Annotated revision programs

Victor Marek, Inna Pivkina, and Mirosław Truszczyński

Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046
`marek|inna|mirek@cs.engr.uky.edu`

Abstract. Revision programming was introduced as a formalism to describe and enforce updates of belief sets and databases. Revision programming was extended by Fitting who assigned annotations to revision atoms. Annotations provide a way to quantify certainty (likelihood) that a revision atom holds. The main goal of our paper is to reexamine the work of Fitting, argue that his semantics does not always provide results consistent with intuition and to propose an alternative treatment of annotated revision programs. Our approach differs from that proposed by Fitting in two key aspects: we change the notion of a model of a program and we change the notion of a justified revision. We show that under this new approach fundamental properties of justified revisions of standard revision programs extend to the case of annotated revision programs.

1 Introduction

Revision programming is a formalism to specify and enforce constraints on databases, belief sets and, more generally, on arbitrary sets of elements. Revision programming was introduced and studied in [MT95,MT98]. The formalism was shown to be closely related to logic programming with stable model semantics [MT98,PT97]. In [MPT99], a simple correspondence of revision programming with the general logic programming system of Lifschitz and Woo [LW92] was discovered. Roots of another recent formalism of dynamic programming [ALP$^+$98] can also be traced back to revision programming.

Revision rules come in two forms of *in-rules* and *out-rules*:

$$\mathbf{in}(a) \leftarrow \mathbf{in}(a_1), \ldots, \mathbf{in}(a_m), \mathbf{out}(b_1), \ldots, \mathbf{out}(b_n) \tag{1}$$

and

$$\mathbf{out}(a) \leftarrow \mathbf{in}(a_1), \ldots, \mathbf{in}(a_m), \mathbf{out}(b_1), \ldots, \mathbf{out}(b_n). \tag{2}$$

Expressions $\mathbf{in}(a)$ and $\mathbf{out}(a)$ are called *revision atoms*. Informally, the atom $\mathbf{in}(a)$ stands for "a is in the current set" and $\mathbf{out}(a)$ stands for "a is not in the current set." The rules (1) and (2) have the following imperative, or computational, interpretation: whenever elements a_k, $1 \leq k \leq m$, belong to the current set (database, belief set) and none of the elements b_l, $1 \leq l \leq n$, belongs to the current set then, in the case of rule (1), the item a must be added to the set (if it is not there already), and in the case of rule (2), a must be eliminated from the database (if it is there). The rules (1) and (2) have also an obvious declarative interpretation.

To provide a precise semantics to *revision programs*, that is, collections of revision rules, the concept of a *justified revision* was introduced in [MT95,MT98]. Informally, given an initial set B_I and a revision program P, a justified revision of B_I with respect to P (or, simply, a P-justified revision of B_I) is obtained from B_I by adding some elements to B_I and by removing some other elements from B_I so that each change is, in a certain sense, justified by the program.

The formalism of revision programs was extended by Fitting [Fit95] to the case when revision atoms are assigned *annotations*. These annotations can be interpreted as the degree of confidence that a revision atom holds. For instance, an annotated atom $(\mathbf{in}(a){:}0.2)$ can be regarded as the statement that a is in the set with probability 0.2. In his paper, Fitting described the concept of a justified revision of an annotated program and studied properties of that notion.

The main goal of our paper is to reexamine the work of Fitting, argue that his semantics does not always provide results consistent with intuition, and to propose an alternative treatment of annotated revision programs. Our approach differs from that proposed by Fitting in two key aspects: we change the notion of a model of a program and we change the notion of a justified revision. We show that under this new approach all fundamental properties of justified revisions of standard revision programs extend to the case of annotated revision programs.

We also show that annotated revision programming can be given a more uniform treatment if the syntax of revision programs is somewhat modified. The new syntax yields a formalism that is equivalent to the original formalism of annotated revision programs. The advantage of the new syntax is that it allows us to generalize the shifting theorem proved in [MPT99] and used there to establish the equivalence of revision programming with general logic programming of Lifschitz and Woo [LW92].

Finally, in the paper we also address briefly the issue of disjunctive annotated programs and other possible research directions.

2 Preliminaries

Throughout the paper we consider a fixed *universe* U whose elements are referred to as *atoms*. Expressions of the form $\mathbf{in}(a)$ and $\mathbf{out}(a)$, where $a \in U$, are called *revision atoms*. In the paper we assign annotations to revision atoms. These annotations are members of a complete distributive lattice with the de Morgan complement (an order reversing involution). Throughout the paper this lattice is denoted by $\mathcal{T}$. The partial ordering on $\mathcal{T}$ is denoted by $\leq$ and the corresponding meet and join operations by $\wedge$ and $\vee$, respectively. The de Morgan complement of $a \in \mathcal{T}$ is denoted by $\bar{a}$.

An *annotated revision atom* is an expression of the form $(\mathbf{in}(a){:}\alpha)$ or $(\mathbf{out}(a){:}\alpha)$, where $a \in U$ and $\alpha \in \mathcal{T}$. An *annotated revision rule* is an expression of the form

$$p \leftarrow q_1, \ldots, q_n,$$

where $p, q_1, \ldots, q_n$ are annotated revision atoms. An *annotated revision program* is a set of annotated revision rules.

A $\mathcal{T}$*-valuation* is a mapping from the set of revision atoms to $\mathcal{T}$. A $\mathcal{T}$-valuation v describes our information about the membership of the elements from U in some (possibly unknown) set $B \subseteq U$. For instance, $v(\mathbf{in}(a)) = \alpha$ can be interpreted as saying that $a \in B$ with certainty α. A $\mathcal{T}$-valuation v *satisfies* an annotated revision atom $(\mathbf{in}(a){:}\alpha)$ if $v(\mathbf{in}(a)) \geq \alpha$. Similarly, v *satisfies* $(\mathbf{out}(a){:}\alpha)$ if $v(\mathbf{out}(a)) \geq \alpha$. The $\mathcal{T}$-valuation v *satisfies* a list or a set of annotated revision atoms if it satisfies each member of the list or the set. A $\mathcal{T}$-valuation *satisfies* an annotated revision rule if it satisfies the head of the rule whenever it satisfies the body of the rule. Finally, a $\mathcal{T}$-valuation *satisfies* an annotated revision program (is a *model* of the program) if it satisfies all rules in the program.

Given a revision program P we can assign to it an operator on the set of all $\mathcal{T}$-valuations. Let $t_P(v)$ be the set of the heads of all rules in P whose bodies are satisfied by v. We define an operator T_P as follows:

$$T_P(v)(l) = \bigvee \{\alpha | (l{:}\alpha) \in t_P(v)\}$$

(note that $\perp$ is the join of an empty set of lattice elements). The operator T_P is a counterpart of the well-know van Emden-Kowalski operator from logic programming and it will play an important role in our paper.

It is clear that under $\mathcal{T}$-valuations, the information about an element $a \in U$ is given by a pair of elements from $\mathcal{T}$ that are assigned to revision atoms $\mathbf{in}(a)$ and $\mathbf{out}(a)$. Thus, in the paper we will also consider an algebraic structure $\mathcal{T}^2$ with the domain $\mathcal{T} \times \mathcal{T}$ and with an ordering $\leq_k$ defined by:

$$\langle \alpha_1, \beta_1 \rangle \leq_k \langle \alpha_2, \beta_2 \rangle \quad \text{if} \quad \alpha_1 \leq \alpha_2 \text{ and } \beta_1 \leq \beta_2.$$

If a pair $\langle \alpha_1, \beta_1 \rangle$ is viewed as a measure of our information about membership of a in some unknown set B then $\alpha_1 \leq \alpha_2$ and $\beta_1 \leq \beta_2$ imply that the pair $\langle \alpha_2, \beta_2 \rangle$ represents higher degree of knowledge about a. Thus, the ordering $\leq_k$ is often referred to as the *knowledge* or *information* ordering. Since the lattice $\mathcal{T}$ is complete, $\mathcal{T}^2$ is a complete lattice with respect to the ordering $\leq_k$[1].

The operations of meet, join, top, and bottom under $\leq_k$ are denoted $\otimes$, $\oplus$, $\top$, and $\perp$, respectively. In addition, we make use of an additional operation, *conflation*. Conflation is defined as $\perp \langle \alpha, \beta \rangle = \langle \bar{\beta}, \bar{\alpha} \rangle$. An element $A \in \mathcal{T}^2$ is *consistent* if $A \leq_k \perp A$.

A $\mathcal{T}^2$-valuation is a mapping from *atoms* to elements of $\mathcal{T}^2$. If $B(a) = \langle \alpha, \beta \rangle$ under some $\mathcal{T}^2$-valuation B, we say that under B the element a is in a set with certainty α and it is not in the set with certainty β. We say that a $\mathcal{T}^2$-valuation is *consistent* if it assigns a consistent element of $\mathcal{T}^2$ to every atom in U.

[1] There is another ordering that can be associated with $\mathcal{T}^2$. We can define $\langle \alpha_1, \beta_1 \rangle \leq_t \langle \alpha_2, \beta_2 \rangle$ if $\alpha_1 \leq \alpha_2$ and $\beta_1 \geq \beta_2$. This ordering is often called the *truth ordering*. Since $\mathcal{T}$ is a distributive lattice, $\mathcal{T}^2$ with both orderings $\leq_k$ and $\leq_t$ forms a bilattice (see [Gin88,Fit99] for a definition). In this paper we will not use the ordering $\leq_t$ nor the fact that $\mathcal{T}^2$ is a bilattice.

In the paper, $\mathcal{T}^2$-valuations will be used to represent current information about sets (databases) as well as change that needs to be enforced. Let B be a $\mathcal{T}^2$-valuation representing our knowledge about a certain set and let C be a $\mathcal{T}^2$-valuation representing change that needs to be applied to B. We define the revision, B', of B by C by

$$B' = (B \otimes \perp C) \oplus C.$$

The intuition is as follows. After the revision, the new valuation must contain at least as much knowledge about atoms being in and out as C. On the other hand, this amount of knowledge must not exceed implicit bounds present in C and expressed by $\perp C$, unless C directly implies so (if $C(a) = \langle \alpha, \beta \rangle$, then evidence for $\mathbf{in}(a)$ must not exceed $\bar{\beta}$ and the evidence for $\mathbf{out}(a)$ must not exceed $\bar{\alpha}$, unless C directly implies so). Since we prefer explicit evidence of C to implicit evidence expressed by $\perp C$, we perform the change by first using $\perp C$ and then applying C (however, let us note here that the order matters only if C is inconsistent; if C is consistent, $(B \otimes \perp C) \oplus C = (B \oplus C) \otimes \perp C$). This specification of how a change modeled by a $\mathcal{T}^2$-valuation is enforced plays a key role in our definition of justified revisions in Section 4.

There is a one-to-one correspondence θ between $\mathcal{T}$-valuations (of revision atoms) and $\mathcal{T}^2$-valuations (of atoms). For a $\mathcal{T}$-valuation v, the $\mathcal{T}^2$-valuation $\theta(v)$ is defined by: $\theta(v)(a) = \langle v(\mathbf{in}(a)), v(\mathbf{out}(a)) \rangle$. The inverse mapping of θ is denoted by θ^{-1}. Clearly, using the mapping θ, the notions of satisfaction defined earlier for $\mathcal{T}$-valuations can be extended to $\mathcal{T}^2$-valuations. Similarly, the operator T_P gives rise to a related operator T_P^b. The operator T_P^b is defined on the set of all $\mathcal{T}^2$-valuations by $T_P^b = \theta \circ T_P \circ \theta^{-1}$. The key property of the operator T_P^b is its $\leq_k$-monotonicity.

Theorem 1. *Let P be an annotated revision program and let B and B' be two $\mathcal{T}^2$-valuations such that $B \leq_k B'$. Then, $T_P^b(B) \leq_k T_P^b(B')$.*

By Tarski-Knaster Theorem it follows that the operator T_P^b has a least fixpoint in $\mathcal{T}^2$ [KS92]. This fixpoint is an analogue of the concept of a least Herbrand model of a Horn program. It represents the set of annotated revision atoms that are implied by the program and, hence, must be satisfied by any revision under P of *any* initial valuation. Given an annotated revision program P we will refer to the least fixpoint of the operator T_P^b as the *necessary change* of P and will denote it by $NC(P)$. The present concept of the necessary change generalizes the corresponding notion introduced in [MT95,MT98] for the original unannotated revision programs.

To illustrate concepts and results of the paper, we will consider two special lattices. The first of them is the lattice with the domain $[0, 1]$ (interval of reals) and with the standard ordering $\leq$ and the standard complement operation. We will denote this lattice by $\mathcal{T}_{[0,1]}$. Intuitively, the annotated revision atom $(\mathbf{in}(a)\colon x)$, where $x \in [0, 1]$, stands for the statement that a is "in" with likelihood (certainty) x.

The second lattice is the Boolean algebra of all subsets of a given set X. It will be denoted by $\mathcal{T}_X$. We will think of elements from X as experts. The annotated revision atom $(\mathbf{out}(a){:}Y)$, where $Y \subseteq X$, will be understood as saying that a is believed to be "out" by those experts that are in Y (the atom $(\mathbf{in}(a){:}Y)$ has a similar meaning).

3 Models and c-models

The semantics of annotated revision programs will be based on the notion of a model as defined in the previous section. The following result provides a characterization of the concept of a model in terms of the operator T_P^b.

Theorem 2. *A $\mathcal{T}^2$-valuation B of an annotated revision program P is a model of P (satisfies P) if and only if $B \geq_k T_P^b(B)$.*

Given an annotated revision program P, its necessary change $NC(P)$ satisfies $NC(P) = T_P^b(NC(P))$. Hence, $NC(P)$ is a model of P.

As we will argue now, not all models are appropriate for describing the meaning of an annotated revision program. The problem is that $\mathcal{T}^2$-valuations may contain inconsistent information about elements from U. When studying the meaning of an annotated revision program we will be interested in those models only whose inconsistencies are limited by the information explicitly or implicitly present in the program.

Consider the annotated revision program P, consisting of the following rule:

$$(\mathbf{in}(a){:}\{q\}) \leftarrow (\mathbf{out}(a){:}\{p\})$$

(the literals are annotated with elements of the lattice $\mathcal{T}_{\{p,q\}}$). Some models of this program are consistent (for instance, the $\mathcal{T}^2$-valuation that assigns $\langle\{q\},\{p\}\rangle$ to a). However, P also has inconsistent models. Let us consider first the $\mathcal{T}^2$-valuation B_1 such that $B_1(a) = \langle\{p,q\},\{p\}\rangle$. Clearly, B_1 is a model of P. Moreover, it is an inconsistent model — the expert p believes both $\mathbf{in}(a)$ and $\mathbf{out}(a)$. Let us notice though that this inconsistency is not disallowed by the program. The rule $(\mathbf{in}(a):\{q\}) \leftarrow (\mathbf{out}(a):\{p\})$ is applicable with respect to B_1 and, thus, provides an explicit evidence that q believes in $\mathbf{in}(a)$. This fact implicitly precludes q from believing in $\mathbf{out}(a)$. However, this rule does not preclude that expert p believes in $\mathbf{out}(a)$. In addition, since no rule in the program provides any information about $\mathbf{out}(a)$, it prevents neither p nor q from believing in $\mathbf{in}(a)$. To summarize, the program allows for p to have inconsistent beliefs (however, q's beliefs must be consistent).

Next, consider the $\mathcal{T}^2$-valuation B_2 such that $B_2(a) = \langle\{p,q\},\{p,q\}\rangle$. This valuation is also a model of P. In B_2 both p and q are inconsistent in their beliefs. As before, the inconsistent beliefs of p are not disallowed by P. However, reasoning as before we see that the program disallows q to believe in $\mathbf{out}(a)$. Thus the inconsistent beliefs of expert q cannot be reconciled with P. In our study of annotated revision programs we will restrict ourselves only to consistent models and to those inconsistent models whose all inconsistencies are not disallowed by the program.

Speaking more formally, by direct (or explicit) evidence we mean evidence provided by heads of program rules applicable with respect to B. It can be described as $T_P^b(B)$. The implicit bound on allowed annotations is given by a version of the closed world assumption: if the evidence for a revision atom l provided by the program is α then, the evidence for the dual revision atom l^D ($\mathbf{in}(a)$, if $l = \mathbf{out}(a)$, or $\mathbf{out}(a)$, otherwise) must not exceed $\bar{\alpha}$ (unless explicitly forced by the program). Thus, the implicit upper bound on allowed annotations is given by $\perp T_P^b(B)$. Hence, a model B of a program P contains no more evidence than what is implied by P given B if $B \leq_k T_P^b(B) \oplus (\perp T_P^b(B))$. This discussion leads us to a refinement of the notion of a model of an annotated revision program.

Definition 1. *Let P be an annotated revision program and let B be a $\mathcal{T}^2$-valuation. We say B is a* c-model *of P if*

$$T_P^b(B) \leq_k B \leq_k T_P^b(B) \oplus (\perp T_P^b(B)).$$

Thus, coming back to our example, the $\mathcal{T}^2$-valuation B_1 is a c-model of P and B_2 is not.

The "c" in the term c-model is to emphasize that c-models are "as consistent as possible", that is, inconsistencies are limited to those that are not explicitly or implicitly disallowed by the program. The notion of a c-model will play an important consideration in our considerations.

Clearly, by Theorem 2, a c-model of P is a model of P. In addition, it is easy to see that the necessary change of an annotated program P is a c-model of P (it follows directly from the fact that $NC(P) = T_P^b(NC(P))$).

The distinction between models and c-models appears only in the context of inconsistent information. This observation is formally stated below.

Theorem 3. *Let P be an annotated revision program. A consistent $\mathcal{T}^2$-valuation B is a c-model of P if and only if B is a model of P.*

4 Justified revisions

In this section, we will extend to the case of annotated revision programs the notion of a justified revision introduced for revision programs in [MT95]. The reader is referred to [MT95,MT98] for the discussion of motivation and intuitions behind the concept of a justified revision and of the role of the *inertia principle* (a version of the closed world assumption).

There are several properties that one would expect to hold when the notion of justified revision is extended to the case of programs with annotations. Clearly, the extended concept should specialize to the original definition if annotations can be dropped. Next, all main properties of justified revisions studied in [MT98,MPT99] should have their counterparts in the case of justified revisions of annotated programs. In particular, justified revisions of an annotated logic program should satisfy it. Finally, there is one other requirement that naturally arises in the context of programs with annotations.

Consider two annotated revision rules r and r' that are exactly the same except that the body of r contains two annotated revision atoms $l{:}\beta_1$ and $l{:}\beta_2$, while the body of r' instead of $l{:}\beta_1$ and $l{:}\beta_2$ contains annotated revision atom $l{:}\beta_1 \vee \beta_2$:

$$r = \qquad \ldots \leftarrow \ldots, (l{:}\beta_1), \ldots, (l{:}\beta_2), \ldots$$

$$r' = \qquad \ldots \leftarrow \ldots, (l{:}\beta_1 \vee \beta_2), \ldots$$

It is clear, that for any $\mathcal{T}^2$-valuation B, B satisfies $(l{:}\beta_1)$ and $(l{:}\beta_2)$ if and only if B satisfies $(l{:}\beta_1 \vee \beta_2)$. Consequently, replacing rule r by rule r' (or vise versa) in an annotated revision program should have no effect on justified revisions In fact, any reasonable semantics for annotated revision programs should be invariant under such operation, and we will refer to this property of a semantics of annotated revision programs as *invariance under join*.

In this section we introduce the notion of the justified revision of an annotated revision program and contrast it with an earlier proposal by Fitting [Fit95]. In the following section we show that our concept of a justified revision satisfies all the requirements listed above.

Let a $\mathcal{T}^2$-valuation B_I represent our current knowledge about some subset of the universe U. Let an annotated revision program P describe an update that B_I should be subject to. The goal is to identify a class of $\mathcal{T}^2$-valuations that could be viewed as representing updated information about the subset, obtained by revising B_I by P. As argued in [MT95,MT98], each appropriately "revised" valuation B_R must be grounded in P and in B_I, that is, any difference between B_I and the revised $\mathcal{T}^2$-valuation B_R must be justified by means of the program and the information available in B_I.

To determine whether B_R is grounded in B_I and P, we use the *reduct* of P with respect to the two valuations. The construction of reduct consists of two steps and mirrors the original definition of the reduct of an unannotated revision program [MT98]. In the first step, we eliminate from P all rules whose bodies are not satisfied by B_R (their use does not have an *a posteriori* justification with respect to B_R). In the second step, we take into account the initial valuation B_I.

How can we use the information about the initial $\mathcal{T}^2$-valuation B_I at this stage? Assume that B_I provides evidence α for a revision atom l. Assume also that an annotated revision atom $(l{:}\beta)$ appears in the body of a rule r. In order to satisfy this premise of the rule, it is enough to derive, from the program resulting from step 1, an annotated revision atom $(l{:}\gamma)$, where $\alpha \vee \gamma \geq \beta$. The least such element exists (due to the fact that $\mathcal{T}$ is complete and distributive). Let us denote it by $pcomp(\alpha, \beta)$[2].

Thus, in order to incorporate information about a revision atom l contained in the initial $\mathcal{T}^2$-valuation B_I, which is given by $\alpha = (\theta^{-1}(B_I))(l)$, we proceed

[2] The operation $pcomp(\cdot,\cdot)$ is known in the lattice theory as the *relative pseudocomplement*, see [RS70].

as follows. In the bodies of rules of the program obtained after step 1, we replace each annotated revision atom of the form $(l{:}\beta)$ by the annotated revision atom $(l{:}pcomp(\alpha, \beta))$.

Now we are ready to formally introduce the notion of *reduct* of an annotated revision program P with respect to the pair of $\mathcal{T}^2$-valuations, initial one, B_I, and a candidate for a revised one, B_R.

Definition 2. *The* reduct $P_{B_R}|B_I$ *is obtained from* P *by*

1. *removing every rule whose body contains an annotated atom that is not satisfied in* B_R*,*
2. *replacing each annotated atom* $(l{:}\beta)$ *from the body of each remaining rule by the annotated atom* $(l{:}\gamma)$*, where* $\gamma = pcomp((\theta^{-1}(B_I))(l), \beta)$.

We now define the concept of a *justified revision.* Given an annotated revision program P, we first compute the reduct $P_{B_R}|B_I$ of the program P with respect to B_I and B_R. Next, we compute the necessary change for the reduced program. Finally we apply the change thus computed to the $\mathcal{T}^2$-valuation B_I. A $\mathcal{T}^2$-valuation B_R is a justified revision of B_I if the result of these three steps is B_R. Thus we have the following definition.

Definition 3. B_R *is a* P-justified revision *of* B_I *if* $B_R = (B_I \otimes \bot C) \oplus C$*, where* $C = NC(P_{B_R}|B_I)$ *is the necessary change for* $P_{B_R}|B_I$.

We will now contrast the above approach with one proposed by Fitting in [Fit95]. In order to do so, we recall the definitions introduced in [Fit95]. The key difference is in the way Fitting defines the reduct of a program. The first step is the same in both approaches. However, the second steps, in which the initial valuation is used to simplify the bodies of the rules not eliminated in the first step of the construction, differ.

Definition 4 (Fitting). *Let* P *be an annotated revision program and let* B_I *and* B_R *be* $\mathcal{T}^2$*-valuations. The* F-reduct *of* P *with respect to* (B_I, B_R) *(denoted* $P^F_{B_R}|B_I$*) is defined as follows:*

1. *Remove from* P *every rule whose body contains an annotated revision atom that is not satisfied in* B_R.
2. *From the body of each remaining rule delete any annotated revision atom that is satisfied in* B_I.

The notion of justified revision as defined by Fitting differs from our notion only in that the necessary change of the F-reduct is used. We call the justified revision using the notion of F-reduct, the *F-justified revision.*

In the remainder of this section we show that the notion of the F-justified revision does not in general satisfy some basic requirements that we would like justified revisions to have. In particular, F-justified revisions under an annotated revision program P are not always models of P.

Example 1. Consider the lattice $\mathcal{T}_{\{p,q\}}$. Let P be a program consisting of the following rules:

$$(\mathbf{in}(a){:}\{p\}) \leftarrow (\mathbf{in}(b){:}\{p,q\}) \quad \text{and} \quad (\mathbf{in}(b){:}\{q\}) \leftarrow$$

and let B_I be an initial valuation such that $B_I(a) = \langle\emptyset,\emptyset\rangle$ and $B_I(b) = \langle\{p\},\emptyset\rangle$. Let B_R be a valuation given by $B_R(a) = \langle\emptyset,\emptyset\rangle$ and $B_R(b) = \langle\{p,q\},\emptyset\rangle$. Clearly, $P^F_{B_R}|B_I = P$, and B_R is an F-justified revision of B_I (under P). However, B_R does not satisfy P.

The semantics of F-justified revisions also fails to satisfy the invariance under join property.

Example 2. Let P be a revision program consisting of the following rules:

$$(\mathbf{in}(a){:}\{p\}) \leftarrow (\mathbf{in}(b){:}\{p,q\}) \quad \text{and} \quad (\mathbf{in}(b){:}\{q\}) \leftarrow$$

and let P' consist of

$$(\mathbf{in}(a){:}\{p\}) \leftarrow (\mathbf{in}(b){:}\{p\}), (\mathbf{in}(b){:}\{q\}) \quad \text{and} \quad (\mathbf{in}(b){:}\{q\}) \leftarrow$$

Let the initial valuation B_I be given by $B_I(a) = \langle\emptyset,\emptyset\rangle$ and $B_I(b) = \langle\{p\},\emptyset\rangle$. The only F-justified revision of B_I (under P) is a $\mathcal{T}^2$-valuation B_R, where $B_R(a) = \langle\emptyset,\emptyset\rangle$ and $B_R(b) = \langle\{p,q\},\emptyset\rangle$. The only F-justified revision of B_I (under P') is a $\mathcal{T}^2$-valuation B'_R, where $B'_R(a) = \langle\{p\},\emptyset\rangle$ and $B'_R(b) = \langle\{p,q\},\emptyset\rangle$. Thus, replacing in the body of a rule annotated revision atom $(\mathbf{in}(b){:}\{p,q\})$ by $(\mathbf{in}(b){:}\{p\})$ and $(\mathbf{in}(b){:}\{q\})$ affects F-justified revisions.

However, in some cases the two definitions of justified revision coincide. The following result provides a complete characterization of those cases.

Theorem 4. *F-justified revisions and justified revisions coincide if and only if the lattice $\mathcal{T}$ is linear (that is, for any two elements $a, b \in \mathcal{T}$ either $a \leq b$ or $b \leq a$).*

Theorem 4 explains why the difference between the justified revisions and F-justified revisions is not seen when we limit our attention to revision programs as those considered in [MT98]. Namely, the lattice $\mathcal{TWO} = \{\mathbf{f},\mathbf{t}\}$ of boolean values is linear. Similarly, the lattice of reals from the segment $[0,1]$ is linear, and there the differences cannot be seen either.

5 Properties of justified revisions

In this section we study basic properties of justified revisions. We show that key properties of justified revisions in the case of revision programs without annotations have their counterparts in the case of justified revisions of annotated revision programs.

First, we will observe that revision programs as defined in [MT95] can be encoded as annotated revision programs (with annotations taken from the lattice $\mathcal{TWO} = \{\mathbf{f}, \mathbf{t}\}$). Namely, a revision rule

$$p \leftarrow q_1, \ldots q_m$$

(where p and all q_is are revision atoms) can be encoded as

$$(p{:}\mathbf{t}) \leftarrow (q_1{:}\mathbf{t}), \ldots, (q_m{:}\mathbf{t})$$

In [Fit95], Fitting argued that under this encoding the semantics of F-justified revisions generalizes the semantics of justified revisions introduced in [MT95]. Since for lattices whose ordering is linear the approach by Fitting and the approach presented in this paper coincide, and since the ordering of $\mathcal{TWO}$ is linear, the semantics of justified revisions discussed here extends the semantics of justified revisions from [MT95].

Next, let us recall that in the case of revision programs without annotations, justified revisions under a revision program P are models of P. In the case of annotated revision programs we have a similar result.

Theorem 5. *Let P be an annotated revision program and let B_I and B_R be $\mathcal{T}^2$-valuations. If B_R is a P-justified revision of B_I then B_R is a c-model of P (and, hence, also a model of P).*

In the case of revision programs without annotations, a model of a program P is its unique P-justified revision. In the case of programs with annotations, the situation is slightly more complicated. The next result characterizes those models of an annotated revision program that are their own justified revisions.

Theorem 6. *Let a $\mathcal{T}^2$-valuation B_I be a model of an annotated revision program P. Then, B_I is a P-justified revision of itself if and only if B_I is a c-model of P.*

As we observed above, in the case of programs without annotations, models of a revision program are their own *unique* justified revisions. This property does not hold, in general, in the case of annotated revision programs.

Example 3. Consider an annotated revision program P (with annotations belonging to $\mathcal{T}_{\{p,q\}}$) consisting of the clauses:

$$(\mathbf{out}(a){:}\{q\}) \leftarrow \quad \text{and} \quad (\mathbf{in}(a){:}\{q\}) \leftarrow (\mathbf{in}(a){:}\{q\})$$

Consider a $\mathcal{T}^2$-valuation B_I such that $B_I(a) = \langle \{q\}, \{q\} \rangle$. It is easy to see that B_I is a c-model of P. Hence, B_I is its own justified revision (under P).

However, B_I is not the only P-justified revision of B_I. Consider the $\mathcal{T}^2$-valuation B_R such that $B_R(a) = \langle \emptyset, \{q\} \rangle$. We have $P_{B_R}|B_I = \{(\mathbf{out}(a){:}\{q\}) \leftarrow\}$. Let us denote the corresponding necessary change, $NC(P_{B_R}|B_I)$, by C. Then, $C(a) = \langle \emptyset, \{q\} \rangle$. Hence, $\perp C = \langle \{p\}, \{p, q\} \rangle$ and $((B_I \otimes \perp C) \oplus C)(a) = \langle \emptyset, \{q\} \rangle = B_R(a)$. Consequently, B_R is a P-justified revision of B_I.

The same behavior can be observed in the case of programs annotated with elements from other lattices.

Example 4. Let P be an annotated revision program (annotations belong to the lattice $\mathcal{T}_{[0,1]}$) consisting of the rules:

$$(\mathbf{out}(a){:}1) \leftarrow \quad \text{and} \quad (\mathbf{in}(a){:}0.4) \leftarrow (\mathbf{in}(a){:}0.4)$$

Let B_I be a valuation such that $B_I(a) = \langle 0.4, 1\rangle$. Then, B_I is a c-model of P and, hence, it is its own P-justified revision. Consider a valuation B_R such that $B_R(a) = \langle 0, 1\rangle$. We have $P_{B_R}|B_I = \{(\mathbf{out}(a){:}1) \leftarrow\}$. Let us denote the necessary change $NC(P_{B_R}|B_I)$ by C. Then $C(a) = \langle 0, 1\rangle$ and $\bot C = \langle 0, 1\rangle$. Thus, $((B_I \otimes \bot C) \oplus C)(a) = \langle 0, 1\rangle = B_R(a)$. That is, B_R is a P-justified revision of B_I.

Note that in both examples the additional justified revision B_R of B_I is smaller than B_I with respect to the ordering $\leq_k$. It is not coincidental as demonstrated by our next result.

Theorem 7. *Let B_I be a model of an annotated revision program P. Let B_R be a P-justified revision of B_I. Then, $B_R \leq_k B_I$.*

Finally, we observe that if a *consistent* $\mathcal{T}^2$-valuation is a model (or a c-model; these notions coincide in the class of consistent valuations) of a program then, it is its *unique* justified revision.

Theorem 8. *Let B_I be a consistent model of an annotated revision program P. Then, B_I is the only P-justified revision of itself.*

To summarize, when we consider inconsistent valuations (they appear naturally, especially when we measure beliefs of groups of independent experts), we encounter an interesting phenomenon. An *inconsistent* valuation B_I, even when it is a model of a program, may have different justified revisions. However, all these additional revisions must be less informative than B_I. In the case of consistent models this phenomenon does not occur. If a valuation B is consistent and satisfies P then it is its unique P-justified revision.

6 An alternative way of describing annotated revision programs and order-isomorphism theorem

We will now provide an alternative description of annotated revision programs. Instead of evaluating separately revision atoms (i.e. expressions of the form $\mathbf{in}(a)$ and $\mathbf{out}(a)$) we will evaluate atoms. However, instead of evaluating revision atoms in $\mathcal{T}$, we will evaluate atoms in $\mathcal{T}^2$ (i.e. $\mathcal{T} \times \mathcal{T}$). This alternative presentation will allow us to obtain a result on the preservation of justified revisions under order isomorphisms of $\mathcal{T}^2$. This result is a generalization of the "shift theorem" of [MPT99].

An expression of the form $a\langle\alpha,\beta\rangle$, where $\langle\alpha,\beta\rangle \in \mathcal{T}^2$, will be called an *annotated atom* (thus, annotated atoms are *not* annotated revision atoms). Intuitively, an atom $a\langle\alpha,\beta\rangle$ stands for both $(\mathbf{in}(a){:}\alpha)$ and $(\mathbf{out}(a){:}\beta)$. An *annotated rule* is an expression of the form $p \leftarrow q_1,\ldots,q_n$ where $p,q_1,\ldots,q_n$ are annotated atoms. An *annotated program* is a set of annotated rules.

A $\mathcal{T}^2$-valuation B satisfies an annotated atom $a\langle\alpha,\beta\rangle$ if $\langle\alpha,\beta\rangle \leq_k B(a)$. This notion of satisfaction can be extended to annotated rules and annotated programs.

We will now define the notions of reduct, necessary change and justified revision for the new kind of program. The reduct of a program P with respect to two valuations B_I and B_R is defined in a manner similar to Definition 2. Specifically, we leave only the rules with bodies that are satisfied by B_R, and in the remaining rules we reduce the annotated atoms (except that now the transformation θ is no longer needed!). Next, we compute the least fixpoint of the operator associated with the reduced program. Finally, as in Definition 3, we define the concept of justified revision of a valuation B_I with respect to a revision program P.

It turns out that this new syntax does not lead to a new notion of justified revision. Since we talk about two different syntaxes, we will use the term "old syntax" to denote the revision programs as defined in Section 2, and "new syntax" to describe programs introduced in this section. Specifically we now exhibit two mappings. The first of them, tr_1, assigns to each "old" in-rule

$$(\mathbf{in}(a){:}\alpha) \leftarrow (\mathbf{in}(b_1){:}\alpha_1),\ldots,(\mathbf{in}(b_m){:}\alpha_m),(\mathbf{out}(s_1){:}\beta_1),\ldots,(\mathbf{out}(s_n){:}\beta_n),$$

a "new" rule

$$a\langle\alpha,\bot\rangle \leftarrow b_1\langle\alpha_1,\bot\rangle,\ldots,b_m\langle\alpha_m,\bot\rangle,s_1\langle\bot,\beta_1\rangle,\ldots,s_n\langle\bot,\beta_n\rangle.$$

Encoding of an "old" out-rule

$$(\mathbf{out}(a){:}\beta) \leftarrow (\mathbf{in}(b_1){:}\alpha_1),\ldots,(\mathbf{in}(b_m){:}\alpha_m),(\mathbf{out}(s_1){:}\beta_1),\ldots,(\mathbf{out}(s_n){:}\beta_n)$$

is analogous:

$$a\langle\bot,\beta\rangle \leftarrow b_1\langle\alpha_1,\bot\rangle,\ldots,b_m\langle\alpha_m,\bot\rangle,s_1\langle\bot,\beta_1\rangle,\ldots,s_n\langle\bot,\beta_n\rangle.$$

Translation tr_2, in the other direction, replaces a revision "new" rule by one in-rule and one out-rule. Specifically, a "new" rule

$$a\langle\alpha,\beta\rangle \leftarrow a_1\langle\alpha_1,\beta_1\rangle,\ldots,a_n\langle\alpha_n,\beta_n\rangle$$

is replaced by two "old" rules (with identical bodies but different heads)

$$(\mathbf{in}(a){:}\alpha) \leftarrow (\mathbf{in}(a_1){:}\alpha_1),(\mathbf{out}(a){:}\beta_1),\ldots,(\mathbf{in}(a_n){:}\alpha_n),(\mathbf{out}(a_n){:}\beta_n)$$

and

$$(\mathbf{out}(a){:}\beta) \leftarrow (\mathbf{in}(a_1){:}\alpha_1),(\mathbf{out}(a){:}\beta_1),\ldots,(\mathbf{in}(a_n){:}\alpha_n),(\mathbf{out}(a_n){:}\beta_n).$$

The translations tr_1 and tr_2 can be extended to programs. We then have the following theorem.

Theorem 9. *Both transformations tr_1, and tr_2 preserve justified revisions. That is, if B_I, B_R are valuations in $\mathcal{T}^2$ and P is a program in the "old" syntax, then B_R is a P-justified revision of B_I if and only if B_R is a $tr_1(P)$-justified revision of B_I. Similarly, if B_I, B_R are valuations in $\mathcal{T}^2$ and P is a program in the "new" syntax, then B_R is a P-justified revision of B_I if and only if B_R is a $tr_2(P)$-justified revision of B_I.*

In the case of unannotated revision programs, the shifting theorem proved in [MPT99] shows that for every revision program P and every two initial databases B and B' there is a revision program P' such that there is a one-to-one correspondence between P-justified revisions of B and P'-justified revisions of B'. In particular, it follows that the study of justified revisions (for unannotated programs) can be reduced to the study of justified revisions of empty databases. We will now present a counterpart of this result for annotated revision programs. The situation here is more complex. It is no longer true that a $\mathcal{T}^2$-valuation can be "shifted" to any other $\mathcal{T}^2$-valuation. However, the shift is possible if the two valuations are related to each other by an order isomorphism of the lattice of all $\mathcal{T}^2$-valuations.

There are many examples of order isomorphisms on the lattice of $\mathcal{T}^2$-valuations. For instance, the mapping $\psi : \mathcal{T}^2 \rightarrow \mathcal{T}^2$ defined by $\psi(\langle\alpha, \beta\rangle) = \langle\beta, \alpha\rangle$ is an order isomorphism of $\mathcal{T}^2$. In the case of a specific lattice $\mathcal{T}_X$, other order isomorphisms of $\mathcal{T}_X^2$ are generated by permutations of the set X. An order isomorphism on $\mathcal{T}^2$ can be extended to annotated atoms, programs and valuations. The extension to valuations is again an order isomorphism, this time on the lattice of all $\mathcal{T}^2$-valuations.

The following result generalizes the shifting theorem of [MPT99].

Theorem 10. *Let ψ be an order-isomorphism on the set of $\mathcal{T}^2$-valuations. Then, B_R is a P-justified revision of B_I if and only if $\psi(B_R)$ is a $\psi(P)$-justified revision of $\psi(B_I)$.*

7 Conclusions and further research

The main contribution of our paper is a new definition of the reduct (and hence of justified revision) for the annotated programs considered by Fitting in [Fit95]. This new definition eliminates some anomalies (specifically the fact that the justified revisions of [Fit95] do not have to be models of the program). We also found that in cases where the intuition of [Fit95] is very clear (for instance in case when annotations are numerical degrees of belief), the two concepts coincide.

Due to the limited space of the extended abstract, some results were not included. Below we briefly mention two research areas that are not discussed here but that will be discussed in the full version of the paper.

First, the annotation programs can be generalized to disjunctive case, that is to programs admitting "nonstandard disjunctions" in the heads of rules. It turns out that a definition of justified revisions by means of such programs is possible, and one can prove that the disjunctive revisions for programs that have the head consisting of just one literal reduce to the formalism described above.

Second, one can extend the formalism of annotated revision programs to the case when the lattice of annotations is not distributive. However, in such case only some of the results discussed here still hold.

8 Acknowledgments

This work was partially supported by the NSF grants CDA-9502645 and IRI-9619233.

References

[ALP$^+$98] J.J. Alferes, J.A. Leite, L.M. Pereira, H. Przymusinska, and T.C. Przymusinski. Dynamic logic programming. In *Proceedings of KR'98: Sixth International Conference on Principles of Knowledge Representation and Reasoning, Trento, Italy*, pages 98 – 110. Morgan Kaufmann, 1998.

[Fit95] M. C. Fitting. Annotated revision specification programs. In *Logic programming and nonmonotonic reasoning (Lexington, KY, 1995)*, volume 928 of *Lecture Notes in Computer Science*, pages 143–155. Springer-Verlag, 1995.

[Fit99] M. C. Fitting. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science*, 1999. To appear.

[Gin88] M.L. Ginsberg. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[KS92] M. Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programs and its applications. *Journal of Logic Programming*, 12:335–367, 1992.

[LW92] V. Lifschitz and T.Y.C. Woo. Answer sets in general nonmonotonic reasoning. In *Proceedings of the 3rd international conference on principles of knowledge representation and reasoning, KR '92*, pages 603–614, San Mateo, CA, 1992. Morgan Kaufmann.

[MPT99] W. Marek, I. Pivkina, and M. Truszczyński. Revision programming = logic programming + integrity constraints. In *Computer Science Logic, 12th International Workshop, CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 73–89. Springer, 1999.

[MT95] W. Marek and M. Truszczyński. Revision programming, database updates and integrity constraints. In *Proceedings of the 5th International Conference on Database Theory — ICDT 95*, volume 893 of *Lecture Notes in Computer Science*, pages 368–382. Springer-Verlag, 1995.

[MT98] W. Marek and M. Truszczyński. Revision programming. *Theoretical Computer Science*, 190(2):241–277, 1998.

[PT97] T. C. Przymusinski and H. Turner. Update by means of inference rules. *Journal of Logic Programming*, 30(2):125–143, 1997.

[RS70] H. Rasiowa and R. Sikorski. *The Mathematics of metamathematics*. PWN—Polish Scientific Publishers, Warsaw, 1970.