 2008). Dung's Preferred Extensions Dung (1995) are maximal sets of negative hypotheses yielding consistent models. Preferred Extensions, however, are not guaranteed to always yield 2-valued complete models. Our previous approaches Pereira and Pinto (2007a,b) to argumentation have already addressed the issue of 2-valued model existence guarantee, and the MH semantics also solves that problem by virtue of positive, instead of negative, hypotheses assumption.

Other comparisons have already been mentioned at the end of the Motivation subsection.

Our current work is not an alternative approach to deal with inconsistent knowledge in general; comparisons with ways to deal with inconsistencies are therefore not relevant in the scope of this work. Our contribution is just a re-definition of how normal logic rules and default negation can be viewed so as to deal with inconsistencies brought about by OLONs, all the while staying close to the spirit and paradigm of well-accepted NLP semantics.

MH models of *P* as *SM*s of a transform P^* of P: justification and examples

Given a NLP *P*, without infinite descending chains (cf. Example 6), the authors define $MH \rightarrow SM$, a program transformation of *P* into P^*, such that the *MH* models of *P* can be obtained from the *SM*s of P^*. We provide its justification and give a number of examples of use.

Motivation

Showing how the *MH*s of a *P* can be seen as *SM*s of a transform dubbed P^* helps mainly to further understand their differences, and to show the way *MH*s can be computed by explaining their computation in terms of the well-known computation of the Stable Models. The rationale of the transformation $MH \rightarrow SM$ that constructs the transform provides intuitions about the two semantics' relationship, illustrated in a number of detailed examples. Moreover, $MH \rightarrow SM$ provides a first proof-of-principle approach to implementing *MH* semantics on existing Answer Set Programming (ASP) systems. We have not fully performed this implementation, nor purport that it is efficacious as it stands, hence its complexity analysis is not relevant at this stage. Furthermore, it resorts to algorithms not shown here, namely for computing the Strongly Connected Components of a program, and for calculating circumscribed set minimization.

Rationale

Let $Hyps(P)$ be the set of assumable hypotheses of P (cf. definition 18), and H some subset of $Hyps(P)$. Recall Definition 20: a 2-valued model M of P is a Minimal Hypotheses model of P iff

$$M = facts(\widehat{P \cup H}) = heads(\widehat{P \cup H})$$

where $H = \emptyset$ or H is non-empty set-inclusion minimal (the set-inclusion minimality is considered only for non-empty Hs). For expressing a Stable Model M of P^* in a way similar to above, but rather than joining as facts to P^* positive hypotheses $S \subseteq CHyps(P^*)$ defined in 19, one takes instead their complement default literals in P^*, with respect to model M, say NS, and factor these into program P^* by means of the well-known program division operation $/$ of Gelfond and Lifschitz (1988), represented as P^*/M), by ensuring $S \subseteq M$. So, by appropriately defining P^*, one wishes to have the property, dubbed here $MHSM$, formulated analagously to the one above: A Stable Model M of P^* is a MH model of P iff $S \subseteq M \wedge$

$$M = facts(\widehat{P^*/M}) = heads(\widehat{P^*/M})$$

where $S = \emptyset$ or S is non-empty set-inclusion minimal (the set-inclusion minimality is considered only for non-empty Ss). Note that $M = facts(\widehat{P^*/M})$ by itself warrants M is a Stable Model of P, because of the use of the Remainder operator (definition 16). The rationale is that, by minimizing S one is maximising the NS in P^*, and that the S are the only atoms in a loop over default negation, i.e. those that at most need to be hypothesized about.

The Transformation $MH \rightarrow SM$

Since our goal is to transform P into a P^* for which the SMs of P^*, that are minimal with respect to $Hyps(P)$, are the MHs of P, one must transform P in a way that $Hyps(P) = CHyps(P^*)$, so that the same candidates H and S are available. It will also be apparent that, with the transformation below, $CHyps(P^*) = Hyps(P^*)$. Because $\mathring{P}$ is used for defining $Hyps(P)$, and $\widehat{P}^*$ is used for defining $CHyps(P)$, one wishes to have $\mathring{P}^* = \widehat{P}^*$. Now, the Layered Remainder $\mathring{P}^*$ (definition 17) of P^*, and the Remainder $\widehat{P}^*$ (definition 16) of P^*, differ only in that the first makes use of operator $\mapsto_{LN}$ (Layered Negative reduction), whereas the second makes use of operator $\mapsto_N$ (Negative reduction), otherwise using the same other rewrite operators. We thus need, in the transform P^*, to have $\mapsto_{LN} = \mapsto_N$. That is, one must make the Loops Over Negation (LONs) in each Strongly Connected Component (SCC) independent. To achieve that, one needs to replace all negated literals *not* A and their positive counterparts A, which participate in a joint LON in one same SCC, to be respectively transformed into new literals, say, *not* A' and A', in order to make the

*LON*s in each *SCC* independent. To that effect, one may employ as many priming sequences as necessary, in order to distinguish the different *SCCs* where such *LON*s exist; e.g. A'', *not* A''', etc., for each distinct *SCC*. Furthermore, to provide "output" from each *SCC* one adds a rule, for every such primed A:

$$A \leftarrow A^n \text{ with } n \text{ standing for } n \text{ occurrences of prime symbol } '$$

P^* must also guarantee that the same hypotheses candidates are available, i.e. $S = H$. This is achieved by introducing for every A primed, say A^n, the *ELON*

$$A^n \leftarrow not\ A^{*n}$$
$$A^{*n} \leftarrow not\ A^n$$

Our hypotheses now all have the form A^n or A^{*n}, and because one wants to minimize H one now wants to minimize the A^n, by maximizing the A^{*n} to start with.

Computing the Stable Models of the Transform

After applying definition 20 to obtain *MH* models for P^*, and retaining first only those with set inclusion maximal starred atoms, next one ignores in them the primed and the starred atoms to obtain a desired model clean candidate *MH* model M. Only the Ms with minimal S, modulo $\emptyset$, are candidate *MH* models of P^*, of which only those with minimal $Hyps(P^*)$ are retained, of course.

Given the property *MHSM*, one may as well obtain the *MH*s simply by first computing the *SM*s of P^*, and then obtain the *MH*s, which are the cleaned *SM* models gotten first from the maximal starred and then minimal S ones, as just described above. Indeed, the *ELON*s introduced for the hypotheses make it irrelevant to add these as facts to P^*. And the usual Gelfond-Lifschitz-based computation of the stable models of P^* replaces the use of the Remainder without loss. Even if property *MHSM* were not invoked, one can still argue, based on the intuitions buttressing the construction of the transformation, that the *SM*s of P^*, computed in the usual way, provide the (clean and minimal) *MH*s of P. This transformation thus provides a method for using the Stable Models implementation systems (viz. ASP ones) to compute *MH*s. The extra steps of identifying maximally starred and minimal hypotheses models introduces extra complexity with regard to *SM* semantics, as to be expected. The initial step of finding the *LON*s or the *SCC*s (definition 13) is of polynomial complexity, Tarjan (1972). Indeed, the introduction of the mentioned *ELON*s guarantees that every atom A in a *LON* also partakes of an independent *ELON* (made by introducing the *-atoms), thereby permitting the breaking up of pure *OLON*s. If the *OLON*s are broken in some other way, then the maximizing of the new *ELON*'s starred A^{*n} atoms will not produce new A. Also, such new *ELON*s do not affect existing *ELON*s because the choices made in the existing ones will force the new *ELON*s to comply with them, on account of their *not* A^n bodies. If the existing *ELON*s are broken in some other way, then the maximizing of the new

ELON's starred A^{*n} atoms will not produce new A. Moreover, the (independent) "output" rules will ensure compatibility of choices in the *LON*s, and propagation of the consequences of those choices.

A note on abduction

Given the abductive flavour of *MH* semantics, to equal effect one may instead introduce new *ELON*s of the form:

$$\begin{aligned} A^{n} &\leftarrow A^{+n} \\ A^{+n} &\leftarrow \mathit{not}\ A^{*n} \\ A^{*n} &\leftarrow \mathit{not}\ A^{+n} \end{aligned}$$

so that now, rather than speaking of maximizing the starred atoms, one may equivalently speak of minimizing the plussed atoms introduced above. The plussed atoms may be envisaged as abducible hypotheses to be minimized, which can be coded, as usual, by an *ELON*.

MH $\rightarrow$ *SM* ***Transformation Rules for*** P ***to Obtain*** P^*

For each strongly connected component *SCC* of P:

1. Rewrite into new reserved atoms, by priming them (i.e. affixing with " ' ", single, double, triple, etc., according to the *SCC* component) each of the atoms A within that *SCC* that take part in a *LON* (Loop Over Negation, whether odd (*OLON*) or even (*ELON*)). That is, those atoms A having a *not* A in some rule body and an A head in a rule, in that *SCC*.
2. Indicate, by means of a new rule, that each different so primed atom arising from an atom A entails the corresponding prime-free atom A. This rule is used for outputing A from the *SCC* and for its consumption by the other *SCC*s.
3. Introduce a new *ELON* for each primed atom A by resorting to a corresponding new reserved starred A^* atom, affected with the same number of primes as A. These *ELON*s will act as generators of *SM* hypotheses or their negation. Starred atoms will be maximized in the final step of model selection to prevent initial excessive generation of hypotheses.
4. The *MH*s of P will be the *SM*s of P^*, minimal with respect to $Hyps(P) = Hyps(P^*)$, which are considered modulo their primed and maximized asterisked atoms, and in which the set of non-primed atoms A having been primed is set-inclusion minimal, ignoring the empty set $\emptyset$ for carrying out of minimization filtering of other sets.

Transformation Examples and Results

Example 8. Take P, comprised of one *SCC*, where $Hyps(P) = \{r\}$:

$$r \leftarrow not\ r$$

The only *MH* model of P is: $\mathbf{M} = \{\mathbf{r}\}$. In bold is the only hypothesis chosen. When the above rules are applied to P, one obtains the transformed P^*:

SCC′ (single primed):

$$\begin{array}{rll} r' \leftarrow not\ r' & & \% \text{ rewritten rule} \\ r \leftarrow r' & & \% \text{ output rule} \\ r' \leftarrow not\ r^{*\prime} & & \% \text{ the two generating } ELON \text{ rules} \\ r^{*\prime} \leftarrow not\ r' & & \end{array}$$

The only *SM* model of P^* is shown first with primes (and asterisks, of which none), followed by their removal version:

$$M_1^* = \{\mathbf{r}', r\} \qquad M_1 = \{\mathbf{r}\} = \mathbf{M}$$

$$\% \text{ the choice of } r^{*\prime} \rightarrow r' \text{ and is not stable.}$$

Example 9. Take P, made up of one *SCC*, where $Hyps(P) = \{p, q\}$:

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

The two *MH* models of P are: $\mathbf{M_1} = \{\mathbf{p}\}$ and $\mathbf{M_2} = \{\mathbf{q}\}$, where in bold are shown the minimal hypotheses chosen in each of the *MH* models. Both are *SM*s. When the above rules are applied to P, one obtains the transformed P^*:

SCC1′ (single primed):

$$\begin{array}{rl} p' \leftarrow not\ q' & \\ q' \leftarrow not\ p' & \\ & \% \text{ output rules:} \\ p \leftarrow p' & \\ q \leftarrow q' & \\ & \% \text{ generating ELONs:} \\ p' \leftarrow not\ p^{*\prime} & \\ p^{*\prime} \leftarrow not\ p' & \\ & \\ q' \leftarrow not\ q^{*\prime} & \\ q^{*\prime} \leftarrow not\ q' & \end{array}$$

The *SM*s of P^* are, shown first with primes and asterisks, and immediately followed by their removal versions:

$$\begin{array}{ll} M_1^* = \{q^{*\prime}, \mathbf{p}', p\} & M_1 = \{\mathbf{p}\} \\ M_2^* = \{p^{*\prime}, \mathbf{q}', p\} & M_2 = \{\mathbf{q}\} \\ M_3^* = \{\mathbf{q}', \mathbf{p}', p, q\} & \%\ M_3 = \{p,q\} \text{ is asterisk non-maximal} \end{array}$$

All *SM*s found are indeed *MH*s of P.

Example 10. Take P, made up of two *SCC*s, where $Hyps(P) = \{a, b\}$:

SCC2:

$$\begin{array}{l} a \leftarrow not\ b \\ b \leftarrow not\ a \end{array}$$

SCC1:

$$b$$

The two *MH* models of P are: $\mathbf{M_1} = \{\mathbf{a}, b\}$ and $\mathbf{M_2} = \{\mathbf{b}\}$, where in bold are shown the minimal hypotheses chosen in each of the *MH* models. Only M_2 is a *SM*. When the above rules are applied to P, one obtains the transformed P^* :

SCC2" (twice primed):

$$\begin{array}{l} \qquad\qquad\qquad\qquad \%\ \text{rewritten rules:} \\ a'' \leftarrow not\ b'' \\ b'' \leftarrow not\ a'' \\ \qquad\qquad\qquad\qquad \%\ \text{output rules:} \\ a \leftarrow a'' \\ b \leftarrow b'' \\ \qquad\qquad\qquad\qquad \%\ \text{generating ELONs:} \\ a'' \leftarrow not\ a^{*\prime\prime} \\ a^{*\prime\prime} \leftarrow not\ a'' \\ \\ b'' \leftarrow not\ b^{*\prime\prime} \\ b^{*\prime\prime} \leftarrow not\ b'' \end{array}$$

SCC1' = SCC1

$$\begin{array}{l} \qquad\qquad\qquad\qquad \%\ \text{rewritten rules (no change):} \\ b \\ \qquad\qquad\qquad\qquad \%\ \text{output rules} \\ a \leftarrow a'' \\ b \leftarrow b'' \end{array}$$

The *SM*s of P^* are, shown first with primes and asterisks, and immediately followed by their removal versions:

$M^*_{11} = \{b, b^{*\prime\prime}, \mathbf{a}'', a\}$ $\quad M_1 = \{\mathbf{a}, b\} = \mathbf{M_1}$
$M^*_{12} = \{b, \mathbf{b}'', \mathbf{a}'', a\}$ $\quad \%\ M_{12} = \{\mathbf{a}, \mathbf{b}\}$ is asterisk non-maximal
$M^*_{13} = \{b, \mathbf{b}'', a^{*\prime\prime}\}$ $\quad M_{13} = \{\mathbf{b}\} = \mathbf{M_2}$

% the choice of $b^{*\prime\prime}$ and $a^{*\prime\prime}$ is not stable

Example 11. Take P, made up of four *SCC*s, where $Hyps(P) = \{r, p, q, a, b, c\}$:

SCC4:

$a \leftarrow not\ b$
$b \leftarrow not\ c$
$c \leftarrow not\ a, r, p$

SCC3:

$a \leftarrow p$

SCC2:

$p \leftarrow not\ q$
$q \leftarrow not\ p$

SCC1:

$r \leftarrow not\ r$

The two *MH* models of P are: $\mathbf{M_1} = \{\mathbf{r}, \mathbf{p}, a, b\}$ and $\mathbf{M_2} = \{\mathbf{r}, \mathbf{q}, b\}$, where in bold are shown the minimal hypotheses chosen in each of the *MH* models. When the above rules are applied to P, one obtains the transformed P^* :

SCC4" (twice primed):

% rewritten rules:
$a'' \leftarrow not\ b''$
$b'' \leftarrow not\ c''$
$c'' \leftarrow not\ a'', r, p$

% output rules:
$a \leftarrow a''$
$b \leftarrow b''$
$c \leftarrow c''$

% generating *ELON*s:
$a'' \leftarrow not\ a^{*\prime\prime}$
$a^{*\prime\prime} \leftarrow not\ a''$

$b'' \leftarrow not\ b^{*\prime\prime}$
$b^{*\prime\prime} \leftarrow not\ b''$

$c'' \leftarrow not\ c^{*\prime\prime}$
$c^{*\prime\prime} \leftarrow not\ c''$

SCC3 (non-primed, no *LON*s):

$$a \leftarrow p$$

SCC2' (single primed):

$$\begin{array}{rl} & \% \text{ rewritten rules:} \\ p' \leftarrow not\ q' & \\ q' \leftarrow not\ p' & \\ & \% \text{ output rules:} \\ p \leftarrow p' & \\ q \leftarrow q' & \\ & \% \text{ generating } ELON\text{s :} \\ p' \leftarrow not\ p^{*\prime} & \\ p^{*\prime} \leftarrow not\ p' & \\ & \\ q' \leftarrow not\ q^{*\prime} & \\ q^{*\prime} \leftarrow not\ q' & \end{array}$$

SCC1''' (thrice-primed):

$$\begin{array}{rl} & \% \text{ rewritten rules:} \\ r''' \leftarrow not\ r''' & \\ & \% \text{ generating } ELON\text{s:} \\ r''' \leftarrow not\ r^{*'''} & \\ r^{*'''} \leftarrow not\ r''' & \\ & \% \text{ output rule:} \\ r \leftarrow r''' & \end{array}$$

In essence, this example exhibits the separation of *SCC*s from one another in what concerns each one's *LON*s; an output contribution of each such *SCC LON*, both for possible uptake by other *SCC*s and for contributing to the global model; and a generation made possible by *ELON*s of positive/negative hypotheses for those *LON*s, subject to the global minimization imposed by *MH* semantics. Atoms not participating in *LON*s remain untouched. The *SM* semantics, or the Remainder, are charged with the directional propagation of consequences from lower *SCC*s to the ones above them. Only the *SM*s of P^*, clean of primes and asterisks, figuring non-minimal hypotheses modulo the empty set $\emptyset$ are retained as the *MH*s of P.

The *SM*s of P^* are, shown first with primes and asterisks, and immediately followed by their removal versions:

First one considers the models containing r''' and p'. The latter choice, p', causes ignoring the possibility of q', on account of the maximization of asterisked atoms rule, which in this case can be perform beforehand because it leads to the same result, since q or *not* q do not figure in *SCC*2 and thus cannot directly influence choices there.

$$\begin{array}{ll} M^*_{11} = \{\mathbf{r'''}, r, \mathbf{p'}, p, a', a, \mathbf{a''}, \mathbf{c}^{*\prime\prime}, b'', b, \mathbf{b}^{*\prime\prime}\} & M_{11} = \{r, p, a, b\} = \mathbf{M_1} \\ M^*_{12} = \{r''', r, \mathbf{p'}, p, a', a, a^{*\prime\prime}, \mathbf{c''}, c, \mathbf{b}^{*\prime\prime}\} & \% \text{ not asterisk maximal given } M^*_{11} \end{array}$$

$$\% \text{ choices with } \mathbf{r'''}, \mathbf{p'}, \mathbf{a^{*''}}, \mathbf{c^{*''}} \rightarrow \mathbf{c''} \text{ and are unstable}$$

$M^*_{13} = \{\mathbf{r'''}, r, \mathbf{p'}, p, a', a, a'', c'', c, b^{*''}\}$ % not asterisk maximal given M^*_{11}

Choosing $r^{*'''}$ provokes non-stability because of rule $r''' \leftarrow not\ r'''$, and hence no model contains it. Next consider models with r''' and q', and ignore choosing p' as well because of the reason given before.

$$\% \text{ choices with } r''', q', a^{*''}, b^{*''} \rightarrow a' \text{ and are unstable}$$

$M^*_{21} = \{\mathbf{r'''}, r, \mathbf{q'}, q, \mathbf{a^{*''}}, \mathbf{b''}, b, \mathbf{c^{*''}}\}$ $M_{21} = \{r, q, b\} = \mathbf{M_2}$
$M^*_{22} = \{\mathbf{r'''}, r, \mathbf{q'}, q, \mathbf{a^{*''}}, \mathbf{b''}, b, \mathbf{c''}, c\}$ $\% M_{22} = \{r, q, b, c\} \subseteq M_{21}\ Hyps(P)$ non-minimal
% also not asterisk maximal given M^*_{21}

All selected *SM*s found are indeed the *MH*s of P.

Example 12. Take P, with just one *SCC*, and $Hyps(P) = \{beach, travel, mountain\}$:

SCC1:

$$beach \leftarrow not\ travel$$
$$travel \leftarrow not\ mountain$$
$$mountain \leftarrow not\ beach$$

henceforth abbreviated to

$$b \leftarrow not\ t$$
$$t \leftarrow not\ m$$
$$m \leftarrow not\ b$$

and $Hyps(P) = \{b, t, m\}$. The three *MH* models of P are: $\mathbf{M_1} = \{\mathbf{b}, t\}$, $\mathbf{M_2} = \{\mathbf{t}, m\}$, $\mathbf{M_3} = \{\mathbf{m}, b\}$, where in bold are shown the minimal hypotheses chosen in each of the *MH* models. When the above rules are applied to P, one obtains the transformed P^* :

SCC1' (single primed):

$$
\begin{array}{rl}
 & \% \text{ rewritten rules:} \\
b' \leftarrow not\ t' & \\
t' \leftarrow not\ m' & \\
m' \leftarrow not\ b' & \\
 & \% \text{ output rules:} \\
b \leftarrow b' & \\
t \leftarrow t' & \\
m \leftarrow m' & \\
 & \% \text{ generating } ELON\text{s:} \\
b' \leftarrow not\ b^{*\prime} & \\
b^{*\prime} \leftarrow not\ b' & \\
 & \\
t' \leftarrow not\ t^{*\prime} & \\
t^{*\prime} \leftarrow not\ t' & \\
 & \\
m' \leftarrow not\ m^{*\prime} & \\
m^{*\prime} \leftarrow not\ m' &
\end{array}
$$

The *SM*s of P^* are, shown first with primes and maximal asterisks, and immediately followed by their removal versions:

$$
\begin{array}{ll}
M_3^* = \{\mathbf{m'}, m, \mathbf{t}^{*\prime}, \mathbf{b'}, b\} & M_3 = \{m, b\} = \mathbf{M_3} \\
M_2^* = \{\mathbf{m'}, m, \mathbf{t'}, t, \mathbf{b}^{*\prime}\} & M_2 = \{t, m\} = \mathbf{M_2} \\
M_1^* = \{\mathbf{b'}, b, \mathbf{m}^{*\prime}, \mathbf{t'}, t\} & M_1 = \{b, t\} = \mathbf{M_1}
\end{array}
$$

When the single fact *beach* is added to P, a new SCC2 obtains:

SCC2:

$$b$$

The two *MH* models of P are: $\mathbf{M_1} = \{\mathbf{b}, t\}$, $\mathbf{M_3} = \{\mathbf{m}, b\}$, where in bold are shown the minimal hypotheses chosen in each of the *MH* models.

The previous *SM*s of P^* are added with fact b, where consistent with minimality of $Hyps(P) = \{b, t, m\}$, and hence $\mathbf{M_2} = \{b, \mathbf{t}, m\}$ is eliminated since not minimal in view of the existence of $\mathbf{M_1}$.

Conclusions and Future Work

Taking a positive hypotheses assumption, abductive, approach the authors defined the 2-valued Minimal Hypotheses semantics for NLPs that guarantees model existence in the absence of ICs, enjoys relevancy and cumulativity, and is also a model conservative generalisation of the SM semantics. Moreover, by adopting positive hypotheses, the authors not only generalised the argumentation based approach of Dung (1995), but the resulting MH semantics lends itself naturally to abductive

reasoning, it being understood as hypothesising plausible reasons sufficient for justifying given observations or supporting desired goals. We also defined the layered support notion, which generalises the classical one of stratification, by recognising the special role of loops, and aptly deals with loop stratification with the concept of layering.

For query answering, the MH semantics provides mainly three advantages: 1) by enjoying Relevancy top-down query-solving is possible, thereby circumventing whole model computation (and grounding) which is unavoidable in other semantics; 2) by considering only the relevant sub-part of the program when answering a query, it is possible to enact grounding of just the sub-part rules, but only if grounding is really desired, as it is not needed since whole models need not be computed to check model existence, whereas with other semantics whole program grounding is inevitable, which is known as a major source of computational time consumption. MH semantics, by enjoying Relevancy, permits curbing this task to the minimum sufficient to answer a query. Solutions to a query need not be ground, as relevancy warrants their extension to an existing complete model, but only if desired. 3) by enjoying Cumulativity, as soon as the truth-value of a literal is determined in a branch for the top query, then it can be stored in a table and its value used to speed up the computations of other branches within the same top query.

Goal-driven abductive reasoning is elegantly modelled by top-down abductive-query-solving. By taking a positive hypotheses abductive approach, whilst enjoying Relevancy, MH semantics caters well as a convenient KRR tool.

Many applications have been developed using the Stable Models/Answer-Sets semantics as the underlying platform. These generally tend to be focused on solving problems that require complete knowledge, such as constrained search problems where all the knowledge represented is relevant to the solutions. However, as Knowledge Bases increase in size and complexity, and as merging and updating of KBs becomes more and more common, e.g. for Semantic Web applications, as argued in Gomes et al. (2010), partial knowledge problem solving's importance grows, as does the need to ensure overall consistency of the merged or updated KBs.

The Minimal Hypotheses semantics is intended as an alternative, not a replacement of Stable Models, though it can be utilised, once specifically and efficiently implemented (by us or by others), for *all* the applications where the Stable Models semantics are themselves used to model KRR and constrained search problems, *plus* all applications where query answering (both under a credulous mode of reasoning and under a skeptical one) is intended, *plus* all applications where abductive reasoning is a requirement. The MH semantics purports to be a sound theoretical platform for 2-valued abductive reasoning with logic programs.

Much work still remains to be done that can be rooted on this platform's contribution. The general topics of using non-normal logic programs, those allowing for default and/or explicit negation in the heads of rules, for Belief Revision, Updates, Preferences, etc., are *per se* orthogonal to the foundational semantics issue and, therefore, such subjects can be addressed with Minimal Hypotheses semantics as the underlying platform. Importantly, MH can guarantee the liveness of updated and self-updating LP programs, such as those of EVOLP in Alferes et al. (2002),

and related applications. The Minimal Hypotheses semantics still lacks to be fully compared with the related Revised Stable Models Pereira and Pinto (2005), PStable Models Osorio and Nieves (2007), and other ever ongoing semantics research.

In short, the authors have provided a fresh platform on which to re-examine ever present issues in Logic Programming and its uses, which aspires to provide one natural approach to continued Logic Programming development.

Acknowledgements

L.M.P. is supported by NOVA LINCS (UIDB/04516/2020) with the financial support of FCT- Fundação para a Ciência e a Tecnologia, Portugal, through national funds.

References

ALFERES, J., BROGI, A., LEITE, J. A., AND PEREIRA, L. M. 2002. Evolving logic programs. In *Procs. JELIA'02*, S. F. et al., Ed. LNCS, vol. 2424. Springer, 50–61.

ALFERES, J. J., PEREIRA, L. M., AND SWIFT, T. 2004. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *Theory and Practice of Logic Programming 4,* 4, 383–428.

BRASS, S., DIX, J., FREITAG, B., AND ZUKOWSKI, U. 2001. Transformation-based bottom-up computation of the well-founded model. *TPLP 1,* 5, 497–538.

BREWKA, G. 2002. Logic programming with ordered disjunction. In *AAAI-02*. AAAI Press, 100–105.

DIX, J. 1995a. A Classification Theory of Semantics of Normal Logic Programs: I. Strong Properties. *Fundam. Inform. 22,* 3, 227–255.

DIX, J. 1995b. A Classification Theory of Semantics of Normal Logic Programs: II. Weak Properties. *Fundam. Inform. 22,* 3, 257–288.

DIX, J., GOTTLOB, G., MAREK, W., AND RAUSZER, C. 1996. Reducing disjunctive to non-disjunctive semantics by shift-operations. *Fundamenta Informaticae 28*, 87–100.

DUNG, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AI 77,* 2, 321–358.

EITER, T., FINK, M., AND MOURA, J. 2010. Paracoherent answer set programming. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)*. 486–496.

EITER, T. AND GOTTLOB, G. 1993. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In *International Logic Programming Symposium*. MIT Press, 266–278.

ESHGHI, K. AND KOWALSKI, R. A. 1989. Abduction compared with negation by failure. In *6th International Conference on Logic Programming (ICLP 1989)*. MIT Press, 234–255.

FAGES, F. 1994. Consistency of clark's completion and existence of stable models. *Methods of Logic in Computer Science 1*, 51–60.

GELDER, A. V., ROSS, K. A., AND SCHLIPF, J. S. 1991. The well-founded semantics for general logic programs. *J. ACM 38,* 3, 620–650.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *ICLP'88*. 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1990. Logic programs with classical negation. In *ICLP*, D. W. et al., Ed. MIT Press, 579–597.

GOMES, A. S., ALFERES, J. J., AND SWIFT, T. 2010. Implementing query answering for hybrid MKNF knowledge bases. In *PADL'10*, M. C. et al., Ed. LNCS, vol. 5937. Springer, 25–39.

KAKAS, A. C., KOWALSKI, R. A., AND TONI, F. 1992. Abductive logic programming. *J. Log. Comput. 2,* 6, 719–770.

MOORE, R. C. 1985. Semantical considerations on nonmonotonic logic. *AI 25,* 1, 75–94.

OSORIO, M. AND NIEVES, J. C. 2007. Pstable semantics for possibilistic logic programs. In *MICAI'07*. LNCS, vol. 4827. Springer, 294–304.

PEREIRA, L. M. AND PINTO, A. M. 2005. Revised stable models - a semantics for logic programs. In *EPIA'05*, C. B. et al., Ed. LNAI, vol. 3808. Springer, 29–42.

PEREIRA, L. M. AND PINTO, A. M. 2007a. Approved models for normal logic programs. In *LPAR'07*, N. Dershowitz and A. Voronkov, Eds. LNAI, vol. 4790. Springer.

PEREIRA, L. M. AND PINTO, A. M. 2007b. Reductio ad absurdum argumentation in normal logic programs. In *ArgNMR'07-LPNMR'07*, G. S. et al., Ed. Springer, 96–113.

PEREIRA, L. M. AND PINTO, A. M. 2008. *Collaborative vs. Conflicting Learning, Evolution and Argumentation, in: Oppositional Concepts in Computational Intelligence*. Studies in Computational Intelligence 155. Springer.

SAKAMA, C. AND INOUE, K. 1995. Paraconsistent stable semantics for extended disjunctive programs. *Journal of Logic and Computation 5,* 3, 265–285.

TARJAN, R. 1972. Depth-first search and linear graph algorithms. *SIAM J. on Computing 1,* 2, 146–160.

WITTEVEEN, C. 1995. Every normal program has a nearly stable model. In *Non-Monotonic Extensions of Logic Programming*, J. Dix, L. Pereira, and T. Przymusinski, Eds. Lecture Notes in Artificial Intelligence, vol. 927. Springer Verlag, Berlin, 68–84.