A Unifying Perspective on Knowledge Updates

Martin Slota and João Leite

CENTRIA & Departamento de Informática
Universidade Nova de Lisboa
Quinta da Torre
2829-516 Caparica, Portugal

Abstract. We introduce an abstract update framework based on viewing a knowledge base as the *set of sets of models* of its elements and performing updates by introducing additional interpretations – *exceptions* – to the sets of models of elements of the original knowledge base. In [36], an instantiation of this framework for performing rule updates has been shown to semantically characterise one of the syntax-based *rule update semantics*. In this paper we show that the framework can also capture a wide range of both model- and formula-based belief update operators which constitute the formal underpinning of existing approaches to *ontology updates*. Exception-driven operators thus form a unifying perspective on both ontology and rule updates, opening new possibilities for addressing *updates of hybrid knowledge bases* consisting of both an ontology and a rule component.

1 Introduction

In this paper we propose a novel generic method for specifying update operators. By viewing a knowledge base as the *set of sets of models of its elements*, and seeing updates as *adding new interpretations* to those sets, we are able to capture a range of model- and formula-based belief update operators. When coupled with the results of [36] in which an instantiation of this framework was shown to characterise a syntax-based rule update semantics, our findings imply that exception-driven operators are the first approach that embraces these two seemingly irreconcilable approaches to updates.

Throughout the last decade, standardisation efforts gave rise to widely accepted knowledge representation languages such as the Web Ontology Language (OWL)[1] and Rule Interchange Format (RIF),[2] based on Description Logics [4] and Logic Programming [16], respectively. This has fostered a large number of ontologies and rule bases with different levels of complexity and scale. Whereas ontologies provide the logical underpinning of intelligent access and information integration, rules are widely used to represent business policies, regulations and declarative guidelines about information.

Since both ontologies and rules offer important features for knowledge representation, considerable effort has been invested in identifying a unified hybrid knowledge framework where expressivity of both formalisms could be seamlessly combined. This task turned out to be very challenging because of the inherent semantic differences between the two knowledge representation paradigms.

[1] http://www.w3.org/TR/owl-overview/
[2] http://www.w3.org/2005/rules/wiki/RIF_Working_Group

Over the years, work on hybrid knowledge bases has matured significantly and fundamental semantic as well as computational problems were addressed successfully (see [20] for an overview). The recent formalisms, based on an embedding to a unifying non-monotonic formalism, such as the Autoepistemic Logic [6] or the Logic of Minimal Knowledge and Negation as Failure (MKNF) [27], provide a tight and semantically neat integration of ontologies and rules, allowing predicates to be defined concurrently in the ontology as well as by rules. Nevertheless, they only deal with *static knowledge*.

One of the main challenges for knowledge engineering and information management is to efficiently and plausibly deal with the incorporation of new, possibly conflicting knowledge and beliefs. In other words, support for *knowledge dynamics* is essential. This topic has been extensively addressed in the context of both Description Logics and Logic Programs, when taken separately.

Ontology Updates. The area of research called *ontology change* encompasses a number of strongly related though distinguishable subareas, such as ontology matching, ontology integration and merging, or ontology translation (a survey can be found in [14]). The purest type of change, concerned with modifications to a single ontology, is generally referred to as *ontology evolution*. Approaches to ontology evolution with a firm semantic underpinning, thus amenable to a formal analysis of their behaviour and properties, are based on research in the area of *belief change*, initiated by the seminal work of Alchourrón, Gärdenfors and Makinson (AGM) [1] who proposed a set of desirable properties of change operators on monotonic logics, now called *AGM postulates*.

Subsequently, *revision* and *update* were distinguished as two very related but ultimately different belief change operations [39,21,28]. While revision deals with incorporating new information about a *static world* into a knowledge base, update takes place when a knowledge base needs to be brought up to date when the *modelled world changes*. While AGM postulates were deemed appropriate for describing revision, a different set of postulates was suggested for belief update [21,28].

Update operators based on these postulates, usually referred to as *model-based*, were later used to partially address ontology updates [25,10], namely to update the part of the ontology with assertions about individuals (the *ABox*). On the other hand, model-based operators are considered inappropriate for updating ontological axioms that define the terminology (the *TBox*) [7,34]. Their antipole, *formula-based operators*, which manipulate the knowledge base at a syntactic level and are strongly related to *base revision operators*, were adopted for performing TBox updates instead [7].

Rule Updates. When updates were tackled in the context of Logic Programming, it was only natural to consider adapting the belief update postulates and operators to deal with them. However, this led to counterintuitive results because the model-based approach fails to capture the essential relationships between literals encoded in rules [23], and the formula-based approach is too crude as it does not allow rules to be reactivated when reasons for their suppression disappear [40]. Although state-of-the-art approaches to rule updates are guided by the same basic intuitions and aspirations as belief updates, they build upon fundamentally different principles and methods.

Many of them are based on the *causal rejection principle* [23,3,12,2] which states that a rule is *rejected* only if it is directly contradicted by a more recent rule. This essentially means that inertia and minimal change, applied at the level of *literals* in

model-based belief update operators, is instead applied to *rules* and the truth values of literals follow from the set of unrejected rules. Causal rejection semantics are useful in a number of practical scenarios and their behaviour is intuitively predictable. Alternative approaches to rule updates employ syntactic transformations and other methods, such as abduction [31], prioritisation and preferences [40,11], or dependencies on default assumptions [32,22]. The main feature of all these approaches is that they need to refer to the syntactic structure of a logic program: the individual rules and, in most cases, also the literals in heads and bodies of these rules. These properties render them seemingly irreconcilable with belief updates since ontology axioms and formulae in Classical Logic simply have no heads and bodies.

Towards Updates of Hybrid Knowledge Bases. The question that arises, then, is: How can we combine methods used for updating ABoxes, TBoxes and rules in a single framework that allows us to *update hybrid knowledge bases*?

In [34,37] we provided partial solutions to this problem but the inherent differences between the distinct approaches to updates have prevented us from suggesting a universal hybrid update semantics. Subsequently, in [33,35,36] we looked for a suitable semantic foundation of rule updates which would be independent of rule syntax and, at the same time, would retain the fundamental properties of existing rule update semantics. This led to the study of *exception-driven rule update operators* and a definition of particular operators that semantically capture the justified update semantics for rule updates [23] and enjoy a number of syntactic as well as semantic properties.

In this paper we go one step beyond and show that exception-driven operators also capture a range of belief update operators, both model- and formula-based. In other words, they form a common basis for both ontology and rule updates and create room for their cross-fertilisation, ripening and further development.

Our main contributions in this paper are as follows:

- We define abstract exception-driven operators for any knowledge representation formalism with a model-theoretic semantics.
- We show that they capture a wide range of belief update operators.
- We discuss the relationship between belief set and belief base revision operators, on the one hand, and exception-driven operators, on the other.

This work has the following structure: In Sect. 2 we introduce update operators for first-order knowledge bases which form the basis for ontology updates. Sections 3 and 4 introduce abstract exception-driven operators and show how they are able to characterise belief updates. We discuss the relationship between our update framework and revision operators and point at interesting future research directions in Sect. 5.

2 Preliminaries

In this section we introduce model- and formula-based update operators for first-order knowledge bases which underlie the formal approaches to ontology updates [25,10,7,24].

One of the main issues with ontology updates is the *expressibility* of the result of an update which arises due to the fact that Description Logics are *fragments* of first-order logic, so the result of an update operator may not be expressible in the DL used to

encode the original ontology and its update [5]. Nevertheless, in this paper we abstract away from this problem, noting that Description Logics for which expressibility is guaranteed have been identified [25,10], approximation techniques for the updated ontology also constitute a viable solution to this problem [10], and the recent work on belief revision within Horn and other fragments of classical logic may show this problem from a new viewpoint (see e.g. [8] and references therein).

Throughout the remainder of this paper we thus assume to be using a function-free first-order language consisting of disjoint non-empty sets of constant and predicate symbols $\boldsymbol{C}$ and $\boldsymbol{P}$. First-order formulae are defined in the standard way and by a *(first-order) knowledge base* we mean a set of first-order sentences.

From a semantic viewpoint we adopt first-order interpretations under the *standard names assumption* in order to simplify comparison between first-order interpretations, problematic when the interpretation of constants may vary [39,10]. More formally, we assume that the set of constant symbols $\boldsymbol{C}$ is infinite and all first-order interpretations are over the universe $\boldsymbol{C}$ where every constant is interpreted by itself. In addition, we assume that the equality predicate $\approx$ is allowed to be interpreted by any congruence relation on $\boldsymbol{C}$ that allows for replacement of equals by equals, enabling us to support updates of equality assertions. The set of all interpretations satisfying these conditions is denoted by $\boldsymbol{I}$. Note that due to Theorems 5.9.4 and 9.3.9 in [13], the semantics we adopt preserves the standard first-order consequences of all finite knowledge bases.

Furthermore, every interpretation $I \in \boldsymbol{I}$ directly corresponds to the set of ground atoms that it entails; in the following we use these two notions interchangeably. We denote the set of models of a knowledge base $\mathcal{B}$ by $[\![\mathcal{B}]\!]$ and say that $\mathcal{B}$ is *consistent* if $\mathcal{B}$ has a model. Given two knowledge bases $\mathcal{B}$, $\mathcal{C}$, we say that $\mathcal{B}$ *entails* $\mathcal{C}$, denoted by $\mathcal{B} \models \mathcal{C}$, if $[\![\mathcal{B}]\!] \subseteq [\![\mathcal{C}]\!]$, and that $\mathcal{B}$ *is equivalent to* $\mathcal{C}$, denoted by $\mathcal{B} \equiv \mathcal{C}$, if $[\![\mathcal{B}]\!] = [\![\mathcal{C}]\!]$.

We liberally define an update operator as any function that takes the original knowledge base and its update as inputs, and returns the updated knowledge base.

Definition 1 (Update Operator). *A* (first-order) update operator *is a binary function over the set of all knowledge bases. Any update operator $\diamond$ is inductively generalised to finite sequences of knowledge bases $\langle \mathcal{B}_i \rangle_{i<n}$ as follows:*

$$\diamondsuit\langle \mathcal{B}_0 \rangle = \mathcal{B}_0 \ , \qquad \diamondsuit\langle \mathcal{B}_i \rangle_{i<n+1} = (\diamondsuit\langle \mathcal{B}_i \rangle_{i<n}) \diamond \mathcal{B}_n \ .$$

In the following we consider two complementary ways of further specifying an update operator. While the first one puts constraints on the models of knowledge bases produced by it, the second directly defines the resulting knowledge base by performing modifications at the syntactic level.

Model-Based Update Operators. The basic idea underlying model-based update operators is that models of the original knowledge base are viewed as *alternative states* of the modelled world, only one of which is the true one. Given this perspective, it is natural to perform an update with $\mathcal{U}$ by updating each of the alternatives independently of the others, making it consistent with $\mathcal{U}$, and thus obtaining a new set of interpretations – the models of the updated knowledge base. Formally this is captured by the equation

$$[\![\mathcal{B} \diamond \mathcal{U}]\!] = \bigcup_{I \in [\![\mathcal{B}]\!]} \mathsf{incorporate}([\![\mathcal{U}]\!], I) \ , \tag{1}$$

where $\mathsf{incorporate}(\mathcal{M}, I)$ returns the members of $\mathcal{M}$ closer to I so that the original information in I is preserved as much as possible. A natural way of defining this set is by assigning a preorder $\leq^I$ over $\boldsymbol{I}$ to each interpretation I and taking the minima of $\leq^I$ within $\mathcal{M}$, i.e. $\mathsf{incorporate}(\mathcal{M}, I) = \min(\mathcal{M}, \leq^I)$. In the following we first formally establish the concept of an *order assignment*; thereafter we define when an update operator is *characterised by an order assignment*.

Given a set $\mathcal{S}$, a *preorder over* $\mathcal{S}$ is a reflexive and transitive binary relation over $\mathcal{S}$; a *strict preorder over* $\mathcal{S}$ is an irreflexive and transitive binary relation over $\mathcal{S}$. Given a preorder $\leq$ over $\mathcal{S}$, we denote by $<$ the strict preorder induced by $\leq$, i.e. $s < t$ if and only if $s \leq t$ and not $t \leq s$. For any subset $\mathcal{T}$ of $\mathcal{S}$, the set of minimal elements of $\mathcal{T}$ w.r.t. $\leq$ is denoted by $\min(\mathcal{T}, \leq)$. A *preorder assignment over* $\mathcal{S}$ is any function ω that assigns a preorder $\leq^s_\omega$ over $\mathcal{S}$ to each $s \in \mathcal{S}$. A preorder assignment ω is *faithful* if for all $s, t \in \mathcal{S}$ with $s \neq t$, $s <^s_\omega t$.

Definition 2 (Model-Based Update Operator [21]). *Let $\diamond$ be a first-order update operator and ω a preorder assignment over $\boldsymbol{I}$. We say that $\diamond$ is* characterised by ω *if for all knowledge bases $\mathcal{B}, \mathcal{U}$,*

$$[\![\mathcal{B} \diamond \mathcal{U}]\!] = \bigcup_{I \in [\![\mathcal{B}]\!]} \min\left([\![\mathcal{U}]\!], \leq^I_\omega\right) \quad .$$

An operator $\diamond$ is model-based *if it is characterised by some faithful preorder assignment.*

The model-based operator that underlies the work on ABox updates [25,10] is Winslett's operator which compares interpretations based on the sets of ground atoms that they interpret differently than the original interpretation.

Definition 3 (Winslett's Operator [39]). *The preorder assignment W is defined for all interpretations $I, J, K \in \boldsymbol{I}$ as $J \leq^I_{\mathsf{W}} K$ if and only if $(J \div I) \subseteq (K \div I)$, where $\div$ denotes the set-theoretic symmetric difference.* Winslett's operator $\diamond_{\mathsf{W}}$ *is a fixed update operator that is characterised by W.*

Formula-Based Operators. The traditional formula-based update operators that manipulate a knowledge base syntactically are Set-Of-Theories, Cross-Product and WIDTIO (see [39] and references therein). The central notion for these operators is that of a *possible remainder* which is a maximal set of formulae from the original knowledge base that is consistent with the update. Formally, given knowledge bases $\mathcal{B}$ and $\mathcal{U}$, the set of possible remainders $\mathsf{rem}(\mathcal{B}, \mathcal{U})$ is the set of maximal subsets $\mathcal{B}'$ of $\mathcal{B}$ such that $\mathcal{B}' \cup \mathcal{U}$ is consistent. The distinct formula-based operators differ in how they deal with multiple possible remainders. The Set-Of-Theories operator returns the set of all alternative results, i.e. the set of knowledge bases $\mathcal{B}' \cup \mathcal{U}$ for every $\mathcal{B}' \in \mathsf{rem}(\mathcal{B}, \mathcal{U})$. Assuming that the initial knowledge base is finite, the Cross-Product operator compiles these different remainders into a single formula and returns a single knowledge base that is equivalent to the "disjunction" of knowledge bases returned by the Set-Of-Theories operator.

Definition 4 (Cross-Product Operator). *The formula-based operator $\diamond_{\mathsf{CP}}$ is defined for all finite knowledge bases $\mathcal{B}, \mathcal{U}$ as $\mathcal{B} \diamond_{\mathsf{CP}} \mathcal{U} = \mathcal{U} \cup \{ \psi \}$ where ψ is the formula*

$$\bigvee_{\mathcal{B}' \in \mathsf{rem}(\mathcal{B}, \mathcal{U})} \bigwedge_{\phi \in \mathcal{B}'} \phi \quad .$$

On the other hand, the operator WIDTIO (When In Doubt, Throw It Out [39]) takes the safe path – it keeps exactly those formulae that belong to the intersection of all remainders and throws away the rest.

Definition 5 (WIDTIO Operator). *The formula-based operator* $\diamond_{\mathsf{WIDTIO}}$ *is defined for all knowledge bases* $\mathcal{B}, \mathcal{U}$ *as* $\mathcal{B} \diamond_{\mathsf{WIDTIO}} \mathcal{U} = \mathcal{U} \cup \bigcap \mathsf{rem}(\mathcal{B}, \mathcal{U})$.

Recently, an operator inspired by WIDTIO was defined in [24] to tackle ABox updates. Additionally, in [7] the new formula-based operator *Bold* was suggested for performing TBox updates because of the counterintuitive behaviour of model-based operators when used for this purpose. The Bold operator solves the problem of multiple remainders by using a selection function to choose one and commit to it.

Definition 6 (Bold Operator [7]). *A* remainder selection function *is a function* s *that assigns to every set of remainders* $\mathcal{R}$ *a remainder* $\mathsf{s}(\mathcal{R}) \in \mathcal{R}$.

Given a remainder selection function s*, the formula-based operator* $\diamond^{\mathsf{s}}_{\mathsf{BOLD}}$ *is for all knowledge bases* $\mathcal{B}, \mathcal{U}$ *defined as* $\mathcal{B} \diamond^{\mathsf{s}}_{\mathsf{BOLD}} \mathcal{U} = \mathcal{U} \cup \mathsf{s}(\mathsf{rem}(\mathcal{B}, \mathcal{U}))$.

3 Exception-Driven Operators

In order to show how belief- and formula-based operators can be characterised in a unified manner, we define an abstract framework for *exception-driven operators*, usable for any knowledge representation formalism with a monotonic model-theoretic semantics. We also demonstrate how the *justified update semantics* (or *JU-semantics*) for rule updates [23] was characterised semantically in [36] using exception-driven operators.

Abstract Exception-Driven Operators. Throughout this subsection we assume to be using some knowledge representation formalism in which a *knowledge base* is a subset of the set of all *knowledge atoms* Ω and $\mathcal{Z}$ denotes the set of all *semantic structures* among which the *models* of knowledge atoms are chosen. The set of models of a knowledge atom α is denoted by $[\![\alpha]\!]$. The *semantic characterisation* of a knowledge base $\mathcal{K}$ is the *set of sets of models* of its knowledge atoms: $\langle\!\langle \mathcal{K} \rangle\!\rangle = \{\, [\![\alpha]\!] \mid \alpha \in \mathcal{K} \,\}$. The models of $\mathcal{K}$ are the models of all its elements, i.e. $[\![\mathcal{K}]\!] = \bigcap \langle\!\langle \mathcal{K} \rangle\!\rangle$.

An exception-driven operator views a knowledge $\mathcal{K}$ through its semantic characterisation $\langle\!\langle \mathcal{K} \rangle\!\rangle$ and *introduces exceptions* to its knowledge atoms by adding new semantic structures to their original sets of models. The formalisation of this idea is straightforward: an exception-driven update operator is characterised by an *exception function* that, given the set of models of a knowledge atom α and the semantic characterisations of the original and updating knowledge base, returns the set of semantic structures that are to be introduced as exceptions to α.

Definition 7 (Exception Function). *An* exception function *is any function*

$$\varepsilon : 2^{\mathcal{Z}} \times 2^{2^{\mathcal{Z}}} \times 2^{2^{\mathcal{Z}}} \to 2^{\mathcal{Z}} \quad .$$

Given such an exception function and knowledge bases $\mathcal{K}, \mathcal{U}$, it naturally follows that the semantic characterisation resulting from updating $\mathcal{K}$ by $\mathcal{U}$ should consist of

sets of models of each knowledge atom α from $\mathcal{K}$, each augmented with the respective exceptions, and also the unmodified sets of models of knowledge atoms from $\mathcal{U}$. In other words, we obtain the set of sets of models

$$\{ [\![\alpha]\!] \cup \varepsilon([\![\alpha]\!], \langle\!\langle \mathcal{K} \rangle\!\rangle, \langle\!\langle \mathcal{U} \rangle\!\rangle) \mid \alpha \in \mathcal{K} \} \cup \langle\!\langle \mathcal{U} \rangle\!\rangle \ . \tag{2}$$

Turning to the syntactic side, an *update operator* is binary function over 2^{Ω} that takes the original knowledge base and its update as inputs and returns the updated knowledge base. An *exception-driven update operator* is then formalised as follows:

Definition 8 (Exception-Driven Update Operator). *We say that an update operator $\oplus$ is* exception-driven *if for some exception function ε, $\langle\!\langle \mathcal{K} \oplus \mathcal{U} \rangle\!\rangle$ is equal to* (2) *for all $\mathcal{K}, \mathcal{U} \subseteq \Omega$. In that case we also say that $\oplus$ is ε-*driven*.*

Before we begin formally comparing model- and formula-based operators with exception-driven ones, we briefly illustrate how the results of [36], where the JU-semantics for rule updates was semantically characterised, fit within our abstract framework for exception-driven operators. Our main intention in doing so is to provide the reader with a broader picture of exception-driven operators; the technical details left out in what follows can be found in [36].

Exception-Driven Rule Updates. We adopt the standard syntax and the stable models semantics of propositional logic programs [16]. In particular, given a set of atoms $\boldsymbol{A}$, a *literal* is an atom $p \in \boldsymbol{A}$ or its default negation $\sim p$, a *rule* consists of a pair of sets of literals $(H(\pi), B(\pi))$, usually written $(H(\pi) \leftarrow B(\pi).)$, and a *program* is a set of rules. An *interpretation* is a subset of $\boldsymbol{A}$ that naturally assigns truth values to atoms and a *model* of a rule is an interpretation that satisfies the rule when interpreted as a classical implication. Models of a program $\mathcal{P}$ are the models of all its rules and an interpretation J is a *stable model* of $\mathcal{P}$ if it is a subset-minimal model of its Gelfond-Lifschitz reduct $\mathcal{P}^J$ [16]. The set of stable models of $\mathcal{P}$ is denoted by $[\![\mathcal{P}]\!]_{\mathsf{SM}}$.

The goal of rule update semantics [26,23,3,12,2,31,40,11,22,32] is to generalise the definition of stable models to *pairs* or *sequences of programs* where each component represents an update of the preceding ones. These semantics are usually constrained to *finite sequences of non-disjunctive programs* which we call *dynamic logic programs* (DLPs). Typically, they are defined by referring to the syntactic structure of the programs in a DLP. As a consequence, analysis of their semantic properties is very daunting and most of them do exhibit undesirable behaviour, e.g. by being sensitive to *tautological updates* which is counterintuitive in the context of updates – a tautology cannot encode a *change* in the modelled world because it is always true. The historically first semantics for DLPs is the JU-semantics [23]. We denote the set of all JU-models of a DLP $\mathcal{D}$ by $[\![\mathcal{D}]\!]_{\mathsf{JU}}$.

The operators introduced in [36] can be seen as an instantiation of the abstract framework introduced above. In this context, the set of knowledge atoms Ω consists of all *rules and programs* and the set of semantic structures $\mathcal{Z}$ of *three-valued interpretations*. A *knowledge base* (or *rule base*) is thus any set of rules and programs and its elements are perceived as atomic pieces of knowledge. Note that a program is a special case of a rule base. The reason why we allow for programs *inside* a rule base is that

when a rule is updated, by adding exceptions to its set of models, the resulting set of models is usually not expressible by a rule, only by a program. Note also that the notion of a *stable model* can be naturally generalised to rule bases by introducing models of a rule base as the models of all its elements and defining the Gelfond-Lifschitz reduct of a rule base $\mathcal{R}$ as $\mathcal{R}^J = \left\{ \Pi^J \;\middle|\; \Pi \in \mathcal{R} \right\}$.

In [36], the semantics assigned to each rule or program in a rule base is given by a refinement of *SE-models* [38], dubbed *RE-models*, which can distinguish additional classes of rules, indispensable in the context of updates. An exception function, here denoted by $\varepsilon_{\mathsf{JU}}$, is then defined. Details about RE-models and $\varepsilon_{\mathsf{JU}}$ can be found in [36].

The main property of $\varepsilon_{\mathsf{JU}}$ is that stable models of the rule base produced by an $\varepsilon_{\mathsf{JU}}$-driven operator, when applied to a DLP $\mathcal{D}$, coincide with its JU-models. This holds whenever $\mathcal{D}$ does not contain *local cycles*, i.e. rules π with both $\{ p, {\sim}p \} \cap H(\pi) \neq \emptyset$ and $\{ p, {\sim}p \} \cap B(\pi) \neq \emptyset$ for some $p \in \boldsymbol{A}$.

Theorem 9 ([36]). *Let $\mathcal{D}$ be a DLP without local cycles, J an interpretation and $\oplus$ an $\varepsilon_{\mathsf{JU}}$-driven rule update operator. Then $[\![\bigoplus \mathcal{D}]\!]_{\mathsf{SM}} = [\![\mathcal{D}]\!]_{\mathsf{JU}}$.*

This means that up to the marginal case of local cycles, $\varepsilon_{\mathsf{JU}}$ can be seen as a semantic characterisation of the JU-semantics: it leads to stable models that coincide with JU-models. In case the DLP contains local cycles, *less* stable models than JU-models are found [36]. Local cycles correspond to two different kinds of rules: tautological rules and rules with the negation of their head in the body. The different behaviour in the presence of tautological rules is a strict improvement over JU-models, as it introduces immunity to tautological updates. The other differences are a consequence of treating constraints such as $(p \leftarrow {\sim}p.)$ and $(\leftarrow {\sim}p.)$ uniformly while the JU-semantics treats them differently under certain circumstances.

This tight relationship allowed us to study the semantic properties of JU-models under a range of different notions of program equivalence and entailment, and to shed new light on the problem of *state condensing* since $\varepsilon_{\mathsf{JU}}$-driven operators compress any DLP into a single equivalent rule base.

Even more importantly, these results, along with the developments in this paper, show that exception-driven operators form a common semantic basis for both ontology and rule updates, and so create room for addressing updates of hybrid knowledge bases.

4 Belief Updates Using Exception-Driven Operators

Concrete exception-driven operators for first-order knowledge bases are obtained from the abstract framework developed in Sect. 3 by identifying the set of knowledge atoms Ω with the set of first-order sentences and the set of semantic structures $\mathcal{Z}$ with first-order interpretations under the standard names assumption, as introduced in Sect. 2.

In [21] it was shown that *propositional* model-based update operators are exactly those that satisfy a collection of eight *update postulates*. These postulates express basic desirable properties of update operators and most of them can be directly generalised to the first-order case. Here we use the following three basic properties of first-order update operators and prove results about the class of all operators satisfying them. The

properties are formulated for an update operator $\diamond$ and quantified over all knowledge bases $\mathcal{B}, \mathcal{C}, \mathcal{U}, \mathcal{V}$.[3]

(U1) $\mathcal{B} \diamond \mathcal{U} \models \mathcal{U}$.

(U2.1) $\mathcal{B} \cup \mathcal{U} \models \mathcal{B} \diamond \mathcal{U}$.

(U4) If $\mathcal{B} \equiv \mathcal{C}$ and $\mathcal{U} \equiv \mathcal{V}$, then $\mathcal{B} \diamond \mathcal{U} \equiv \mathcal{C} \diamond \mathcal{V}$.

The intuitive reading of (U1) is that information from the update must be retained in the updated knowledge base, also known as the *principle of primacy of new information* [9]; (U2.1) expresses that models of $\mathcal{B}$ that are also models of $\mathcal{U}$, and thus need not be updated, are kept as models of the updated knowledge base; (U4) specifies that the operator must be *syntax-independent*, i.e. it must provide equivalent results given equivalent inputs. All model-based update operators, including Winslett's, satisfy these principles:

Proposition 10 (Properties of Model-Based Updates). *Every model-based update operator satisfies (U1), (U2.1) and (U4).*

Furthermore, any operator satisfying these three principles can be faithfully modelled by an exception-driven operator. Formally:

Theorem 11 (Model-Based Updates Using Exception-Driven Operators). *If $\diamond$ is an update operator that satisfies (U1), (U2.1) and (U4), then there exists an exception function ε such that for every ε-driven update operator $\oplus$ and all finite sequences of knowledge bases $\mathcal{D}$, $[\![\diamond \mathcal{D}]\!] = [\![\bigoplus \mathcal{D}]\!]$.*

Similar results can be achieved for formula-based update operators. First we introduce the following principles, counterparts of the respective belief update postulates, which are satisfied by many formula-based operators. We denote by $\langle\!\langle \mathcal{B} \rangle\!\rangle^{\boldsymbol{I}}$ the set $\langle\!\langle \mathcal{B} \rangle\!\rangle \cup \{ \boldsymbol{I} \}$ for any knowledge base $\mathcal{B}$. The principles are as follows:

(F1) $\langle\!\langle \mathcal{B} \diamond \mathcal{U} \rangle\!\rangle \supseteq \langle\!\langle \mathcal{U} \rangle\!\rangle$.

(F2.1) $\langle\!\langle \mathcal{B} \cup \mathcal{U} \rangle\!\rangle \supseteq \langle\!\langle \mathcal{B} \diamond \mathcal{U} \rangle\!\rangle$.

(F4) If $\langle\!\langle \mathcal{B} \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathcal{C} \rangle\!\rangle^{\boldsymbol{I}}$ and $\langle\!\langle \mathcal{U} \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathcal{V} \rangle\!\rangle^{\boldsymbol{I}}$, then $\langle\!\langle \mathcal{B} \diamond \mathcal{U} \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathcal{C} \diamond \mathcal{V} \rangle\!\rangle^{\boldsymbol{I}}$.

We can see that (F1) and (F2.1) are *stronger* versions of (U1), and (U2.1), respectively. While (F1) requires that the sets of models of formulae in $\mathcal{U}$ be retained in the semantic characterisation of $\mathcal{B} \diamond \mathcal{U}$, (F2.1) states that every formula in $\mathcal{B} \diamond \mathcal{U}$ be equivalent to some formula in $\mathcal{B} \cup \mathcal{U}$. Intuitively, this means that $\mathcal{B} \diamond \mathcal{U}$ is obtained from $\mathcal{B} \cup \mathcal{U}$ by deleting some of its elements, modulo equivalence. Finally, (F4) is a reformulation of (U4) that is satisfied by formula-based operators – it can be seen as syntax-independence w.r.t. the set of sets of models of a knowledge base, modulo the presence of tautologies, instead of the overall set of models as in (U4). In some ways it is *weaker* than (U4) as its antecedent is much stronger.

The WIDTIO operator satisfies all of these principles, and so does the Bold operator if it is based on a remainder selection function that selects remainders with the same semantic characterisation when given sets of remainders with the same sets of semantic characterisations. More formally:

[3] Their numbers are as in [21,19].

Definition 12 (Regular Bold Operator). *Let $\mathcal{R}$ be a set of remainders. We denote the set $\{ \langle\!\langle \mathcal{B}' \rangle\!\rangle^{\boldsymbol{I}} \mid \mathcal{B}' \in \mathcal{R} \}$ by $(\!(\mathcal{R})\!)^{\boldsymbol{I}}$.*

We say that the Bold operator $\diamond^{\mathsf{s}}_{\mathsf{BOLD}}$ is regular *if for all sets of remainders $\mathcal{R}_1$, $\mathcal{R}_2$ such that $(\!(\mathcal{R}_1)\!)^{\boldsymbol{I}} = (\!(\mathcal{R}_2)\!)^{\boldsymbol{I}}$ it holds that $\langle\!\langle \mathsf{s}(\mathcal{R}_1) \rangle\!\rangle^{\boldsymbol{I}} = \langle\!\langle \mathsf{s}(\mathcal{R}_2) \rangle\!\rangle^{\boldsymbol{I}}$.*

The regularity condition guarantees a certain degree of independence of syntax, e.g. given the sets of remainders $\mathcal{R}_1 = \{ \{ p \}, \{ q \} \}$ and $\mathcal{R}_2 = \{ \{ p \wedge p \}, \{ q \vee q \} \}$, a regular Bold operator either selects $\{ p \}$ from $\mathcal{R}_1$ and $\{ p \wedge p \}$ from $\mathcal{R}_2$, or it selects $\{ q \}$ from $\mathcal{R}_1$ and $\{ q \vee q \}$ from $\mathcal{R}_2$. A non-regular one might select, say, $\{ p \}$ from $\mathcal{R}_1$ and $\{ q \vee q \}$ from $\mathcal{R}_2$. Thus the regularity condition ensures that the operator is independent of the syntax of individual formulae in the knowledge base.

The Cross-Product operator satisfies (F1), (U2.1) and (F4), but not (F2.1).

Proposition 13 (Properties of Formula-Based Updates). *The WIDTIO and regular Bold operators satisfy (F1), (F2.1) and (F4). The Cross-Product operator satisfies (F1), (U2.1) and (F4) but does not satisfy (F2.1).*

The following result establishes that formula-based operators such as WIDTIO and regular Bold can be fully captured by exception-driven operators. In addition, operators such as Cross-Product can be captured for the case of a single update.

Theorem 14 (Formula-Based Updates Using Exception-Driven Operators). *If $\diamond$ is an update operator that satisfies (F1), (F2.1) and (F4), then there exists an exception function ε such that for every ε-driven update operator $\oplus$ and all finite sequences of knowledge bases $\mathcal{D}$, $[\![\diamond \mathcal{D}]\!] = [\![\bigoplus \mathcal{D}]\!]$.*

If $\diamond$ is an update operator that satisfies (F1), (U2.1) and (F4), then there exists an exception function ε such that for every ε-driven update operator $\oplus$ and all knowledge bases $\mathcal{B}, \mathcal{U}$, $[\![\mathcal{B} \diamond \mathcal{U}]\!] = [\![\mathcal{B} \oplus \mathcal{U}]\!]$.

5 Discussion

We have introduced exception-driven operators for first-order knowledge bases and shown that they can fully capture update operators that form the basis of ontology updates, such as the model-based Winslett's operator, or the formula-based WIDTIO and Bold operators [25,10,7,24]. The Cross-Product operator can be captured when a single update is performed. Furthermore, the same can be said about the Set-Of-Theories operator since for a single update it is equivalent to the Cross-Product operator [39], with alternative knowledge bases interpreted disjunctively. However, neither of these two operators offers a viable alternative for updating ontologies. Cross-Product requires that disjunctions of ontology axioms be performed, which is typically not supported in DLs, and Set-Of-Theories produces a disjunctive ontology which is impractical and deviates from mainstream DL research.

An interesting point regarding the results of Sect. 4 is that the principles (U1), (U2.1) and (U4) are not specific to update operators, they are also satisfied by *AGM revision* operators. These operators are developed for the case of revising a *belief set* which is a set of formulae closed w.r.t. a logical consequence operator Cn. A revision operator $\star$

takes an original belief set $\mathcal{T}$ and a formula μ representing its revision and produces the revised belief set $\mathcal{T} \star \mu$. The typical properties satisfied by AGM revision operators include *success*, *inclusion* and *extensionality* [18], formalised, respectively, as

$$\mu \in \mathcal{T} \star \mu \ , \qquad \mathcal{T} \star \mu \subseteq \mathsf{Cn}(\mathcal{T} \cup \{ \mu \}) \ , \qquad \text{If } \mu \equiv \nu, \text{ then } \mathcal{T} \star \mu = \mathcal{T} \star \nu.$$

These three properties directly imply that (U1), (U2.1) and (U4) are satisfied by AGM revision operators if the initial knowledge base is a belief set and each of its updates a single formula. This essentially means that Theorem 11 directly applies to AGM revision operators as well. Note that the operator adopted for ABox updates in [24], inspired by WIDTIO, performs a deductive closure of the ABox before updating it, so it corresponds to the standard *full meet AGM revision operator*.

Similarly, principles (F1), (F2.1) and (F4) are closely related with the properties of *base revision operators* [15,18], of which direct instances are the WIDTIO and Bold operators. In particular, two types of base revision are identified in [18], the *internal* and *external base revision*. Both of them satisfy base revision counterparts of *success* and *inclusion* and, in addition, internal revision operators satisfy a property called *uniformity*. These three principles together entail that internal revision operators satisfy (F1), (F2.1) and one half of (F4); the other half can be achieved by putting additional constraints on the two-place selection function that generates the revision operator, similar to the *regularity* condition we imposed on the Bold operator above. Such regular internal revision operators are thus directly subject to Theorem 14. The same however does not hold for regular external revision operators as they need not satisfy *uniformity*. Note also that the WIDTIO and Bold operators coincide with *internal full meet base revision* and *internal maxichoice base revision* operators, respectively.

To sum up, in this paper we introduced the abstract framework for *exception-driven operators* which view a knowledge base or program as the *set of sets of models* of its elements, and perform updates by adding new interpretations – *exceptions* – to the sets of models of elements in the original knowledge base or program. The most important feature of this approach is that it provides a common basis for a wide range of model- and formula-based belief update operators as well as for the JU-semantics, a traditional syntax-based approach to rule updates. In other words, exception functions and exception-driven operators offer a uniform framework that bridges two very distinct approaches to updates, previously considered irreconcilable.

Along with this, new possibilities for addressing updates of hybrid knowledge bases arise. The different methods used for dealing with ABox, TBox and rule updates can be viewed uniformly by looking at their associated exception functions. When coupled with a counterpart of SE- or RE-models in the context of hybrid knowledge bases, this can lead to universal hybrid update semantics which in turn can further improve our understanding of the relation between the distinct update paradigms.

Our discussion of the expressivity of exception-driven operators w.r.t. *revision* operators, on both belief sets and belief bases, can be used to tackle and unify approaches to *ontology revision* [29,17,30]. This seems relevant even in the context of ontology *updates* since it has been suggested in the literature that the strict distinction between revision and update is not suitable in the context of ontologies [7].

Furthermore, exception-driven characterisations of additional rule update semantics need to be investigated. This poses a number of challenges due to the need to detect non-

tautological irrelevant updates [2,32]. Insights gained by obtaining exception-driven characterisations of various rule update semantics may also shed light on the problem of *updating disjunctive programs* which has received very little attention up until now.

Acknowledgements. We would like to thank the reviewers for their valuable comments. The authors were partially supported by the FCT funded project ERRO – Efficient Reasoning with Rules and Ontologies (PTDC/EIA-CCO/121823/2010).

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. Symb. Log. 50(2), 510–530 (1985)
2. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Studia Logica 79(1), 7–32 (2005)
3. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic updates of non-monotonic knowledge bases. J. Log. Program. 45(1-3), 43–70 (2000)
4. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge Univ Press, 2nd edn. (2007)
5. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Veloso, M.M., Kambhampati, S. (eds.) Procs. AAAI 2005. pp. 572–577. AAAI/MIT Press (2005)
6. de Bruijn, J., Eiter, T., Polleres, A., Tompits, H.: Embedding nonground logic programs into autoepistemic logic for knowledge-base combination. ACM Trans. Comput. Log. 12(3), 20 (2011)
7. Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of DL-Lite knowledge bases. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) Procs. ISWC 2010. LNCS, vol. 6496, pp. 112–128. Springer (2010)
8. Creignou, N., Papini, O., Pichler, R., Woltran, S.: Belief revision within fragments of propositional logic. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Procs. KR 2012. pp. 126–136. AAAI Press (2012)
9. Dalal, M.: Investigations into a theory of knowledge base revision. In: Procs. AAAI 1988. pp. 475–479. AAAI/MIT Press (1988)
10. De Giacomo, G., Lenzerini, M., Poggi, A., Rosati, R.: On instance-level update and erasure in description logic ontologies. J. Log. Comput. 19(5), 745–770 (2009)
11. Delgrande, J.P., Schaub, T., Tompits, H.: A preference-based framework for updating logic programs. In: Baral, C., Brewka, G., Schlipf, J.S. (eds.) Procs. LPNMR 2007. LNCS, vol. 4483, pp. 71–83. Springer (2007)
12. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: On properties of update sequences based on causal rejection. TPLP 2(6), 721–777 (2002)
13. Fitting, M.: First-Order Logic and Automated Theorem Proving. Graduate texts in computer science, Springer-Verlag, Berlin, Germany, 2nd edn. (1996)
14. Flouris, G., Makanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology change: classification and survey. Knowledge Eng. Review 23(2), 117–152 (2008)
15. Gärdenfors, P.: Belief Revision, chap. Belief Revision: An Introduction, pp. 1–28. Cambridge Univ Press (1992)
16. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Procs. ICLP/SLP 1988. pp. 1070–1080. MIT Press (1988)

17. Halaschek-Wiener, C., Katz, Y.: Belief base revision for expressive description logics. In: Grau, B.C., Hitzler, P., Shankey, C., Wallace, E. (eds.) Procs. OWLED 2006. CEUR Workshop Proceedings, vol. 216. CEUR-WS.org (2006)
18. Hansson, S.O.: Reversing the Levi identity. J. Philosophical Logic 22(6), 637–669 (1993)
19. Herzig, A., Rifi, O.: Propositional belief base update and minimal change. Artif. Intell. 115(1), 107–138 (1999)
20. Hitzler, P., Parsia, B.: Ontologies and rules. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 111–132. International Handbooks on Information Systems, Springer, Berlin, 2nd edn. (2009)
21. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Allen, J.F., Fikes, R., Sandewall, E. (eds.) Procs. KR 1991. pp. 387–394. Morgan Kaufmann Publishers, Cambridge, MA, USA (April 22-25 1991)
22. Krümpelmann, P., Kern-Isberner, G.: On belief dynamics of dependency relations for extended logic programs. In: Procs. NMR 2010. Toronto, Canada (2010)
23. Leite, J.A., Pereira, L.M.: Generalizing updates: From models to programs. In: Dix, J., Pereira, L.M., Przymusinski, T.C. (eds.) Procs. LPKR 1997. LNCS, vol. 1471, pp. 224–246. Springer (1997)
24. Lenzerini, M., Savo, D.F.: On the evolution of the instance level of DL-Lite knowledge bases. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Procs. DL 2011. CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
25. Liu, H., Lutz, C., Miličić, M., Wolter, F.: Updating description logic ABoxes. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Procs. KR 2006. pp. 46–56. AAAI Press (2006)
26. Marek, V.W., Truszczynski, M.: Revision programming. Theor. Comput. Sci. 190(2), 241–277 (1998)
27. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5), 93–154 (2010)
28. Peppas, P., Nayak, A.C., Pagnucco, M., Foo, N.Y., Kwok, R.B.H., Prokopenko, M.: Revision vs. update: Taking a closer look. In: Wahlster, W. (ed.) Procs. ECAI 1996. pp. 95–99. John Wiley and Sons, Chichester (1996)
29. Qi, G., Yang, F.: A survey of revision approaches in description logics. In: Calvanese, D., Lausen, G. (eds.) Procs. RR 2008. LNCS, vol. 5341, pp. 74–88. Springer Verlag (2008)
30. Ribeiro, M.M., Wassermann, R.: Base revision in description logics - preliminary results. In: Procs. IWOD 2007. pp. 69–82 (2007)
31. Sakama, C., Inoue, K.: An abductive framework for computing knowledge base updates. TPLP 3(6), 671–713 (2003)
32. Šefránek, J.: Static and dynamic semantics: Preliminary report. MICAI pp. 36–42 (2011)
33. Slota, M., Leite, J.: On semantic update operators for answer-set programs. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) Procs. ECAI 2010. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 957–962. IOS Press (2010)
34. Slota, M., Leite, J.: Towards Closed World Reasoning in Dynamic Open Worlds. TPLP Special Issue 10(4-6), 547–564 (2010)
35. Slota, M., Leite, J.: Back and forth between rules and SE-models. In: Delgrande, J.P., Faber, W. (eds.) Procs. LPNMR 2011. LNCS, vol. 6645, pp. 174–186. Springer (2011)
36. Slota, M., Leite, J.: Robust equivalence models for semantic updates of answer-set programs. In: Brewka, G., Eiter, T., McIlraith, S.A. (eds.) Procs. KR 2012. pp. 158–168. AAAI Press (2012)
37. Slota, M., Leite, J., Swift, T.: Splitting and updating hybrid knowledge bases. TPLP Special Issue 11(4-5), 801–819 (2011)
38. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. TPLP 3(4-5), 609–622 (2003)
39. Winslett, M.: Updating Logical Databases. Cambridge Univ Press, New York, USA (1990)
40. Zhang, Y.: Logic program-based updates. ACM Trans. Comput. Log. 7(3), 421–472 (2006)