Approved Models for Normal Logic Programs

Luís Moniz Pereira and Alexandre Miguel Pinto
{lmp|amp}@di.fct.unl.pt

Centro de Inteligência Artificial (CENTRIA)
Universidade Nova de Lisboa
2829-516 Caparica, Portugal

Abstract. We introduce an original 2-valued semantics for Normal Logic Programs (NLPs) extending the well-known Argumentation work of Phan Minh Dung on Admissible Arguments and Preferred Extensions. In the 2-valued Approved Models Semantics set forth, an Approved Model (AM) correspond to the minimal positive strict consistent 2-valued completion of a Dung Preferred Extension. The AMs Semantics enjoys several non-trivial useful properties such as (1) Existence of a 2-valued Model for every NLP; (2) Relevancy, and (3) Cumulativity. Crucially, we show that the AMs Semantics is a conservative extension to the Stable Models (SMs) Semantics in the sense that every SM of a NLP is also an AM, thus providing every NLP with a model: a property not enjoyed by SMs. Integrity constraints, written in a simpler way, are introduced to identify undesired semantic scenarios, whilst permitting these to be produced nevertheless. We end the paper with some conclusions and mention of future work.
Keywords: Argumentation, *Reductio ad Absurdum*, Normal Logic Programs

1 Introduction

This paper introduces a new 2-valued semantics for Normal Logic Programs (NLPs) based upon and inspired by the previous well-know Argumentation work of Phan Minh Dung on Admissible Arguments and Preferred Extensions [6].

After introducing in [14] and [12] the new Revised Stable Models semantics for NLPs (whose complexity has been studied in [11]) further work using the *Reductio ad Absurdum* (RAA) principle has been developed, namely the Revised Well-Founded Semantics [15]. Considering an argument-based view of NLPs, we define a new semantics which inherits the RAA principle studied in [14, 12] and apply it to argumentation. Indeed, NLPs can be viewed as a collection of argumentative statements (rules) based on arguments (default negated literals) [2, 6, 4, 7].

We start by presenting the general Motivation of this paper and, after introducing some needed Background Notation and Definitions, the more detailed problem description. We proceed by setting forth our proposal — the Approved Models Semantics — and show how it extends previous known results.

The Approved Models (AMs) Semantics enjoys several useful properties such as (1) Existence of a 2-valued Model for every Normal Logic Program; (2) Relevancy, which allows for the development of purely top-down query-driven proof procedures; and (3) Cumulativity. Our approach, in accordance with the previous work on Revision

Complete Scenarios [13] and taking an argumentation point-of-view, allows one to obtain the Stable Models (SMs) as a special case with the advantage that every NLP has a model, a property not enjoyed by the SMs Semantics.

Indeed, we show that the Approved Models Semantics is a conservative extension to the Stable Models Semantics — in the sense that every Stable Model of a Normal Logic Program is also an Approved Model. This allows us to prove that, for certain specific types of NLPs, the Approved Models Semantics and the Stable Models Semantics coincide; and, therefore, for those NLPs the Stable Models Semantics also enjoys guarantee of Existence of a Model, Relevancy and Cumulativity. These important and useful results about the Stable Models Semantics are in line with those studied before for the Revised Stable Models Semantics [12, 14].

In the quest for finding an Approved Model one can guess it and check its compliance with the properties that characterize Approved Models. Using an innovative alternative approach, suiting the purposes of a Collaborative Argumentation setting, we can start with an arbitrary set of hypotheses (default negated literals), which can be the result of joining together several different alternative sets of hypotheses, calculate its consequences, and make revisions to the initial assumptions if necessary (when inconsistencies between the hypotheses and the consequences arise) in order to achieve positive minimality respecting stratification, 2-valued Completeness and Consistency, whilst also working toward guaranteeing that the set of negative hypotheses includes a Preferred Extension.

The set of negative hypotheses is made conflict-free, admissible, and maximal; thus including a Preferred Extension [6]). This set is then 2-valued consistently completed with the remaining atoms ensuring the whole 2-valued complete model is approvable (a notion we will present formally in the sequel).

Finally, integrity constraints are introduced to identify undesired models, whilst permitting these to be produced nevertheless. Conclusions and mention of Future Work finish the paper.

1.1 Motivation

Ever since the beginning of Logic Programming the scientific community has formally defined, in several ways, the meaning or semantics of a Logic Program. Several were defined, some 2-valued, some 3-valued, and even multi-valued semantics. The current standard 2-valued semantics for NLPs — the Stable Models Semantics [9] — has been around for almost 20 years now, and it is generally accepted as the *de facto* standard for NLPs. This thoroughly studied semantics, however, lacks some important properties among which the guarantee of Existence of a Model for every NLP.

In [12] we defined a 2-valued semantics — the Revised Stable Models — which extends the Stable Models Semantics, guarantees Existence of a Model for every Normal Logic Program, enjoys Relevancy (allowing for top-down query-driven proof-procedures to be built) and Cumulativity. Its main drawback is definitely that its definition is hard to grasp and understand.

Aiming to find a general perspective to seamlessly unify the Stable Models Semantics and the Revised Stable Models Semantics in a clear way, we drew our attention to Argumentation as a means to achieve it. This is the main motivation of the work

we present in this paper: by taking the Argumentation perspective we intend to show methods of identifying and finding a 2-valued complete Model for any NLP.

In [8], François Fages proved that a NLP might have no Stable Models iff it contains Odd Loops Over Negation (OLONs)[1] and/or Infinite Chains Over Negation (ICONs)[2]. Both these notions of OLON and ICON were formally and thoroughly studied in [14, 12] and the original Revised Stable Models Semantics was defined having also the OLONs and ICONs in mind.

For self-containment and to clarify the context, we now present a few informal motivating examples of OLONs and ICONs and how we solve them, so that every NLP has a semantics, employing a reasoning by contradiction argument. This kind of *reductio ad absurdum* reasoning takes place only when it is absolutely necessary in order to ensure 2-valued completeness of the resulting model, this "last resort" flavor of the RAA coming from the requirement of keeping the semantics maximally skeptical.

Example 1. **An invasion problem.** Some political leader thinks that "If Iran will have Weapons of Mass Destruction then we intend to invade Iran", also "If we do not intend to invade then surely they will have Weapons of Mass Destruction", rendered as the following OLON (where "not " denotes default negation)

$$\begin{array}{ll} intend\_to\_invade & \leftarrow iran\_will\_have\_WMD \\ iran\_will\_have\_WMD & \leftarrow not\ intend\_to\_invade \end{array}$$

The literal involved in the OLON is $intend\_to\_invade$ which is the literal that depends on itself through and odd number of default negations. The literal $iran\_will\_have\_WMD$ is just an in between stepping-stone for the OLON.

If we assume that "we do not intend to invade Iran" then, according to this program we will conclude that "Iran will have Weapons of Mass Destruction" and "we intend to invade Iran". These conclusions, in particular "we intend to invade Iran", contradict the initial hypothesis "we do not intend to invade Iran". So, reasoning by *Reductio ad Absurdum* in a 2-valued setting, we should "intend to invade Iran" in the first place.

This example gives a hint on how we resolve inconsistent arguments in the remainder of the paper, further exemplified below.

Example 2. **A going-out problem.** John likes Mary a lot so he asked her out: he said "We could go to the movies". Mary is more of a sports girl, so she replies "Either that, or we could go to the swimming pool". "Now, that's an interesting idea", John thought. The problem is that John cannot swim because he hasn't started learning to. He now thinks "Well, if I'm going to the swimming pool with Mary, and I haven't learned how to swim, I might risk drowning! And if I'm risking drowning then I really should start learning to swim". Here is the Normal Logic Program corresponding to these sentences:

$$\begin{array}{ll} start\_learning\_to\_swim & \leftarrow risk\_drowning \\ risk\_drowning & \leftarrow go\_to\_pool, not\ start\_learning\_to\_swim \\ go\_to\_pool & \leftarrow not\ go\_to\_movies \\ go\_to\_movies & \leftarrow not\ go\_to\_pool \end{array}$$

[1] Intuitively, an OLON is just a cycle in the dependency-graph induced by the NLP (an atom a depends on itself) where there is an odd number of default negations along the cycle.

[2] An example of an ICON with the explanation of the concept is presented below.

If John is not willing to go to the swimming pool — assuming $not\ go\_to\_pool$ — he just concludes go_to_movies and maybe he can convince Mary to join him. On the other hand, if the possibility of having a nice swim with Mary is more tempting, John assumes he is not going to the movies $not\ go\_to\_movies$ and therefore he concludes go_to_pool. In this case, since John does not know how to swim he could also assume $not\ start\_learning\_to\_swim$. But since John is going to the swimming pool, he concludes $risk\_drowning$. And because of $risk\_drowning$ he also concludes $start\_learning\_to\_swim$. That is, he must give up the hypothesis of $not\ start\_learning\_to\_swim$ in favour of $start\_learning\_to\_swim$ because he wants to go to the pool with Mary. As a nice side-effect he no longer risks drowning.

Example 3. **Middle region politics.** In a Middle Region two factions are at odds. One believes that if terrorism does not stop then oppression will do it and hence become unnecessary.

$oppression \leftarrow not\ end\_of\_terrorism \quad end\_of\_terrorism \leftarrow oppression$

The other faction believes that if oppression does not stop then terrorism will do it and hence become unnecessary.

$terrorism \leftarrow not\ end\_of\_oppression \quad end\_of\_oppression \leftarrow terrorism$

According to these rules, if we assume that $not\ end\_of\_terrorism$ we conclude that there is $oppression$ which in turn will cause the $end\_of\_terrorism$. So, the $end\_of\_terrorism$ should be true in the first place, instead of $not\ end\_of\_terrorism$. The same happens with $end\_of\_oppression$. In spite of the peaceful resulting solution we propose, $\{end\_of\_oppression, end\_of\_terrorism\}$, there is no Stable Model for this program.

Example 4. **An infinite chain over negation.** The classical example of an ICON was first presented in [8] and is

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

with a single constant 0 (zero), and so its ground version is

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

As we can see, the ground version has an Infinitely long descending Chain in the dependency graph for every literal $p(X)$ Over default Negation. In [14] the author proved that every possible ICON can be reduced to this canonical case, i.e., other ICONs may exist with more rules besides these, but the main structure (relevant for its properties) of every ICON relies on the same as the one presented here.

According to the Stable Models semantics this program has no models. But there is one Approved Model where every $p(X)$ is true. To see this, by *reductio ad absurdum*, assume that $p(X)$ was false for some X; then the two bodies of each clause above would have to be false, meaning that $p(s(X))$ would be true by the second one; but then, by the first one, $p(X)$ would be true as well, thereby contradicting the default assumption. Hence, by *Reductio ad Absurdum* reasoning, $p(X)$ must be true, for arbitrary X.

1.2 Background Notation and Definitions

Definition 1. ***Logic Rule.*** *A Logic Rule r has the general form*

$L \leftarrow B_1, B_2, \ldots, B_n, not\ C_1, not\ C_2, \ldots, not\ C_m$
where L is an atom h, the B_i and C_j are atoms.

We call L the head of the rule — also denoted by $head(r)$. And $body(r)$ denotes the set $\{B_1, B_2, \ldots, B_n, not\ C_1, not\ C_2, \ldots, not\ C_m\}$ of all the literals in the body of r. Throughout this paper we will use 'not ' to denote default negation.

When the body of the rule is empty, we say the head of rule is a fact and we write the rule just as h or $not\ h$. Since we are considering only Normal Logic Programs facts will never be of the form $not\ h$.

Definition 2. ***Logic Program.*** *A Logic Program (LP for short) P is a (possibly infinite) set of ground Logic Rules of the form in definition 1.*

In this paper we focus solely on Normal LPs (NLPs), those whose heads of rules are positive literals, i.e., simple atoms; and there is just default negation in the bodies of the rules. Hence, when we write just "program" or "logic program" we mean a NLP.

We use the notation $Atoms(P)$ to denote the set of all atoms of P; and $not\ S$ — where S is a set of literals (both positive and/or default negated) — to denote the set resulting from default negating every literal of S. Note that the default negation of an already default negated literal $not\ a$ is just the positive literal, i.e., $not\ not\ a \equiv a$.

Definition 3. ***2-valued Interpretation.*** *A 2-valued interpretation $I = I^+ \cup I^-$ of P is a set of literals, both positive $I^+ \subseteq Atoms(P)$ and negative $I^- \subseteq not\ Atoms(P)$, such that*

- *$I^+ \cap not\ I^- = \emptyset$ — it is consistent, i.e., there is no atom a of $Atoms(P)$ such that both a and $not\ a$ are in I*
- *$I^+ \cup not\ I^- = Atoms(P)$ — it is 2-valued complete, i.e., there is no atom a of $Atoms(P)$ such that both a and $not\ a$ are not in I*

Definition 4. ***$\vdash$ operator.*** *Let P be a NLP and I a 2-valued interpretation of P. P' is the Horn theory obtained from P by replacing every default literal of the form $not\ L$ in P by the atom not_L. I' is likewise obtained from I using the same replacement rule. By definition, $P' \cup I'$ is a Horn theory, and so it has a least model $M' = least(P' \cup I')$. We define $\vdash$ in the following way, where a is any atom of P:*

$P \cup I \vdash a$ *iff* $a \in M'$ $\qquad$ $P \cup I \vdash not\ a$ *iff* $not\_a \in M'$

In the sequel, we will write $M = least(P \cup I)$ to mean $M = \{L : P \cup I \vdash L\}$, where L is any positive (a) or negative ($not\ a$) literal derived from I in P. M corresponds thus to the result of applying the inverse substitution of literals not_a by $not\ a$ in M'. We will also sometimes refer to $least(P \cup I)$ as the consequences of I in P.

Definition 5. ***Internally Consistent 2-valued Interpretation.*** *Let P be a NLP and I a 2-valued interpretation I of P. I is Internally Consistent in P iff $least(P \cup I) \subseteq I$.*

Since we are considering only NLPs, where there are no negations in the heads of rules, we can never derive a negative literal by applications of the $least$ operator. Hence, the literals in $least(P \cup I)$ — besides including all the literals in I — include only positive literals (some heads of rules of P). By requiring $least(P \cup I) \subseteq I$ we ensure that all the literals in $least(P \cup I)$ do not contradict any of the literals in I^-;

knowing beforehand that literals in I^+ cannot ever be contradicted by $least(P \cup I)$ because P is a Normal Logic Program.

In fact, since I is 2-valued complete, if $least(P \cup I) \subseteq I$ then necessarily $least(P \cup I) = I$. Moreover, in [10], the authors prove that if $least(P \cup I^-) = I$, then I is a Stable Model of P and vice-versa, with the appropriate language translation. We also write $\Gamma_P(I)$ as a shorthand notation for $least(P \cup M^-) \setminus M^-$, i.e., $\Gamma_P(M)$ is just the positive part of $least(P \cup M^-)$. Hence, M **is a Stable Model of** P **iff** $\Gamma_P(M) = M$. For any interpretation I, we consider $\Gamma_P^0(I) = I$, and $\Gamma_P^{n+1}(I) = \Gamma_P(\Gamma_P^n(I))$.

Definition 6. ***Approvable Interpretation.*** *Let P be a NLP and I an Internally Consistent 2-valued Interpretation of P. We say I is an Approvable Interpretation of P iff I is such that I^- is set maximal. I.e., there is no other Internally Consistent 2-valued Interpretation I' of P such that $I'^- \supset I^-$.*

2 The Approved Models Semantics for Normal Logic Programs

The Approved Models Semantics for Normal Logic Programs gets its inspiration from the Argumentation perspective and also from our previous works [14], [12], and [13].

In [6], the author shows that preferred maximal scenarios (with maximum default negated literals — the hypotheses) are always guaranteed to exist for NLPs; and that when these yield 2-valued complete (total), consistent, admissible scenarios, they coincide with the Stable Models of the program. However, preferred maximal scenarios are, in general, 3-valued. The problem we address now is how to define 2-valued complete models based on preferred maximal scenarios. In this paper we take a step further from what we achieved in [6], extending its results. We do so by completing a preferred set of hypotheses rendering it approvable (as presented above in definition 6), ensuring whole model consistency and 2-valued completeness.

The resulting semantics thus defined, dubbed Approved Models Semantics, is a conservative extension to the widely known Stable Models semantics [9] in the sense that every Stable Model is also an Approved Model. The Approved Models are guaranteed to exist for every Normal Logic Program, whereas Stable Models are not. The concrete examples above show how NLPs with no Stable Models can usefully model knowledge, as well as produce additional models, as in Example 2. Moreover, this guarantee is crucial in program composition (say, from knowledge originating in divers sources) so that the result has a semantics. It is important too to warrant the existence of semantics after external updating, or in Stable Models based self-updating [1].

Moreover, the Approved Models Semantics enjoys the Relevancy property which allows for the development of purely top-down, program call-graph based, query driven methods to determine whether a literal belongs to some model or other. These methods can thus simply return a partial model, guaranteed extendable to a complete one, there being no need to compute all models or even to complete models in order to answer a query. Relevancy is crucial too for modeling abduction, it being query driven. Finally, the Approved Models Semantics enjoys Cumulativity, so lemmas may be stored and reused.

Before presenting the formal definition of an Approved Model we give a general intuitive idea to help the reader grasp the concept. For the formal definition of Approved Models Semantics we will also need some preliminary auxiliary definitions.

2.1 Intuition

In [4] the authors prove that every SM of a NLP corresponds to a stable set of hypotheses, and these correspond in turn to a 2-valued complete, consistent, admissible scenario. In order to guarantee the Existence of a 2-valued total Model for every NLP we allow in all the negative hypotheses which are self-conflict-free and admissible ([6]), and include a Preferred Extension [6]. The extra negative hypotheses approved beyond a Preferred Extension are criteriously allowed (only the approvable ones) to ensure that the resulting 2-valued completion is consistent.

2.2 Underlying Notions

Most of the ideas and notions underlying the work we now present come from the Argumentation field — mainly from the foundational work of Phan Minh Dung in [6] — plus the *Reductio ad Absurdum* reasoning studied in [14], [12], and [13]. For self-containment we now present the basic notions of argument (or set of hypotheses), attack, conflict-free set of arguments, acceptable argument, and admissible set of arguments (all original from [6]).

Definition 7. ***Argument.*** *In [6] the author presents an argument as*

> *"an abstract entity whose role is solely determined by its relations to other arguments. No special attention is paid to the internal structure of the arguments."*

In this paper, since we are focusing on Normal Logic Programs, we will pay attention to the internal structure of an argument by considering an argument *(or set of hypotheses) as a set S of default negated literals of a Normal Logic Program P, i.e., $S \subseteq not\ Atoms(P)$. Thus, a* simple argument *$not\ a$ of S (or simple hypothesis) is just an element of an* argument *S.*

Using these notions of argument *and* simple argument *we can define the set of* Arguments *of a NLP P — $Arguments(P)$ — as the set of all* arguments *of P, i.e., the set of all subsets of $not\ Atoms(P)$.*

Definition 8. ***Attack* — Argument B *Attacks* simple argument $not\ a$ *in* P *[6].*** *In [6] Dung does not specify what the* attacks *relationship concretely is, this way ensuring maximal generality of the argumentation framework. In our present work, since we are considering NLPs, and* arguments *as sets of default negated literals (each a* simple argument*), the* attacks *relationship corresponds to deriving a positive literal contradicting one* simple argument *of the attacked* argument.

Formally, if P is a NLP, $B \in Arguments(P)$, and $not\ a \in not\ Atoms(P)$, we say B attacks $not\ a$ in P iff $P \cup B \vdash a$, i.e., $a \in least(P \cup B)$. For simplicity, we just write $attacks_P(B, not\ a)$.

Abusing this notation, we also write $attacks_P(B, A)$*, where both* A *and* B *are* Arguments *of* P*, to mean that the* Argument B *attacks* Argument A *in* P*. This means that* $\exists_{not\ a \in A} attacks_P(B, not\ a)$*.*

Definition 9. ***Conflict-free argument*** A ***[6].*** *An* argument A *of* P *is said to be conflict-free iff there is no* simple argument $not\ a$ *in* A *such that* $attacks_P(A, not\ a)$*. I.e.,* A *does not attack itself.*

Definition 10. ***Acceptable argument [6].*** *An argument* $A \in AR$ *— where* AR *is a set of* arguments *— of* P *is said to be Acceptable with respect to a set* S *of arguments iff for each argument* $B \in AR$*: if* B *attacks* A *then* B *is attacked by* S*.*

Definition 11. ***Admissible Argument*** A ***of*** P***[6].*** *A conflict-free* argument A *is* admissible *in* P *iff* A *is acceptable with respect to* $Arguments(P)$*. Intuitively,* A *is* admissible *if it counter-attacks every argument* B *in* $Arguments(P)$ *attacking* A*. Formally,*

$\forall_{B \in Arguments(P)} \forall_{not\ a \in A} attacks_P(B, not\ a) \Rightarrow \exists_{not\ b \in B} attacks_P(A, not\ b)$

This definition corresponds to the definition of Admissible Set presented in [6]. Notice that it is not required an attacking set B *to be consistent with its consequences, i.e.,* $least(P \cup B)$ *is not required to be consistent.*

Definition 12. ***Preferred Extension[6].*** *A Preferred Extension is a maximal (with respect to set inclusion) admissible Argument of* P*.*

2.3 Definition of the Approved Models Semantics

Definition 13. ***Approved Models.*** *Let* P *be a NLP and* $M = M^+ \cup M^-$ *a 2-valued interpretation of* P*. We say* M *is an Approved Model of* P *iff:*

- M *is an Approvable Interpretation of* P*, and*
- M^- *contains a Preferred Extension of* P

We use the notation $AM_P(M)$ *to mean that* M *is an Approved Model of* P*. The Approved Models Semantics of a Normal Logic Program* P *is just the intersection of the positive parts of all its Approved Models. We write*

- $AM^+(P)$ *to denote the set of atoms of* P *considered* true *by the Approved Models Semantics. This corresponds to the Approved Models Semantics of* P*, i.e.,* $AM^+(P) = \bigcap_{AM_P(M)} M^+$
- $AM^-(P)$ *to denote the set of atoms of* P *considered* false *by the Approved Models Semantics. I.e.,* $AM^-(P) = not\ \bigcap_{AM_P(M)} M^-$
- $AM^u(P)$ *to denote the set of atoms of* P *considered* undefined *by the Approved Models Semantics. I.e.,* $AM^u(P) = Atoms(P) - \big(AM^+(P) \cup AM^-(P)\big)$

2.4 Properties of the Approved Models Semantics

Stable Models Extension

Theorem 1. ***Every Stable Model is also an Approved Model.*** *If P is a NLP, and SM a Stable Model of P, then $M = SM \cup M^-$, where $M^- = not\ (Atoms(P) \setminus SM)$, is an Approved Model of P.*

Proof. Let P be a NLP, and SM a Stable Model of P. Since every Stable Model SM is a Minimal Herbrand Model of P it means that $not\ (Atoms(P) \setminus SM)$ is Maximal. Therefore, we conclude that $SM \cup not\ (Atoms(P) \setminus SM)$ is an Approvable Interpretation of P.

Moreover, since SM is a Stable Model of P we know, by [6], that $not\ (Atoms(P) \setminus SM)$ is Admissible in P. Since SM is a SM of P, it is minimal and therefore $not\ (Atoms(P) \setminus SM)$ is maximal. Hence, $not\ (Atoms(P) \setminus SM)$ is a Preferred Extension and $SM \cup not\ (Atoms(P) \setminus SM)$ is an Approved Model of P. □

Existence

Theorem 2. ***Every Normal Logic Program has at least one Approved Model.***

Proof. Let P be a NLP. In [6] the author proves that every NLP has at least one preferred extension, and that preferred extensions are Maximal Admissible Arguments. Consider $M'^- \in Arguments(P)$ one such preferred extension. It is always possible to construct one $M^- = M'^- \cup S^-$, where $S^- \in Arguments(P)$, such that $M = M^- \cup M^+$, and $M^+ = Atoms(P) \setminus not\ M^-$, is an Approvable Interpretation. The limit case would be $S^- = \emptyset$ and $M^+ = Atoms(P) \setminus (not\ M^-)$.

The intuitive idea is that no more *simple arguments* can be added to a preferred extension M'^- because, for at least one $not\ s \in S^-$, $M'^- \cup \{not\ s\}$ would not be an Acceptable Argument in the sense of definition 10, so a corresponding positive atom s can be added instead, plus a new maximal addition of default atoms as a result of adding the positive one, for any such positive atom.

The addition of such positive atoms corresponds to resolving an inconsistency that would follow from adding the corresponding default negation literal, and computing the semantics according to definition 4. It is thus a form of reasoning by contradiction and the positive atom can be justified as a positive hypothesis introduced by that reasoning. Indeed, the reason a preferred extension is not complete is, according to Fage's result [8], because there may be some ICONs or OLONs that prevent turning it into an SM.

□

In [13] we showed iterative and incremental methods for calculating the Revision Complete Scenarios. The same methods can be used to calculate the Approved Models for these coincide with the models yielded by the Revision Complete Scenarios. In a nutshell, to calculate a Revision Complete Scenario we start with a set of hypotheses which initially is the set of all default negated literals of the program $H^- = not\ Atoms(P)$. Next, we compute the least model of the program considering those hypotheses, i.e, $least(P \cup H^-)$. Some inconsistencies (pairs $l, not\ l$) will be present in $least(P \cup H^-)$: we select one such l and remove $not\ l$ from H^- thus reducing it. We repeat this process until there are no more inconsistencies in $least(P \cup H^-)$. At that point we just add the remaining positive atoms needed in order to ensure 2-valued completeness and consistency. For greater clarity we show how this process develops with Example 2:

Example 5. **Iteratively calculating an Approved Model.** Recall the example 2 with John, Mary, going to the swimming pool or to the movies.

$$
\begin{array}{ll}
start\_learning\_to\_swim & \leftarrow risk\_drowning \\
risk\_drowning & \leftarrow go\_to\_pool, not\ start\_learning\_to\_swim \\
go\_to\_pool & \leftarrow not\ go\_to\_movies \\
go\_to\_movies & \leftarrow not\ go\_to\_pool
\end{array}
$$

For simplicity, we will use abbreviations for the literals. To iteratively calculate the Approved Models of this program let us do as explained above. First, we start with $H^- = not\ Atoms(P)$, i.e., $H^- = \{not\ sls, not\ rd, not\ gp, not\ gm\}$. Now we compute $least(P \cup H^-) = \{not\ sls, sls, not\ rd, rd, not\ gp, gp, not\ gm, gm\}$. There are several inconsistencies in $least(P \cup H^-)$, we can choose any one default negated literal participating in one inconsistency and remove it from H^-. Take notice that, since we are removing one $not\ l$ from H^- and in the end we want a 2-valued complete model, we will necessarily have l as *true* in that model — we say l was revised from *false* ($not\ l$) to *true* (l). Let us choose to revise $not\ gp$. H^- thus becomes $\{not\ sls, not\ rd, not\ gm\}$. We recompute $least(P \cup H^-) = \{not\ sls, not\ rd, not\ gm, gp, rd, sls\}$. We now repeat the process of choosing any one assumption $not\ l$ participating in an inconsistency to revise it, and update H^- accordingly. We now revise $not\ sls$. H^- is now $H^- = \{not\ rd, not\ gm\}$. The corresponding Approved Model is $M = \{not\ rd, not\ gm, sls, gp\}$ which is not a Stable Model. The only other Approved Model is $M' = \{not\ rd, gm, not\ sls, not\ gp\}$ which is also a Stable Model.

Relevancy

Definition 14. $Rel_P(a)$ ***— Relevant part of NLP*** P ***for the positive literal*** a***.***

The Relevant part of a NLP P for some positive literal a, $Rel_P(a)$ is defined as $Rel_P(a) = \{r \in P : head(r) = a\} \cup \bigcup_{x:r \in P \wedge head(r)=a \wedge (x \in body(r) \vee not\ x \in body(r))} Rel_P(x)$

Intuitively, the relevant part of a program for some atom a is just the set of rules with head a and the rules relevant for each atom in the body of the rules for a. I.e., the relevant part is the set of rules in the call-graph for a.

Definition 15. ***Relevant Semantics.*** *This definition is taken from [5]. A Semantics Sem for NLPs is said to be Relevant iff for every program P and positive literal a of P* $\quad a \in Sem(P) \Leftrightarrow a \in Sem(Rel_P(a))$

Theorem 3. ***The Approved Models Semantics is Relevant.***
$a \in AMS(P) \Leftrightarrow AMS(Rel_P(a))$

Proof. Let P be a NLP. By definition (see above), the semantics of P according to the Approved Models Semantics is the intersection of the positive parts of all the Approved Models of P. I.e., $AMS(P) = AM^+(P) = \bigcap_{AM_P(M)} M^+$

"⇒" Let a be an atom of P, i.e., a positive literal of P, belonging to $AMS(P)$. Then, necessarily $a \in M^+$ for any $AM_P(M)$. We know by definition that $a \in M^+$ iff $a \in least(P \cup M)$, and since the $least$ operator just computes the set of heads of rules

whose body is satisfied — in this case by the interpretation M — a is in M because one of the rules with a as head, i.e. one of the Relevant Rules for a, has its body satisfied by M. Then, in this case, clearly, $a \in AMS(Rel_P(a))$.

"$\Leftarrow$" Consider now that $a \in AMS(Rel_P(a))$. For the same reason — knowing that $a \in M$ iff $a \in least(P \cup M)$ — it is easy to see that any rule $r \in P$ we might add to $Rel_P(a)$ which does not change $Rel_P(a)$, i.e. r is not Relevant for a in P, will not affect the fact that $a \in AMS(Rel_P(a))$ and hence that $a \in AMS(Rel_P(a) \cup \{r\})$. Therefore we can conclude that $a \in AMS(P)$, where $P \supseteq Rel_P(a)$. □

Cumulativity

Definition 16. ***Cumulative Semantics.*** *This definition is taken from [5]. Let P be a Normal Logic Program, and Sem a semantics for NLPs. We say the semantics Sem is Cumulative (or that it enjoys Cumulativity) iff the Semantics of P remains unchanged when any atom considered* true *in the Semantics is added to P as a fact*

$Cumulative(Sem) \Leftrightarrow \forall_P \forall_{a,b} a \in Sem(P) \wedge b \in Sem(P) \Rightarrow a \in Sem(P \cup \{b\})$

Theorem 4. ***The Approved Models Semantics is Cumulative.***

$\forall_P \forall_{a,b} a \in AMS(P) \wedge b \in AMS(P) \Rightarrow a \in AMS(P \cup \{b\})$

Proof. Let P be a NLP. $a \in AMS(P) \wedge b \in AMS(P)$, i.e., both a and b are in all Approved Models of P. By definition of Approved Model we know that every Approved Model M of P satisfies $M = least(P \cup M)$, and since both $a \in M$ and $b \in M$ we know that $P \cup M = P \cup M \cup \{b\}$. Therefore, $least(P \cup M) = least(P \cup M \cup \{b\}) = M \ni a$. Since this hold for every literal a and b, and for every Approved Model M, it also holds for their intersection. □

2.5 Further requirements: Respecting the Well-Founded Model

Although the Approved Models semantics enjoys already some nice properties as seen above, it still lacks a feature important for practical implementations and applicability: the respect for the program's natural stratification. This is the motivation for the further requirements we present next. These, when enforced, guarantee that respect which was already guaranteed by the Revised Stable Models semantics and this is the reason why we relate the Revised Stable Models semantics and the Approved Models semantics.

Definition 17. ***Well-Founded Model of a Normal Logic Program*** P. *According to [3] the* true *atoms of the Well-Founded Model of P (the irrefutably* true *atoms of P) can be calculated as the Least Fixed Point of the Gelfond-Lifschitz operator iterated twice — Γ_P^2. Formally, $WFM^+(P) = lfp(\Gamma_P^2 \uparrow^\omega (\emptyset))$. Also according to [3] the* true *or* undefined *literals of a program can be calculated as the consequences of assuming as true the* true *atoms of the Well-Founded Model. Formally, $WFM^{+u}(P) = \Gamma_P(WFM^+(P)) \supseteq WFM^+(P)$. So, the just* undefined *literals of P are $WFM^u(P) = WFM^{+u}(P) \setminus WFM^+(P)$. And finally, the* false *atoms of the Well-Founded Model of a Program P (the irrefutably* false *atoms of P) are, necessarily, the atoms of P which are neither* true *nor* undefined*. Formally, $WFM^-(P) = Atoms(P) \setminus WFM^{+u}(P)$.*

Definition 18. ***Atom*** a ***respects the subset*** S ***of Atoms of*** P**.** *Let P be a NLP, a an atom of P, and $S \subseteq Atoms(P)$ a subset of Atoms of P. We say a respects S in P iff a remains* true *or* undefined *in the Well-Founded Model of P when the context S is added to P as facts. Formally, $Respects_P(a, S) \Leftrightarrow a \in WFM^{+u}(P \cup S)$*

Definition 19. ***Undefined part of the Model*** M **—** M^u**.** *Let P be a NLP, and M an Approvable Interpretation of P. We write M^u to denote the subset of M^+ of which all atoms are* undefined *in the Well-Founded Model of P. Formally, $M^u = M^+ \cap WFM^u(P)$. These are the only atoms that might come under* reductio ad absurdum*.*

The concept of $US_P(M, a)$, presented next, is based on the core idea of finding the subset of atoms of M^u which influence the truth value of a. To find such set we recur to the Well-Founded Model calculus because, as it is based upon the Γ_P^2 monotonic operator, it draws conclusions from premises according to the implication-based rules of the program. Therefore, when an atom a is concluded *true* by the Well-Founded Model we know that its truth value was calculated using only the part of the program in the call-graph for the atom a. We show this and the following concepts in example 6.

Definition 20. ***Undefined Support of Model*** M ***for atom*** a ***in*** P **—** $US_P(M, a)$**.** *Let P be a Normal Logic Program, M an Approvable Interpretation of P and a an atom of M^+. We write $US_P(M, a)$ to denote the subset of atoms of M^u which support the truth value of a. Formally,*

$US_P(M, a) = \{b \in M^u : \exists_{S \subseteq Atoms(P)} \big(a \in WFM^u(P \cup S) \wedge a \notin WFM^u(P \cup S \cup \{b\})\big)\}$

By this definition we see that b has a critical influence on the undefinedness of the truth value of a under some context S: when b is not added as a fact a remains undefined*, when b is added as a fact a becomes defined (either* true *or* false*, but no longer* undefined*).*

Next, we present the concept of $USS_P(M, a)$. The idea behind it is a stricter form of the previous one. We want $USS_P(M, a)$ to include all the atoms in $US_P(M, a)$ except for those which are in a loop (a call-graph loop) with a. So, roughly, the atoms in $USS_P(M, a)$ are all in strictly lower strata if we considered a kind of stratification where all the atoms in a loop belong to the same strata. To obtain such a set we remove from $US_P(M, a)$ all the atoms b for which a is an element of $US_P(M, b)$. If a depends on b and b depends on a then there is a loop between a and b.

Definition 21. ***Undefined Strict Support of Model*** M ***for atom*** a ***in*** P **—** $USSup_P(M, a)$**.** *Let P be a Normal Logic Program, M an Approvable Interpretation of P and a an atom of M^+. We write $USS_P(M, a)$ to denote the subset of atoms of M^u which support the truth value of a and whose truth value is in turn not influenced by a. Formally,*

$USS_P(M, a) = US_P(M, a) \setminus \{b \in US_P(M, a) : a \in US_P(M, b)\}$

In a nutshell, the atoms in $USS_P(M, a)$ are guaranteed not to be in a loop with a — as happens with b in, for example, a program like $a \leftarrow not\ b \quad b \leftarrow not\ a$*.*

We wish to assume the literals in I to be *true*, hence we add them to P as facts obtaining $P \cup I$. Now we want to calculate what is necessarily *true* and necessarily

false in that resulting program $P \cup I$. For that we calculate the Well-Founded Model of $P \cup I$, namely its Positive and Negative parts. We now say M Respects I iff all the atoms in the Positive part of the Well-Founded Model of $P \cup I$ are also considered *true* by M, and all the atoms in the Negative part of the Well-Founded Model of $P \cup I$ are also considered *false* by M. So, there are no contradictions between M and the Well-Founded Model of $P \cup I$. This is the rationale behind the next definition.

Definition 22. *M **Respects Interpretation** I **in** P — $Respects_P(M, I)$. Let P be a Normal Logic Program, M an Approvable Interpretation of P, and I an arbitrary Interpretation in P. We say M Respects the Interpretation I in P iff the Positive Part of the Well-Founded Model of $P \cup I$ is totally contained in M^+, and the Negative Part of the Well-Founded Model of $P \cup I$ is totally contained in M^-. Formally,*

$Respects_P(M, I) \Leftrightarrow \big(M^+ \supseteq WFM^+(P \cup I)\big) \wedge \big(M^- \supseteq not\, WFM^-(P \cup I)\big)$

Since $M^+ \cap M^- = \emptyset$ it follows trivially that if $Respects_P(M, I)$ then $M^+ \cap WFM^-(P \cup I) = \emptyset$ and $(not\ M^-) \cap WFM^+(P \cup I) = \emptyset$

Example 6. **Respect and Dir-Respect for the $USS_P(M, a)$.** Consider the Normal Logic Program $P =$

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

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

There are four Approvable Interpretations for P: $M_1 = \{a, k, not\ b, not\ t, not\ i\}$, $M_2 = \{b, k, not\ a, not\ t, not\ i\}$, $M_3 = \{a, t, i, not\ b, not\ k\}$, and $M_4 = \{b, t, i, not\ a, not\ k\}$. Models M_1 and M_2 are symmetrical on a and b, and the same happens with M_3 and M_4. So, we will just examine M_1 and M_3 without any loss of information.

Let us calculate the $US_P(M_1, a)$. $M_1^u = \{a, k\} = M_1^+$. If we add k as a fact to P, a will remain *undefined*; however, when we add a as a fact to P, k becomes defined, in this case *true*. Since there are no other possible subsets of M_1^u to consider we have $US_P(M_1, a) = \{a\}$ and $US_P(M_1, k) = \{a\}$. It is now simple to calculate $USS_P(M_1, a) = \emptyset$ and $USS_P(M_1, k) = \{a\}$. Notice that the check for $Respects_P(M_1, USS_P(M_1, a))$ is trivial because $USS_P(M_1, a) = \emptyset$. Also, $Respects_P(M_1, USS_P(M_1, k))$ holds because, as we have already seen, adding $\{a\} = USS_P(M_1, k)$ to P as a fact renders a *true*. The case with M_3 is quite different. $M_3^u = \{a, t, i\}$. $US_P(M_3, a) = \{a\}$, $US_P(M_3, t) = \{a\}$, and $US_P(M_3, i) = \{a, t\}$. $USS_P(M_3, a) = \emptyset$, $USS_P(M_3, t) = \{a\}$, and $USS_P(M_3, i) = \{a, t\}$. Clearly, $Respects_P(M_3, USS_P(M_3, a))$ holds. However, $Respects_P(M_3, USS_P(M_3, t))$ does not hold. It suffices to check that adding $\{a\} \subseteq USS_P(M_3, t)$ to P as a fact, t, which was considered *true* in M_3, becomes now *false* in the $WFM(P \cup \{a\})$.

Definition 23. ***Strict Model** M **of a program** P. Let P be a NLP. An Approvable Interpretation M is Strict in P iff for any atom a in M^+, M Respects its own Undefined Strict Support, i.e. $Strict_P(M) \Leftrightarrow \forall_{a \in M^+} Respects_P(M, USS_P(M, a))$*

Conjecture 1. M^+ is a Revised Stable Model [12] of a NLP P iff M is an Approved Model and a Strict Interpretation of P.

Conjecture 2. $M = M^+ \cup M^-$ is an Approved Model of a NLP P iff there is some Revision Complete Scenario H of P, where $H = H^+ \cup H^-$ and $H^+ \subseteq M^+$ and $H^- = M^-$.

Proofs of these conjectures are beyond the scope of this paper, part of future work.

2.6 Integrity Constraints

Example 7. **Middle Region Politics Revisited.** Recall the Example 3 presented earlier. We are now going to add extra complexity to it.

We already know the two factions which are at odds and their thinking.

$oppression \leftarrow not\ end\_of\_terrorism \quad end\_of\_terrorism \leftarrow oppression$
$terrorism \leftarrow not\ end\_of\_oppression \quad end\_of\_oppression \leftarrow terrorism$

We now combine these two sets of rules with the two following Integrity Constraints (ICs) which guarantee that $oppression$ and $end\_of\_oppression$ are never simultaneously true; and the same happens with terror:

$$falsum \leftarrow oppression, end\_of\_oppression$$
$$falsum \leftarrow terrorism, end\_of\_terrorism$$

As the reader can see, we write ICs not as OLONs of length 1 as it is usual in the Stable Models community, but we write them as simple, normal rules which have as head a special literal $falsum$. This literal is "special" not because the Approved Models Semantics treats it in any differently, but just because the programmer writing the rules uses it only for the purpose of writing ICs, i.e., $falsum$ is not to be used for any other purposes in the program. So far so good, there is still a single joint set of hypotheses resulting in a consistent scenario $\{end\_of\_oppression, end\_of\_terrorism\}$. Still, there is no SM for this program. But introducing either one or both of the next two rules, makes it impossible to satisfy the ICs:

$oppression \leftarrow not\ terrorism \quad terrorism \leftarrow not\ oppression$

In this case all the Approved Models contain the atom $falsum$. There are still no Stable Models for the resulting program. The semantics we propose allows two models for this program, which correspond to the 2-valued complete consistent models, both containing $falsum$. We can discard them on this account or examine their failure to satisfy the ICs.

When posing a query to be solved in a top-down call-graph oriented manner, if the user wants to make sure $falsum$ is not part of the answer, (s)he just needs to conjoin $not\ falsum$ to the query conjunction. When using the SMs Semantics, typically ICs are written as OLONs of length 1 (e.g. $falsum \leftarrow IC\_Body, not\ falsum$). This is done to take advantage of the characteristics of that semantics, in particular to take advantage of the fact that the SMs do not deal with OLONs (in the sense that it does not provide a semantics to them) and, therefore, eliminate the interpretations which would "activate" the OLON by rendering the IC_Body *true*. Since the Approved Models Semantics gives a semantics to all NLPs it has no special features to take advantage from in order to prune undesired interpretations. The programmer just writes the ICs as depicted in the example 7 above and, if (s)he wants to, asks for models without the $falsum$ atom.

3 Conclusions and Future Work

We have presented a new 2-valued semantics for Normal Logic Programs, based upon the previous work on Argumentation by Phan Minh Dung. This new semantics conservatively extends the Stable Models Semantics with the advantage of enjoying three useful properties: guarantee of model Existence for every NLP; Relevancy, allowing for top-down query-driven proof-procedures; and Cumulativity.

Using our semantics, we have showed how Integrity Constraints can be written in a simpler way and dealt with in a very intuitive manner.

All previously known results about the Stable Models Semantics, when it exists, and their good properties are kept intact. The work we presented is based upon the valuable results of years of effort the scientific community has placed in studying the Stable Models Semantics and the Argumentation framework of Dung. We take another step forward, by adding *reductio ad absurdum* to argumentation in NLPs.

References

1. J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In S. Flesca et al., editor, *JELIA*, volume 2424 of *LNCS*, pages 50–61. Springer, 2002.
2. J. J. Alferes and L. M. Pereira. An argumentation theoretic semantics based on non-refutable falsity. In J. Dix et al., editor, *NMELP*, pages 3–22. Springer, 1994.
3. Chitta Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *J. Autom. Reasoning*, 10(3):399–420, 1993.
4. A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93:63–101, 1997.
5. Jürgen Dix. A Classification-Theory of Semantics of Normal Logic Programs: I, II. *Fundamenta Informaticae*, XXII(3):227–255, 257–288, 1995.
6. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.
7. P. M. Dung, R. A. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artif. Intell.*, 170(2):114–159, 2006.
8. F. Fages. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science*, 1:51–60, 1994.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080. MIT Press, 1988.
10. A. C. Kakas and P. Mancarella. Negation as stable hypotheses. In *LPNMR*, pages 275–288. MIT Press, 1991.
11. M. Malý. Complexity of revised stable models. Master's thesis, Comenius University Bratislava, 2006.
12. L. M. Pereira and A. M. Pinto. Revised stable models - a semantics for logic programs. In G. Dias et al., editor, *Progress in AI*, volume 3808 of *LNCS*, pages 29–42. Springer, 2005.
13. L. M. Pereira and A. M. Pinto. Reductio ad absurdum argumentation in normal logic programs. In *Argumentation and Non-monotonic Reasoning (ArgNMR'07) workshop at LPNMR'07*, pages 96–113, 2007.
14. A. M. Pinto. Explorations in revised stable models — a new semantics for logic programs. Master's thesis, Universidade Nova de Lisboa, February 2005.
15. L. Soares. Revising undefinedness in the well-founded semantics of logic programs. Master's thesis, Universidade Nova de Lisboa, 2006.