A Generalised Semantics for Belief Updates —An Equivalence-based Approach

Juan C. Acosta and Jorge Hernández

Computer Systems
UAEH-ESH, Mexico
jguadarrama@gmail.com; jhcjorge@gmail.com

Abstract. As suggested in the literature, revising and updating beliefs and knowledge bases is an important unsolved topic in knowledge representation and reasoning that requires a solid theoretical basis, particularly in current applications of Artificial Intelligence where an agent can work in an open dynamic environment with incomplete information. Various researchers have combined postulates and Answer Set Programming as key components to set up their approaches. However, many of such proposals still present some shortcomings when dealing with persistence situations, redundant information, contradictions or lack of *further properties of knowledge evolution*. In need to satisfy more general principles and suggesting a frame of reference, this paper consists of a study of *consistency preservation and restoration*, and a new *AGM-characterisation* of a semantics for updates of logic programs. It consists in performing updates of *epistemic states* that meets well-accepted *belief revision postulates*. Besides the set of properties that this framework shares with other equivalent update semantics, this proposal is also supported by a solver prototype as an important component of logic programming and automatic testbed of its declarative version. The existence of a solver for a theoretical framework helps to automatically compute agents' knowledge bases for more complex prototypes and frameworks.

1 Introduction

One of the goals of Artificial Intelligence and in particular of commonsense reasoning is how to make an agent intelligent that may be autonomous and capable of acting in an open dynamic environment. As suggested in the logic-programming literature, such a goal requires a solid theoretical basis on knowledge representation and nonmonotonic reasoning, and in particular, in *knowledge updates*. Logic programming is a classical well-known mechanism to code and represent agents' knowledge by means of a set of clauses called logic program. Such a program might be called a knowledge base and we code it into a semantics called Answer Sets Programming [15] or ASP in short. However, logic programming has typically been static in the sense that it provides no mechanism to automatically make changes (belief revision or updates) to the knowledge base.

In particular, when updating knowledge one needs a way to avoid inconsistencies due to *potential contradictory information* upcoming from new evidence

that is typically incomplete. Much work has been done in the context of logic programming based on a common ASP basis by satisfying certain properties and postulates: [1, 7, 12, 14, 22, 23, 24, 25]. However, despite the existence of several semantics for updates [3] and a vast analysis of general properties [14, 21], we claim we are still far from having a general one that can satisfy many existing and well-known principles to represent *"correct" dynamic knowledge*.

For instance, one of the missing and obvious properties in current semantics for updates is *persistence* that others don't manage well for several reasons, mainly for their approach on *sequences of updates* [3]. Such a problem has been introduced and overcome by Sakama and Inoue [23] by means of a semantics based on an extended particular version of abductive logic programming [16]. However, their semantics is strongly based on syntactical changes that has other problems (described later) and lacks of a proper characterisation of principles for updates (or belief revision), presumably due to their different goals of their referred approach.

With the aim to define a *general characterised semantics* and to succeed in the mentioned persistence situation (and in many others more), this paper consists of an alternate more-general approach, founded on generalised answer sets, and based upon further well-known principles for belief change (AGM-postulates [6]) that make it *syntax independent*, more intuitive and have generally-accepted properties.

Besides satisfying most belief revision postulates, this paper exhibits an approach that consists in performing *successive updates* so that the semantics can deal with the problem that, according to Sakama and Inoue, produces counter-intuitive interpretations in most approaches.

This paper is organised as follows: Section 2 is an analysis of the problem that Sakama and Inoue pointed out, while Section **??** consists of the general basic terminology for the current framework. The core of the paper is Section 4, which includes consistency properties and postulates for belief revisionas well as a slightly different formulation to the approaches presented in [1, 5] and shows the main results and comparison with its predecessors in Section 5 . Finally, concluding remarks are discussed in Section **??**.

Partial results of this article have already appeared in preliminary versions (without proofs or other properties) in [2], as well as in the extended abstract in [4]

2 Problem Description

As introduced in previous paragraphs, once there existed a strong theoretical *basis for belief revision and updates*, a few authors in the field of logic programming proposed mechanisms for updates with a vast analysis, *taxonomy* and comparison of various known semantics [14, 23, 25]. Some others even suggested new principles, like [7, 8], with alternate logic-programming frameworks to ASP. However, owing to the different foundations each semantics has (even within ASP),

yet some problems arise to meet a significant number of well-accepted *principles for updates* [3].

Although it is unlikely to come off with a semantics that may satisfy all of them [14, 21, 23], a more general one that fulfils most of widely-accepted principles is still necessary, which is not an easy task.

For instance, researchers studying updates of logic programs following principles for updates[1] (coded into eight well-known postulates for updates called AGM theory [6]) —and in other principles around it[2]— boil down to the difficulty in satisfying many of them by means of a *non-monotonic framework* like ASP, owing to the *monotonic nature* of the postulates themselves —[10, 14, 21, 23]. Nonetheless, Osorio and Cuevas [21] achieved an interpretation of six of the original eight AGM postulates in terms of the *monotonic* (non-classical) $\mathcal{N}_2$-logic, for general "update" operators. They have chosen (the monotonic) $\mathcal{N}_2$-logic apparently because it is one of several that characterise ASP and includes two types of negation: negation-as-failure and strong negation. The authors' results in terms of a general semantics, however, seem to be inconclusive [21].

On the other hand, the nearest proposal (to the best of our knowledge) that seemed[3] to meet most of the existing principles is due to Sakama and Inoue [23], who have introduced and overcome an interesting *persistence situation* that others fail to represent well for different reasons, as pointed out by themselves and in [2]. In particular, the main feature of that interesting situation is that Sakama and Inoue's semantics is capable of maintaining a *knowledge base* (coded into a logic program) throughout its own evolution, and that is also the main motivation behind A. Guadarrama [2] to propose his approach. They both lack, however, of a more-general belief-update characterisation, besides other problems.

Although their object-level approach makes it a good candidate to be considered by the belief-revision community, Sakama and Inoue's minimal-change principle is still *syntactic*: the *changes to a knowledge base* are to be minimal, as they themselves explain, and that has other problems. The main issue, however, in that they have no characterisation of their semantics with further general *belief-change principles*, arguing that they aren't applicable to nonmonotonic propositional theories in general [23].

In addition to that, a big disadvantage of *syntactical approaches*, as discussed in [14, 21] is that, in general, they do not satisfy the *structural properties* proposed in [21, 24], and Sakama and Inoue's approach is not an exception.

Regardless the polemic that approach might cause (especially in *planning* domains) and the deployment of extended-abduction properties in their article, the lack of further and more *general properties for belief change* makes it hard to compare with other alternatives for updates of logic programs. Indeed, their

[1] Actually the principles are for belief revision rather than updates. We explain the difference later.

[2] For a nice analysis and compilation of such principles, see [14].

[3] To my knowledge, there is no evidence that their semantics satisfies most of them, although it does overcome most of the problems that other semantics present.

first goal (as they themselves explain) is the converse: to provide a mechanism of updates to characterise their *extended abduction framework* [23].

In need to define a *general semantics* and regardless the difference between belief revision and updates due to Katsuno and Mendelzon [19], this paper includes further results and a slightly different and fundamental approach from the preliminary alternate basic solution in [2, 4] within the same studied foundation of *Minimal Generalised Answer Sets* (MGAS hereafter, from Kakas and Mancarella [17]) and with an alternate approach to both Zacarías et al. [24] and A. Guadarrama [1], proposing a *simpler general formulation* that likewise performs multiple updates, but at the object level rather than sequences of updates, which overcomes the kind of problems already described. Moreover, the simpler semantics meets the *structural properties for updates* proposed by [14, 21, 24], as well as the *satisfaction of five of the six most general belief revision postulates* and many other relevant properties.

3 Preliminaries

A main foundation of this proposal is the well-known AGM-postulates [6] in a particular interpretation and notation [4], followed by a brief basic background of Answer Sets and Generalised Answer Sets. Owing to space constraints, however, this paper excludes sections of three ASP's characterising logic systems (intuitionistic logic, Nelson's logic and $\mathcal{N}_2$), which are more evidence of the solid foundation earlier suggested. Finally, in this paper it is assumed that the reader is familiar with basic notions of AGM-theory, as well as logic programming and in particular with ASP.

3.1 Logic Programming and Answer Sets

The following formalism gives a short recap of ASP, which is identified with other names like *Stable Logic Programming* or *Stable Model Semantics* [15] and A-Prolog. Its formal language and some more notation are introduced as follows.

Definition 1 (ASP Language of logic programs, $\mathcal{L}_{\mathsf{ASP}}$). *In the following $\mathcal{L}_{ASP}$ is a language of* propositional logic *with* propositional symbols*: $a_0, a_1, \ldots$;* connectives*: "," (*conjunction*) and meta-connective ";";* disjunction*, denoted as "|"; $\leftarrow$ (*derivation*, also denoted as $\rightarrow$);* propositional constants $\bot$ *(*falsum*); $\top$ (*verum*); "$\neg$" (*default negation *or* weak negation*, also denoted with the word* not*); "$\sim$" (*strong negation*, equally denoted as "$-$"); *auxiliary symbols*: "(", ")" (*parentheses*). The propositional symbols are called* atoms *too or* atomic propositions*. A* literal *is an atom or a* strong-negated atom*. A* rule *is an ordered pair $\mathit{Head}(\rho) \leftarrow \mathit{Body}(\rho)$.*

An intuitive meaning of *strong negation* "$\sim$" in logic programs with respect to the default negation "$\neg$" is the following: a rule $\rho_0 \leftarrow \neg\rho_1$ allows to derive ρ_0 when there is no *evidence* of ρ_1, while a rule like $\rho_0 \leftarrow \sim\rho_1$ derives ρ_0 only when there is an *evidence* for $\sim\rho_1$, i.e. when it can be proved that ρ_1 is false.

With the notation introduced in Definition 1, one may construct clauses of the following general form that are well known in the literature.

Definition 2 (EDLP). *An* extended disjunctive logic program *is a set of rules of form*

$$\ell_1 \vee \ell_2 \vee \ldots \vee \ell_l \leftarrow \ell_{l+1}, \ldots, \ell_m, \neg\ell_{m+1}, \ldots, \neg\ell_n \tag{1}$$

where ℓ_i *is a literal and* $0 \leq l \leq m \leq n$.

Naturally, an *extended logic program* (or ELP hereafter) is a finite set of rules of form (1) with $l = 1$; while an *integrity constraint* (also known in the literature as *strong constraint*) is a rule of form (1) with $l = 0$; while a *fact* is a rule of the same form with $l = m = n$. In particular, for a literal ℓ, the *complementary literal* is $\sim\ell$ and vice versa; for a set $\mathcal{M}$ of literals, $\sim\mathcal{M} = \{\sim\ell \mid \ell \in \mathcal{M}\}$, and $Lit_{\mathcal{M}}$ denotes the set $\mathcal{M} \cup \sim\mathcal{M}$; finally, a *signature* $\mathfrak{L}_{\Pi}$ is a finite set of literals occurring in Π. Additionally, given a set of literals $\mathcal{M} \subseteq \mathcal{A}$, the complement set $\overline{\mathcal{M}} = \mathcal{A} \setminus \mathcal{M}$.

Although we have introduced ASP as propositional (*ground*) programs, fixed *non-ground ASP-programs* of arbitrary *arity* are also considered in the same way than Dantsin et al. [11] do. Accordingly, non-ground ASP-programs with variables or constants as *arguments* are seen as a simplified expressions of larger ground (propositional) ones without variables, where each *ground program* Π is a set of its *ground rules* $\rho \in \Pi$. In addition, a ground rule is the set obtained by all possible substitutions of variables in ρ by constants occurring in Π [11].

Although ASP is a strong theoretical framework to represent knowledge, it can't model changes to that knowledge by itself, nor can it represent conflict situations amongst an agent and its dynamic environment. So, a more general and relaxed semantics is necessary, to choose potential models amongst *conflicting information*, which ought to reflect the general *principles* and *postulates* under consideration. In particular, a suitable framework to our purposes has been *Abductive Logic Programming* due to Kakas and Mancarella and is briefly presented in the following section.

3.2 Abductive Programs and MGAS

As one of the semantics to interpret abductive programs, *Minimal Generalised Answer Sets* (MGAS) provides a more general and flexible semantics than standard ASP, with a wide range of applications. This framework is briefly introduced in the following set of definitions.

Definition 3 (17). *An* abductive logic program *is a pair* $\langle\, \Pi, \mathcal{A}^* \,\rangle$ *where* Π *is an arbitrary program and* $\mathcal{A}^*$ *a set of literals, called abducibles.*

On the other hand, there already exists a semantics to interpret abductive programs, called *generalised answer sets* (GAS) due to Kakas and Mancarella.

Definition 4 (GAS, 17). *The expression* $\mathcal{M}(\Delta)$ *is a* generalised answer set *of the abductive program* $\langle\, \Pi, \mathcal{A}^* \,\rangle$ *if and only if* $\Delta \subseteq \mathcal{A}^*$ *and* $\mathcal{M}(\Delta)$ *is an answer set of* $\Pi \cup \{\alpha \leftarrow \top \mid \alpha \in \Delta\}$.

In case there are more than one generalised answer sets, a *preferred inclusion order* may be established:

Definition 5 (9). *Let $\mathcal{M}(\Delta_1)$ and $\mathcal{M}(\Delta_2)$ be generalised answer sets of $\langle \Pi, \mathcal{A}^* \rangle$. The relation $\mathcal{M}(\Delta_1) \leq_{\mathcal{A}^*} \mathcal{M}(\Delta_2)$ holds if and only if $\Delta_1 \subseteq \Delta_2$.*

Last, one can easily establish the *minimal generalised answer sets* from an abductive inclusion order with the following definition

Definition 6 (MGAS, 9). *Let $\mathcal{M}(\Delta)$ be a* minimal generalised answer set *(MGAS) of $\langle\, \Pi, \mathcal{A}^* \,\rangle$ if and only if $\mathcal{M}(\Delta)$ is a generalised answer set of $\langle\, \Pi, \mathcal{A}^* \,\rangle$ and it is minimal with respect to its* abductive inclusion order*.*

4 Updating Epistemic States

One of the main goals of this proposal is to meet most well-accepted *principles* for updates at the *object level* and in Minimal Generalised Answer Sets (MGAS), besides other relevant properties. The approach consists in setting up the needed models for the desired properties in an *iterated fashion*, rather than a sequence of updates, as earlier explained.

A first analysis of the problem at the object level, a solution, *justification, basic model-oriented properties* and *comparison with other semantics* are available in [2]. However, the semantics hasn't been characterised with more general principles, which is necessary both to avoid counterintuitive behavuour and to provide a common *frame of reference* to compare with other approaches. So, let us briefly introduce it, followed by a *characterisation of Belief Revision.*

The semantics is formally expressed with the following set of definitions, revised from [2] to make it simpler, precise, and to comply even more with the postulates, which is part of the *contribution of this paper* and the difference with the one in [2] and the extended abstract in [4]. So, let us start with some definitions.

An *α-relaxed rule* is a rule ρ that is *weakened* by a default-negated atom α in its body: $\mathsf{Head}(\rho) \leftarrow \mathsf{Body}(\rho) \cup \{\neg\alpha\}$. In addition, an *$\alpha$-relaxed program* is a set of α-relaxed rules. On the other hand, a *generalised program* of $\mathcal{A}^*$ is a set of rules of form $\{\ell \leftarrow \top \mid \ell \in \mathcal{A}^*\}$, where $\mathcal{A}^*$ is a given set of literals.

Accordingly, updating a program with another consists in transforming an ordered pair of programs into a single abductive program, as follows.

Definition 7 ($\bullet$-update Program). *Given an updating pair of extended logic programs, denoted as $\Pi_1 \bullet \Pi_2$, over a set of atoms $\mathcal{A}$; and a set of unique abducibles $\mathcal{A}^*$, such that $\mathcal{A} \cap \mathcal{A}^* = \emptyset$; and the α-relaxed program Π' from Π_1, such that $\alpha \in \mathcal{A}^*$; and the abductive program $\Pi_{\mathcal{A}^*} = \langle\, \Pi' \cup \Pi_2, \mathcal{A}^* \,\rangle$. Its* $\bullet$-update program *is $\Pi' \cup \Pi_2 \cup \Pi_G$, where Π_G is a generalised program of $\mathcal{M} \cap \mathcal{A}^*$ for some* minimal generalised answer set *$\mathcal{M}$ of $\Pi_{\mathcal{A}^*}$ and "$\bullet$" is the corresponding update operator.*

Obviously, Definition 7 allows none or more $\bullet$-*update programs.* Let us formalise yet another minor obvious property:

Corollary 1. *Let Π_G be a generalised program out of a minimal generalised answer set $\mathcal{M}$ from $\Pi_{\mathcal{A}^*}$ and $\mathcal{M}_1$ an answer set of Π_G. The following two statements hold:*

a) $\mathcal{M}_1 = \mathcal{M} \cap \mathcal{A}^*$.
b) $\mathcal{M}_1 \subseteq \mathcal{M}$.

Last but not least, the associated *models $\mathcal{S}$ of the new knowledge base* correspond to the answer sets of a $\bullet$-update program as follows.

Definition 8 ($\bullet$-update Answer Set). *Let $\mathbf{\Pi}_\bullet = (\Pi_1 \bullet \Pi_2)$ be an update pair over a set of atoms $\mathcal{A}$. Then, $\mathcal{S} \subseteq \mathcal{A}$ is a* $\bullet$-answer set of $\mathbf{\Pi}_\bullet$ *if and only if $\mathcal{S} = \mathcal{S}' \cap \mathcal{A}$ for some* minimal generalised answer set $\mathcal{S}'$ *of its $\bullet$-update program.*

Intuitively, this formulation establishes an order with respect to the *latest update* —which corresponds to Katsuno and Mendelzon [18, 19]'s first postulate (R∘1)— and with respect to a *minimal change* when choosing the most preferred model: MGAS.

5 Properties

The following sets of properties of this simpler formulation are the main contribution of this current semantics for *iterated updates* of *epistemic states*. They are classified into a study of equivalence with other variants and the satisfaction itself of KM′-postulates.

A basic set of *structural properties* was first introduced in [13, 14, 24], just like the formulation in [2], besides other properties from the literature. Accordingly, this section is divided into a comparison with operators $\oslash$ and $\otimes$ (Section 5.1); Section **??** generalises the approach from [2] in a set of more *general principles*; and finally, other relevant properties from the literature are presented in Section **??**.

5.1 Equivalence

One contribution of this paper is to show that this approach coincides with operator $\oslash$ in [24] for the case of *single updates* and single updates with cardinality preference. The main difference, however, is the limitation to deal with *multiple updates*.

Before starting with equivalence between operators, a recapitulation of their respective definitions is in order: From [24], the update operator for *pairs of programs* in the current notation is

Definition 9 ($\oslash$-Update Program [24]). *Given an update pair $\mathbf{\Pi} = (\Pi_1, \Pi_2)$ of extended logic programs over a set of atoms $\mathcal{A}$, an* update program $\mathbf{\Pi}_\oslash = \Pi_1 \oslash \Pi_2$ *corresponds to the abductive program $\langle\, \Pi' \cup \Pi_2, \mathcal{A}^* \,\rangle$, where $\mathcal{A}^*$ extends $\mathcal{A}$ by new unique abductive atoms and Π' is constructed as follows:*

(i) *all constraints in* Π_1.
(ii) *for each non-constraint rule* $\rho \in \Pi_1$ *there is a unique abducible* α *(a new atom) and the rule is replaced by* $\mathsf{Head}(\rho) \leftarrow \mathsf{Body}(\rho), \neg\alpha$.

where $\oslash$ *represents the the corresponding* update operator.

Next, the corresponding update models come out of the MGAS's from the abductive program as follows.

Definition 10 ($\oslash$-update Answer Set [24]). *Let* $\mathbf{\Pi} = (\Pi_1, \Pi_2)$ *be an update pair over a set of atoms* $\mathcal{A}$. *Then,* $\mathcal{S} \subseteq \mathcal{A}$ *is an* update answer set *of* $\mathbf{\Pi}$ *if and only if* $\mathcal{S} = \mathcal{S}' \cap \mathcal{A}$ *for some* minimal generalised answer set $\mathcal{S}'$ *of* $\mathbf{\Pi}$.

Finally, operators $\bullet$ and $\oslash$ are equivalent in pairs of programs, as formally expressed in the following proposition.

Proposition 1. *[[2]] Suppose an initial extended logic program* Π_1, *without integrity constraints, updated with any* Π_2. *Then,*

$$\Pi_1 \oslash \Pi_2 \equiv \Pi_1 \bullet \Pi_2$$

As a consequence, this semantics inherits most of the assets from $\oslash$: the characterisation of *model-content updates* that preserves the system from being *counterintuitive* when particular situations of updating with *tautological (inert) information* arise, as well as the set of *structural properties* listed in [2].

Before the main results, there are two particular properties suggested much earlier in [24] that are necessary for the rest of them. Additionally, the reader should note that a statement like $\Pi_1 \equiv \Pi_2$ means that both Π_1 and Π_2 have the same answer sets —or alternately $\Pi_1 \equiv_{\mathsf{ASP}} \Pi_2$. By a slight abuse of notation, when establishing equivalence between updates, indeed it means that they have the same (or different) update answer sets. Finally, the two properties from the literature (ref. [20, 21]), interpreted in our own notation, are the following.

$\bullet$-SP-8, Strong Consistency, SC: If $\Pi_1 \cup \Pi_2$ is *consistent*, then $\Pi_1 \bullet \Pi_2 \equiv \Pi_1 \cup \Pi_2$. The update coincides with the union when $\Pi_1 \cup \Pi_2$ is consistent.
$\bullet$-SP-9, Weak Irrelevance of Syntax, WIS: Let Π, Π_1, and Π_2 be logic programs under the same language. If $T_{\mathcal{N}_2}(\Pi_1) \equiv_{\mathcal{N}_2} T_{\mathcal{N}_2}(\Pi_2)$ then $\Pi \bullet \Pi_1 \equiv \Pi \bullet \Pi_2$.

Theorem 1. *[[2]] Suppose that* Π, Π_1, Π_2 *and* Π_3 *are* ELP. *Operator* $\bullet$ *satisfies the properties* $\bullet$-SP-8 *and* $\bullet$-SP-9.

This preliminary result shall be helpful to simplify further properties in upcoming sections.

5.2 Discussion

This section is an introduction to new general properties characterising $\bullet$-operator that go from the structural properties, most of them inherited from its equivalent counterpart in [24], as $\oslash$ and $\otimes$-operators to more general ones encoded

in our particular interpretation of belief revision postulates. The satisfaction of AGM-postulates in ASP is something new and important, provided that other current approaches either don't meet most of them or have discarded them for considering that their *monotonic nature* is incompatible with non-monotonic frameworks like ASP, as previously discussed in Section **??**.

Another issue other approaches have is when updating in a sequence rather than an iterated fashion, which leads to counterintuitive results, especially in the persistence situation and when new updates arise afterwards. By following the original AGM paradigm, we also claim that the iterative approach has other more natural properties than its sequenced counterpart.

The section is also a *study of inconsistencies* not only due to new information that contradicts current knowledge, but also from both an *originally-inconsistent knowledge base*, as well as *new originally-inconsistent observations* that not necessarily contradict current beliefs. The former is something that may be considered a key feature of belief revision. However, as one of the main goals of this paper is to provide a strong general framework to correctly represent knowledge, and making a strict distinction with belief update theory might result controversial.

On the other hand, dealing with *originally-inconsistent observations* might seem counterintuitive to some researches, but it does not mean that observing such contradictions may not be possible in a *changing environment*. Take for example two *concurrent observations* that contradict each other, updating a current knowledge base in, say, a problem of Ambient Intelligence when a sensor fails and another one contradicts it. Another example is an *observation that is inconsistent* due to a typo or another kind of human error. Traditionally, those problems are left to future debugging, but with a tendency to model even-more *autonomous entities*, tolerating inconsistencies is not only reasonable, but also necessary to preserve a knowledge base from collapse.

6 Conclusions and Related Work

This paper consists of a *generalisation* of $\bullet$-operator that satisfies five of six suitable *belief-revision postulates* for *updating epistemic states*, as well as other useful properties from previous proposals, towards a more general formulation to update logic programs. Additionally, this paper has introduced a study of *consistency preservation* and *consistency recovery* (also known as *consistency restoration*) as a very important issue to be considered when updating logic programs that may help achieve particular needs and preserve the epistemic state from collapse. As a result, this framework provides a *strong theoretical foundation* on well-known principles and other fundamental properties shared mainly with other operators. Although $\bullet$-operator overcomes the persistency problems that Sakama and Inoue have pointed out, because of its *object-level approach*, both proposals have lacked of a general and fundamental *characterisation of Belief-revision* postulates, and is here provided as a *frame of reference* and *comparison*. By combining the operational features of ASP, its characterising logics and non-

monotonic theory that underpin it, and a broad set of well-known belief-change principles, one should be capable of configuring agents' knowledge bases that are well-behaved and robust-enough against unexpected circumstances. Finally, as a classical component of Logic Programming, this operator has an implemented *solver prototype*[4] as an approximation and automatic testbed that makes the semantics more *accessible* (in a classroom, i.e.), and potential component for further more complex prototypes in administration of (toy?) knowledge systems, with precise properties.

7 Acknowledgements

This work has been supported by a SEP project.

[4] The prototype is available at `http://www.in.tu-clausthal.de/~guadarrama/updates/o.html`.

Bibliography

[1] A. Guadarrama, J. C. 2007a. Implementing knowledge update sequences. In *MICAI 2007: Advances in Artificial Intelligence*, A. Gelbukh and A. Kuri Morales, Eds. LNCS, vol. 4827. Springer-Verlag, Aguascalientes, Mexico, 260–270. 2, 4

[2] A. Guadarrama, J. C. 2007b. Maintaining knowledge bases at the object level. In *Special Session of the 6th International MICAI Conference*, A. Gelbukh and A. F. Kuri Morales, Eds. IEEE Computer Society, Aguascalientes, Mexico, 3–13. ISBN: 978-0-7695-3124-3. 2, 3, 4, 6, 7, 8

[3] A. Guadarrama, J. C. 2007c. A road map of updating in ASP. ISSN: 1860-8477 IfI-07-16, Institute für Informatik, TU-Clausthal, Clausthal, Germany. December. 35pp. 2, 3

[4] A. Guadarrama, J. C. 2008. AGM postulates in answer sets. In *LANMR'08 Fourth Latin American Workshop on Non-monotonic Reasoning*, M. Osorio and I. Olmos, Eds. ISSN 1613-0073, vol. 408. CEUR, Benemérita Universidad Autónoma de Puebla, Puebla, México, 3pp. 2, 4, 6

[5] A. Guadarrama, J. C., Dix, J., and Osorio, M. 2006. Update Sequences in Generalised Answer Set Programming Based on Structural Properties. In *Special Session of the 5th International MICAI Conference*, P. Kellenberger, Ed. IEEE Computer Society, Mexico City, Mexico, 32–41. ISBN: 0-7695-2722-1. 2

[6] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic 50,* 2 (June), 510–530. 2, 3, 4

[7] Alferes, J. J., Banti, F., Brogi, A., and Leite, J. A. 2005. The refined extension principle for semantics of dynamic logic programming. *Studia Logica 79,* 1, 7–32. 2

[8] Alferes, J. J., Leite, J. A., Pereira, L. M., Przymusinska, H., and Przymusinski, T. C. 1999. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming 45,* 1–3, 43–70. 2

[9] Balduccini, M. and Gelfond, M. 2003. Logic programs with consistency-restoring rules. In *Proceedings of the AAAI Spring 2003 Symposium.* AAAI Press, Palo Alto, California, 9–18. 6

[10] Brewka, G. 2001. Declarative representation of revision strategies. *Journal of Applied Non-classical Logics 11,* 1–2, 151–167. 3

[11] Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys 33,* 3 (September), 374–425. 5

[12] Delgrande, J. P., Schaub, T., Tompits, H., and Woltran, S. 2008. Belief revision of logic programs under answer set semantics. In *KR*. 411–421. 2

[13] Eiter, T., Fink, M., Sabbatini, G., and Tompits, H. 2000. Considerations on updates of logic programs. In *Logics in Artificial Intelligence, Euro-*

pean Workshop, JELIA 2000, M. Ojeda-Aciego, I. P. de Guzmán, G. Brewka, and L. Moniz Pereira, Eds. Springer Verlag, Malaga, Spain, 2–20. 7

[14] Eiter, T., Fink, M., Sabbatini, G., and Tompits, H. 2002. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming 2,* 6, 711–767. 2, 3, 4, 7

[15] Gelfond, M. and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium ICLP/SLP*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press, Seattle, Washington, 1070–1080. 1, 4

[16] Inoue, K. and Sakama, C. 1995. Abductive framework for nonmonotonic theory change. In *the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*. Morgan Kaufmann Publishers, Montreal, Canada, 204–210. 2

[17] Kakas, A. C. and Mancarella, P. 1990. Generalized Stable Models: A semantics for abduction. In *ECAI*. Stockholm, Sweden, 385–391. 4, 5

[18] Katsuno, H. and Mendelzon, A. O. 1989. A unified view of propositional knowledge base updates. In *the 11th International Joint Conference on Artificial Intelligence, IJCAI-89*, N. S. Sridharan, Ed. Morgan Kaufmann, Detroit, Michigan, USA, 1413–1419. 7

[19] Katsuno, H. and Mendelzon, A. O. 1991a. On the difference between updating a knowledge base and revising it. In *KR'91*. Morgan Kaufmann Publishers, Cambridge, Massachusetts, USA, 387–394. 4, 7

[20] Katsuno, H. and Mendelzon, A. O. 1991b. Propositional knowledge base revision and minimal change. *Artificial Intelligence 52,* 3, 263–294. 8

[21] Osorio, M. and Cuevas, V. 2007. Updates in answer set programming: An approach based on basic structural properties. *Journal of Theory and Practice of Logic Programming 7,* 4, 451–479. 2, 3, 4, 8

[22] Osorio, M. and Zacarías, F. 2004. On updates of logic programs: A properties-based approach. In *FoIKS*. Springer, Wilhelminenburg Castle, Austria, 231–241. 2

[23] Sakama, C. and Inoue, K. 2003. An abductive framework for computing knowledge base updates. *Theory and Practice of Logic Programming 3,* 6, 671–715. 2, 3, 4, 9

[24] Zacarías, F., Osorio, M., A. Guadarrama, J. C., and Dix, J. 2005. Updates in Answer Set Programming Based on Structural Properties. In *7th International Symposium on Logical Formalizations of Commonsense Reasoning*, S. McIlraith, P. Peppas, and M. Thielscher, Eds. Fakultät Informatik, ISSN 1430-211X, Corfu, Greece, 213–219. 2, 3, 4, 7, 8

[25] Zhang, Y. and Foo, N. 2005. A unified framework for representing logic program updates. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, M. M. Veloso and S. Kambhampati, Eds. AAAI Press / The MIT Press, Pittsburgh, Pennsylvania, USA, 707–713. 2