ELSEVIER
Artificial Intelligence 138 (2002) 149–180
Artificial Intelligence
www.elsevier.com/locate/artint

Annotated revision programs

Victor Marek, Inna Pivkina, Mirosław Truszczyński *

Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA

Received 15 May 2000

Abstract

Revision programming is a formalism to describe and enforce updates of belief sets and databases. That formalism was extended by Fitting who assigned annotations to revision atoms. Annotations provide a way to quantify the confidence (probability) 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 annotated case.

Keywords: Knowledge representation; Database updates; Belief revision; Revision programming; Annotated programs

1. Introduction

Revision programming is a formalism to specify and enforce constraints on databases, belief sets and, more generally, on arbitrary sets. Revision programming was introduced and studied in [10,11]. The formalism was shown to be closely related to logic programming with stable model semantics [11,13]. In [9], a simple correspondence of revision programming with the general logic programming system of Lifschitz and Woo [8] was discovered. Roots of another recent formalism of dynamic logic programming [1] can also be traced back to revision programming.

* Corresponding author.

E-mail addresses: marek@cs.engr.uky.edu (V. Marek), inna@cs.engr.uky.edu (I. Pivkina), mirek@cs.engr.uky.edu (M. Truszczyński).

0004-3702/02/$ – see front matter
PII: S0004-3702(02)00184-4

(Unannotated) 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 interpretation: whenever all elements a_k, $1 \leqslant k \leqslant m$, belong to the current set (database, belief set) and none of the elements b_l, $1 \leqslant l \leqslant n$, belongs to the current set then, in the case of rule (1), the item a should be in the revised set, and in the case of rule (2), a should not be in the revised set.

Let us illustrate the use of the revision rules by an example.

Example 1.1. Let program P consist of the following two rules.

$$\mathbf{in}(b) \leftarrow \mathbf{out}(c) \quad \text{and} \quad \mathbf{in}(c) \leftarrow \mathbf{in}(a), \mathbf{out}(b).$$

When the current set (initial database) has only atom a in it, there are two intended revisions. One of them consists of a and b. The other one consists of a and c. If, however, the initial database is empty, there is only one intended revision consisting of atom b.

To provide a precise semantics to *revision programs* (collections of revision rules), the concept of a *justified revision* was introduced in [10,11]. 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. The intended revisions discussed in Example 1.1 are P-justified revisions.

The formalism of revision programs was extended by Fitting [4] to the case when revision atoms occurring in rules are assigned *annotations*. Such annotation 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 the probability 0.2. Thus, annotated atoms and annotated revision programs can be used to model situations when membership status of atoms (whether they are "in" or "out") is not precisely known and when constraints reflect this imprecise knowledge. In his work, Fitting defined the concept of an annotated revision program, described the concept of a justified revision of a database by an annotated revision program, and studied properties of that notion.

The annotations do not have to be numeric. In fact they may come from any set. It is natural, though, to assume that the set of annotations has a mathematical structure of a complete distributive lattice. Such lattices allow us to capture within a single algebraic formalism different intuitions associated with annotations. For instance, annotations expressing probabilities [12], possibilistic annotations [2], and annotations in terms of opinions of groups of experts [4] can all be regarded as elements of certain complete and distributive lattices. The general formalism of lattice-based annotations was studied by Kifer and Subrahmanian [7] but only for logic programs without negations.

In the setting of logic programs, an annotation describes the probability (or the degree of belief) that an atom is implied by a program or, that it is "in" a database. The closed world assumption then implies the probability that an atom is "out". Annotations in the context of revision programs provide us with richer descriptions of the status of atoms. Specifically, a possible interpretation of a *pair* of annotated revision literals ($\mathbf{in}(a){:}\alpha$) and ($\mathbf{out}(a){:}\beta$) is that our confidence in a being in a database is α and that, in the same time, our confidence that a does not belong to the database is β. Annotating atoms with *pairs* of annotations allows us to model incomplete and contradictory information about the status of an atom.

Thus, in annotated revision programming the status of an atom a is, in fact, given by a pair of annotations. Therefore, in this paper we will consider, in addition to a lattice of annotations, which we will denote by $\mathcal{T}$, the product of $\mathcal{T}$ by itself—the lattice $\mathcal{T}^2$. There are two natural orderings on $\mathcal{T}^2$. We will use one of them, the *knowledge* ordering, to compare the degree of incompleteness (or degree of contradiction) of the pair of annotations describing the status of an atom.

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 use the concept of an *s-model* which is a refinement of 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.

Here is a short description of the content and the contributions of our paper. In Section 2, we introduce annotated revision programs, provide some examples and discuss underlying motivations. We define the concepts of a valuation of a set of revision atoms in a lattice of annotations $\mathcal{T}$ and of a valuation of a set of (ordinary) atoms in the corresponding product lattice $\mathcal{T}^2$. We also define the knowledge ordering on $\mathcal{T}^2$ and on valuations of atoms in $\mathcal{T}^2$.

Given an annotated revision program, we introduce the notion of the operator associated with the program. This operator acts on valuations in $\mathcal{T}^2$ and is analogous to the van Emden–Kowalski operator for logic programs [3]. It is monotone with respect to the knowledge ordering and allows us to introduce the notion of the necessary change entailed by an annotated revision program.

In Section 3, we introduce one of the two main concepts of this paper, namely that of an s-model of a revision program. Models of annotated revision programs may be inconsistent. In the case of an s-model, if it is inconsistent, its inconsistencies are explicitly or implicitly supported by the program and the model itself. We contrast the notion of an s-model with that of a model. We show that in general the two concepts are different. However, we also show that under the assumption of consistency they coincide.

In Section 4, we define the notion of a justified revision of an annotated database by an annotated revision program P. Such revisions are referred to as P-justified revisions. They are defined so as to generalize justified revisions of [10,11].

Justified revisions considered here are different from those introduced by Fitting in [4]. We provide examples that show that Fitting's concept of a justified revision fails to satisfy some natural postulates and argue that our proposal more adequately models intuitions associated with annotated revision programs. In the same time, we provide a complete characterization of those lattices for which both proposals coincide. In particular, they coincide in the standard case of revision programs without annotations.

We study the properties of justified revisions in Section 5. We show that annotated revision programs with the semantics of justified revisions generalize revision programming as introduced and studied in [10,11]. Next, we show that P-justified revisions are s-models of the program P. Thus, the concept of an s-model introduced in Section 2 is an appropriate refinement of the notion of a model to be used in the studies of justified revisions. Further, we prove that P-justified revisions decrease inconsistency and, consequently, that a consistent model of a program P is its own unique P-justified revision.

Throughout the paper we adhere to the syntax of annotated revision programs proposed by Fitting in [4]. This syntax stems naturally from the syntax of ordinary revision programs introduced in [10,11] and allows us to compare directly our approach with that of Fitting. However, in Section 6, we propose and study an alternative syntax for annotated revision programs. In this new syntax (ordinary) atoms are annotated by elements of the product lattice $\mathcal{T}^2$. Using this alternative syntax, we obtain an elegant generalization of the shifting theorem of [9].

In Section 7, we provide a brief account of some miscellaneous results on annotated revision programs. In particular, we discuss the case of programs with disjunctions in the heads and the case when the lattice of annotations is not distributive.

2. Preliminaries

We will start with examples that illustrate main notions and a possible use of annotated revision programming. Formal definitions will follow.

Example 2.1. A group of experts is about to discuss a certain proposal and then vote whether to accept or reject it. Each person has an opinion on the proposal that may be changed during the discussion as follows:

- any person can convince an optimist to vote for the proposal,
- any person can convince a pessimist to vote against the proposal.

The group consists of two optimists (Ann and Bob) and one pessimist (Pete). We want to be able to answer the following question: given everybody's opinion on the subject before the discussion, what are the possible outcomes of the vote?

Assume that before the vote Pete is for the proposal, Bob is against, and Ann is indifferent (has no arguments for and no arguments against the proposal). This situation can be described by assigning to atom "*accept*" the annotation $\langle\{Pete\}, \{Bob\}\rangle$, where the first element of the pair is the set of experts who have arguments for the acceptance of the proposal and the second element is the set of experts who have arguments against the proposal. In the formalism of annotated revision programs, as proposed by Fitting in [4], this initial situation is described by a *function* that assigns to each atom in the language (in this example there is only one atom) its annotation. In our example, this function is given by: $B_I(accept) = \langle\{Pete\}, \{Bob\}\rangle$. (Let us mention here that in general, the sets of experts in an annotation need not to be disjoint. An expert may have arguments for and against the proposal at the same time. In such a case the expert is contradictory.)

The ways in which opinions may change are described by the following annotated revision rules:

$$(\mathbf{in}(accept){:}\{Ann\}) \leftarrow (\mathbf{in}(accept){:}\{Bob\})$$

$$(\mathbf{in}(accept){:}\{Ann\}) \leftarrow (\mathbf{in}(accept){:}\{Pete\})$$

$$(\mathbf{in}(accept){:}\{Bob\}) \leftarrow (\mathbf{in}(accept){:}\{Ann\})$$

$$(\mathbf{in}(accept){:}\{Bob\}) \leftarrow (\mathbf{in}(accept){:}\{Pete\})$$

$$(\mathbf{out}(accept){:}\{Pete\}) \leftarrow (\mathbf{out}(accept){:}\{Ann\})$$

$$(\mathbf{out}(accept){:}\{Pete\}) \leftarrow (\mathbf{out}(accept){:}\{Bob\})$$

The first rule means that if Bob accepts the proposal, then Ann should accept the proposal, too, since she will be convinced by Bob. Similarly, the second rule means that if Pete has arguments for the proposal, then he will be able to convince Ann. These two rules describe Ann being an optimist. The remaining rules follow as Bob is an optimist and Pete is a pessimist.

Possible outcomes of the vote are given by justified revisions. In this particular case there are two justified revisions of the initial database B_I. They are $B_R(accept) = \langle\{Ann, Bob, Pete\},\ \{\}\rangle$ and $B'_R(accept) = \langle\{\}, \{Bob, Pete\}\rangle$. The first one corresponds to the case when the proposal is accepted (Ann, Bob and Pete all voted for). This outcome happens if Pete convinces Bob and Ann to vote for. The second revision corresponds to the case when Bob and Pete voted against the proposal (Ann remained indifferent and did not vote). This outcome happens if Bob convinces Pete to change his opinion.

Remark 2.2. It is possible to rewrite annotated revision rules from Example 2.1 as ordinary revision rules (without annotations) if we use atoms "*accept_Ann*", "*accept_Bob*", and "*accept_Pete*". However, ordinary revision programs do not deal with inconsistent or not completely defined databases. In particular, we will not be able to express the fact that initially Ann has no arguments for and no arguments against the proposal in Example 2.1.

In the next example annotations are real numbers from the interval $[0, 1]$ representing different degrees of a particular quality.

Example 2.3. Assume that there are two sources of light: a and b. Each of them may be either On or Off. They are used to transmit two signals. The first signal is a combination of a being On and b being Off. The second signal is a combination of a being Off and b being On.

The sources a and b are located far from an observer. Such factors as light pollution and dust may affect the perception of signals. Therefore, the observed brightness of a light source differs from its actual brightness. Assume that brightness is measured on a scale from 0 (complete darkness) to 1 (maximal brightness). The actual brightness of a light source may be either 0 (when it is Off), or 1 (when it is On).

Initial database B_I represents observed brightness of sources. For example, if observed brightness of source a is α ($0 \leqslant \alpha \leqslant 1$), then $B_I(a) = \langle \alpha, 1-\alpha \rangle$. We may think of the first and the second elements in the pair $\langle \alpha, 1-\alpha \rangle$ as degrees of brightness and darkness of the source respectfully. The task is to infer actual brightness from observed brightness. Thus, revision of the initial database should represent actual brightness of sources.

Suppose we know that dust in the air cannot reduce brightness by more than 0.2. Then, we can safely assume that a light source is On if its observed brightness is 0.8 or more. Assume also that light pollution cannot contribute more than 0.4. That is, if observed darkness of a source is at least 0.6, it must be Off. This information together with the fact that only two signals are possible, may be represented by the following annotated revision program P:

$$(\mathbf{in}(a){:}1) \leftarrow (\mathbf{in}(a){:}0.8), (\mathbf{out}(b){:}0.6)$$
$$(\mathbf{out}(b){:}1) \leftarrow (\mathbf{in}(a){:}0.8), (\mathbf{out}(b){:}0.6)$$
$$(\mathbf{in}(b){:}1) \leftarrow (\mathbf{in}(b){:}0.8), (\mathbf{out}(a){:}0.6)$$
$$(\mathbf{out}(a){:}1) \leftarrow (\mathbf{in}(b){:}0.8), (\mathbf{out}(a){:}0.6)$$

The first two rules state that if the brightness of a is at least 0.8 and darkness of b is at least 0.6, then brightness of a is 1 (the first rule) and darkness of b is 1 (the second rule). This corresponds to the case when the first signal is transmitted. Similarly, the last two rules describe the case when the second signal is transmitted.

Let observed brightness of a and b be 0.3 and 0.9 respectively. That is, $B_I(a) = \langle 0.3, 0.7 \rangle$ and $B_I(b) = \langle 0.9, 0.1 \rangle$. Then, P-justified revision of B_I is the actual brightness. In this case we have $B_R(a) = \langle 0, 1 \rangle$ (a is Off), and $B_R(b) = \langle 1, 0 \rangle$ (b is On).

Now let us move on to formal definitions. Throughout the paper we consider a fixed *universe* U whose elements are referred to as *atoms*. In Example 2.1 $U = \{accept\}$. In Example 2.3 $U = \{a, b\}$. 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 infinitely 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 $\leqslant$ and the corresponding meet and join operations by $\wedge$ and $\vee$, respectively. The De Morgan complement of $a \in \mathcal{T}$ is denoted by $\bar{a}$. Let us recall that it satisfies the following two laws (the De Morgan laws):

$$\overline{a \vee b} = \bar{a} \wedge \bar{b}, \qquad \overline{a \wedge b} = \bar{a} \vee \bar{b}.$$

In Example 2.1, $\mathcal{T}$ is the set of subsets of the set $\{Ann, Bob, Pete\}$, with $\subseteq$ as the ordering relation, and the set-theoretic complement as the De Morgan complement. In Example 2.3, $\mathcal{T} = [0, 1]$ with the usual ordering; the De Morgan complement of α is $1 - \alpha$.

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 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 $v(\mathbf{in}(a)) \geqslant \alpha$. Similarly, v *satisfies* $(\mathbf{out}(a){:}\alpha)$ if $v(\mathbf{out}(a)) \geqslant \alpha$. 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 an annotated 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 a $\mathcal{T}$-valuation v. We define an operator T_P as follows:

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

Here $\bigvee X$ is the join of the subset X of the lattice (note that $\perp$ is the join of an empty set of lattice elements). The operator T_P is a counterpart of the well-known 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 $\leqslant_k$ defined by:

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

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 $\alpha_1 \leqslant \alpha_2$ and $\beta_1 \leqslant \beta_2$ imply that the pair $\langle \alpha_2, \beta_2 \rangle$ represents higher degree of knowledge about a. Thus, the ordering $\leqslant_k$ is often referred to as the *knowledge* or *information* ordering. Since the lattice $\mathcal{T}$ is complete and distributive, $\mathcal{T}^2$ is a complete distributive lattice with respect to the ordering $\leqslant_k$.[1]

The operations of meet, join, top, and bottom under $\leqslant_k$ are denoted $\otimes$, $\oplus$, $\top$, and $\perp$, respectively. In addition, we make use of the *conflation* operation. Conflation is defined as $-\langle \alpha, \beta \rangle = \langle \overline{\beta}, \overline{\alpha} \rangle$. An element $A \in \mathcal{T}^2$ is *consistent* if $A \leqslant_k -A$. In other words, an element $\langle \alpha, \beta \rangle \in \mathcal{T}^2$ is consistent if α is smaller than or equal to the complement of β (the evidence "for" is less than or equal than the complement of the evidence "against") and β is smaller than or equal to the complement of α (the evidence "against" is less than or equal than the complement of the evidence "for").

The conflation operation satisfies the De Morgan laws:

$$-(\langle \alpha, \beta \rangle \oplus \langle \gamma, \delta \rangle) = -\langle \alpha, \beta \rangle \otimes -\langle \gamma, \delta \rangle,$$
$$-(\langle \alpha, \beta \rangle \otimes \langle \gamma, \delta \rangle) = -\langle \alpha, \beta \rangle \oplus -\langle \gamma, \delta \rangle,$$

where $\alpha, \beta, \gamma, \delta \in \mathcal{T}$.

[1] There is another ordering that can be associated with $\mathcal{T}^2$. We can define $\langle \alpha_1, \beta_1 \rangle \leqslant_t \langle \alpha_2, \beta_2 \rangle$ if $\alpha_1 \leqslant \alpha_2$ and $\beta_1 \geqslant \beta_2$. This ordering is often called the *truth ordering*. Since $\mathcal{T}$ is a complete distributive lattice, $\mathcal{T}^2$ with both orderings $\leqslant_k$ and $\leqslant_t$ forms a complete distributive *bilattice* (see [5,6] for a definition). In this paper we will not use the ordering $\leqslant_t$ nor the fact that $\mathcal{T}^2$ is a bilattice.

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.

In this paper, $\mathcal{T}^2$-valuations will be used to represent current information about sets (databases) as well as the 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 of B by C, say B', by

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

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 $-C$, unless C directly implies so. In other words, if $C(a) = \langle \alpha, \beta \rangle$, then evidence for $\mathbf{in}(a)$ must not exceed $\bar{\beta}$ unless $\alpha \geqslant \bar{\beta}$, and the evidence for $\mathbf{out}(a)$ must not exceed $\bar{\alpha}$ unless $\beta \geqslant \bar{\alpha}$. Since we prefer explicit evidence of C to implicit evidence expressed by $-C$, we perform the change by first using $-C$ and then applying C. However, let us note here that the order matters only if C is inconsistent; if C is consistent, $(B \otimes -C) \oplus C = (B \oplus C) \otimes -C$. This specification of how the change modeled by a $\mathcal{T}^2$-valuation is enforced plays a key role in our definition of justified revisions in Section 4.

Example 2.4 (*continuation of Example* 2.1). In Example 2.1, B_I has two revisions. The first one, B_R, is the revision of B_I by C, where $C(\mathit{accept}) = \langle \{\mathit{Ann}, \mathit{Bob}\}, \{\} \rangle$. We have $-C(\mathit{accept}) = \langle \{\mathit{Ann}, \mathit{Bob}, \mathit{Pete}\}, \{\mathit{Pete}\} \rangle$. Thus, $(B_I \otimes -C)(\mathit{accept}) = \langle \{\mathit{Pete}\}, \emptyset \rangle$, and $((B_I \otimes -C) \oplus C)(\mathit{accept}) = \langle \{\mathit{Ann}, \mathit{Bob}, \mathit{Pete}\}, \emptyset \rangle = B_R(\mathit{accept})$.

The second revision, B'_R, is the revision of B_I by C', where $C'(\mathit{accept}) = \langle \{\}, \{\mathit{Pete}\} \rangle$.

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, by 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 $\leqslant_k$-monotonicity.

Theorem 2.5. *Let P be an annotated revision program and let B and B' be two $\mathcal{T}^2$-valuations such that $B \leqslant_k B'$. Then, $T_P^b(B) \leqslant_k T_P^b(B')$.*

By Tarski–Knaster Theorem [15] it follows that the operator T_P^b has a least fixpoint in $\mathcal{T}^2$ (see also [7]). 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 [10,11] 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), with the standard ordering $\leqslant$, and the standard complement operation $\bar{\alpha} = 1 - \alpha$. We will denote this lattice by $\mathcal{T}_{[0,1]}$. Intuitively, the annotated revision atom $(\mathbf{in}(a){:}x)$, 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 s-models

The semantics of annotated revision programs will be based on the notion of a model, as defined in the previous section, and on its refinements. The first two results describe some simple properties of models of annotated revision programs. The first of them characterizes models in terms of the operator T_P^b.

Theorem 3.1. *Let P be an annotated revision program. A $\mathcal{T}^2$-valuation B is a model of P (satisfies P) if and only if $B \geqslant_k T_P^b(B)$.*

Models of annotated revision programs are closed under meets. This property is analogous to a similar property holding for models of Horn programs. Indeed, since $B_1 \otimes B_2 \leqslant_k B_i$, $i = 1,2$, and T_P^b is $\leqslant_k$-monotone, by Theorem 3.1 we obtain

$$T_P^b(B_1 \otimes B_2) \leqslant_k T_P^b(B_i) \leqslant_k B_i, \quad i = 1,2.$$

Consequently,

$$T_P^b(B_1 \otimes B_2) \leqslant_k B_1 \otimes B_2.$$

Thus, again by Theorem 3.1 we obtain the following result.

Corollary 3.2. *The meet of two models of an annotated revision program P is also a model of P.*

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 now argue, 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 to those explicitly or implicitly supported by the program and by the model itself.

Consider the program $P = \{(\mathbf{in}(a){:}\{q\}) \leftarrow\}$ (where the annotation $\{q\}$ comes from the lattice $\mathcal{T}_{\{p,q\}}$). This program asserts that a is "in", according to expert q. By closed world assumption, it also implies an upper bound for the evidence for $\mathbf{out}(a)$. In this case the only expert that might possibly believe in $\mathbf{out}(a)$ is p (this is to say that expert q does not believe in $\mathbf{out}(a)$). Observe that a $\mathcal{T}^2$-valuation B, such that $B(a) = \langle\{q\}, \{q\}\rangle$ is a model of P but it does not satisfy the implicit bound on evidence for $\mathbf{out}(a)$.

Let P be an annotated program and let B be a $\mathcal{T}^2$-valuation that is a model of P. By the explicit evidence we mean evidence provided by heads of program rules applicable with respect to B, that is with bodies satisfied by B. It is $T^b_P(B)$. The implicit information is given by a version of the closed world assumption: if the maximum evidence for a revision atom l provided by the program is α then, the evidence for the dual revision atom l^D ($\mathbf{out}(a)$, if $l = \mathbf{in}(a)$, or $\mathbf{in}(a)$, otherwise) must not exceed $\bar{\alpha}$ (unless explicitly forced by the program). Thus, the implicit evidence is given by $-T^b_P(B)$. Hence, a model B of a program P contains no more evidence than what is directly implied by P given B and what is indirectly implied by P given B if $B \leqslant_k T^b_P(B) \oplus (-T^b_P(B))$ (since the direct evidence is given by $T^b_P(B)$ and the implicit evidence is given by $-T^b_P(B)$). This observation leads us to a refinement of the notion of a model of an annotated revision program.

Definition 3.3. Let P be an annotated revision program and let B be a $\mathcal{T}^2$-valuation. We say that B is an *s-model* of P if

$$T^b_P(B) \leqslant_k B \leqslant_k T^b_P(B) \oplus \bigl(-T^b_P(B)\bigr).$$

The "s" in the term "s-model" stands for "supported" and emphasizes that inconsistencies in s-models are limited to those explicitly or implicitly supported by the program and the model itself.

Clearly, by Theorem 3.1, an s-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 an s-model of P (it follows directly from the fact that $NC(P) = T^b_P(NC(P)))$.

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

Theorem 3.4. *Let P be an annotated revision program. A consistent $\mathcal{T}^2$-valuation B is an s-model of P if and only if B is a model of P.*

Proof. ($\Rightarrow$) Let B be an s-model of P. Then, $T^b_P(B) \leqslant_k B \leqslant_k T^b_P(B) \oplus (-T^b_P(B))$. In particular, $T^b_P(B) \leqslant_k B$ and, by Theorem 3.1, B is a model of P.

($\Leftarrow$) Let B satisfy P. From Theorem 3.1 we have $T^b_P(B) \leqslant_k B$. Hence, $-B \leqslant_k -T^b_P(B)$. Since B is consistent, $B \leqslant_k -B$. Therefore,

$$T^b_P(B) \leqslant_k B \leqslant_k -B \leqslant_k -T^b_P(B). \tag{3}$$

It follows that $T^b_P(B) \leqslant_k -T^b_P(B)$ and $T^b_P(B) \oplus (-T^b_P(B)) = -T^b_P(B)$. By (3), we get

$$T^b_P(B) \leqslant_k B \leqslant_k T^b_P(B) \oplus (-T^b_P(B))$$

and the assertion follows. $\square$

Some of the properties of ordinary models hold for s-models, too. For instance, the following theorem shows that an s-model of two annotated revision programs is an s-model of their union.

Theorem 3.5. *Let P_1, P_2 be annotated revision programs. Let B be an s-model of P_1 and an s-model of P_2. Then, B is an s-model of $P_1 \cup P_2$.*

Proof. Clearly, B is a model of $P_1 \cup P_2$. That is,

$$T^b_{P_1 \cup P_2}(B) \leqslant_k B. \tag{4}$$

It is easy to see that

$$T^b_{P_1 \cup P_2}(B) = T^b_{P_1}(B) \oplus T^b_{P_2}(B). \tag{5}$$

Hence, by the De Morgan law,

$$-T^b_{P_1 \cup P_2}(B) = -T^b_{P_1}(B) \otimes -T^b_{P_2}(B). \tag{6}$$

It follows from the definition of an s-model that

$$B \leqslant_k T^b_{P_1}(B) \oplus -T^b_{P_1}(B) \quad \text{and} \quad B \leqslant_k T^b_{P_2}(B) \oplus -T^b_{P_2}(B).$$

Thus,

$$B \leqslant_k \big(T^b_{P_1}(B) \oplus -T^b_{P_1}(B)\big) \otimes \big(T^b_{P_2}(B) \oplus -T^b_{P_2}(B)\big).$$

By the distributivity of lattice operations in $\mathcal{T}^2$, we obtain

$$B \leqslant_k \big(T^b_{P_1}(B) \otimes \big(T^b_{P_2}(B) \oplus -T^b_{P_2}(B)\big)\big) \oplus \big(-T^b_{P_1}(B) \otimes (T^b_{P_2}(B) \oplus -T^b_{P_2}(B))\big).$$

The first summand is smaller or equal to $T^b_{P_1}(B)$. Thus, by applying distributivity to the second summand, we get the following inequality:

$$B \leqslant_k T^b_{P_1}(B) \oplus \big(-T^b_{P_1}(B) \otimes T^b_{P_2}(B)\big) \oplus \big(-T^b_{P_1}(B) \otimes -T^b_{P_2}(B)\big).$$

Using $-T^b_{P_1}(B) \otimes T^b_{P_2}(B) \leqslant_k T^b_{P_2}(B)$ and then (5) and (6), we get

$$B \leqslant_k T^b_{P_1}(B) \oplus T^b_{P_2}(B) \oplus -T^b_{P_1 \cup P_2}(B) = T^b_{P_1 \cup P_2}(B) \oplus -T^b_{P_1 \cup P_2}(B).$$

In other words,

$$B \leqslant_k T^b_{P_1 \cup P_2}(B) \oplus -T^b_{P_1 \cup P_2}(B). \tag{7}$$

From (4) and (7) it follows that B is an s-model of $P_1 \cup P_2$. □

Not all of the properties of models hold for s-models. For instance, the counterpart of Corollary 3.2 does not hold. The following example shows that the meet of two s-models is not necessarily an s-model.

Example 3.6. Consider the lattice $\mathcal{T}_{\{p,q\}}$. Let P be an annotated program consisting of the following rules:

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

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

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

Let B_1 and B_2 be defined as follows.

$$B_1(a) = \langle\{p\}, \{p\}\rangle, \qquad B_1(b) = \langle\{p\}, \emptyset\rangle;$$
$$B_2(a) = \langle\{p\}, \{p\}\rangle, \qquad B_2(b) = \langle\emptyset, \{p\}\rangle.$$

Let us show that B_1 is an s-model of P. Indeed,

$$T_P^b(B_1)(a) = \langle\{p\}, \{p\}\rangle,$$

$$T_P^b(B_1)(b) = \langle\emptyset, \emptyset\rangle.$$

Hence,

$$-T_P^b(B_1)(a) = \langle\{q\}, \{q\}\rangle,$$

$$-T_P^b(B_1)(b) = \langle\{p,q\}, \{p,q\}\rangle.$$

Therefore,

$$T_P^b(B_1)(a) \leqslant_k B_1(a) \leqslant_k \big(T_P^b(B_1) \oplus -T_P^b(B_1)\big)(a),$$

and

$$T_P^b(B_1)(b) \leqslant_k B_1(b) \leqslant_k \big(T_P^b(B_1) \oplus -T_P^b(B_1)\big)(b).$$

In other words, B_1 is an s-model of P. Similarly, B_2 is an s-model of P. However, $B_1 \otimes B_2$ is *not* an s-model of P. Indeed,

$$(B_1 \otimes B_2)(a) = \langle\{p\}, \{p\}\rangle, \qquad (B_1 \otimes B_2)(b) = \langle\emptyset, \emptyset\rangle.$$

Then,

$$T_P^b(B_1 \otimes B_2)(a) = \langle\emptyset, \{p\}\rangle, \qquad T_P^b(B_1 \otimes B_2)(b) = \langle\emptyset, \emptyset\rangle,$$

and

$$-T_P^b(B_1 \otimes B_2)(a) = \langle\{q\}, \{p,q\}\rangle, \qquad -T_P^b(B_1 \otimes B_2)(b) = \langle\{p,q\}, \{p,q\}\rangle.$$

Hence,

$$(B_1 \otimes B_2)(a) \nleqslant_k \big(T_P^b(B_1 \otimes B_2) \oplus -T_P^b(B_1 \otimes B_2)\big)(a) = \langle\{q\}, \{p,q\}\rangle.$$

Therefore, $B_1 \otimes B_2$ is not an s-model of P.

In this example both B_1 and B_2, as well as their meet $B_1 \otimes B_2$ are inconsistent. For B_1 and B_2 there are rules in P that explicitly imply their inconsistencies. However, for $B_1 \otimes B_2$ the bodies of these rules are no longer satisfied. Consequently, the inconsistency in $B_1 \otimes B_2$ is not implied by P. That is, $B_1 \otimes B_2$ is not an s-model of P.

Let us now investigate what happens when we add to an annotated revision program P a rule $r = (l{:}\alpha) \leftarrow (l{:}\alpha)$ (here l is a revision atom, α is an annotation). Unlike ordinary revision programs where every database is a model of a rule of the form $l \leftarrow l$, not every $\mathcal{T}^2$-valuation is an s-model of r. Therefore, adding such a rule may affect the set of s-models of the program. On the one hand, rule r by imposing additional implicit bound on l^D may give rise to a situation when an s-model of P is not an an s-model of $P \cup \{r\}$ (case (1) of Example 3.7). On the other hand, rule r may provide additional explicit evidence for l that results in a situation when an s-model of $P \cup \{r\}$ is not an s-model of P (case (2) of Example 3.7).

Example 3.7. Let $U = \{a\}$ and the lattice of annotations be $\mathcal{T}_{\{p,q\}}$. Let $B(a) = \langle \{p\}, \{p\} \rangle$. Let $r =$ $(\mathbf{in}(a){:}\{p\}) \leftarrow (\mathbf{in}(a){:}\{p\})$.

(1) Let $P = \{\}$. Then, $T_P^b(B)(a) = \langle \emptyset, \emptyset \rangle$, and $-T_P^b(B)(a) = \langle \{p,q\}, \{p,q\} \rangle$. Hence,

$$T_P^b(B)(a) \leqslant B(a) \leqslant T_P^b(B)(a) \vee \big(-T_P^b(B)\big)(a).$$

Thus, B is an s-model of P. However, B is *not* an s-model of $P \cup \{r\}$. Indeed, $T_{P\cup\{r\}}^b(B)(a) = \langle \{p\}, \emptyset \rangle$, and $-T_{P\cup\{r\}}^b(B)(a) = \langle \{p,q\}, \{q\} \rangle$. Hence,

$$B(a) \nleqslant T_{P\cup\{r\}}^b(B)(a) \vee \big(-T_{P\cup\{r\}}^b(B)\big)(a) = \langle \{p,q\}, \{q\} \rangle.$$

Therefore, B is *not* an s-model of $P \cup \{r\}$.

(2) Let $P = \{(\mathbf{out}(a){:}\{p\}) \leftarrow\}$. Then it is easy to see that B is *not* an s-model of P. However, B is an s-model of $P \cup \{r\}$.

Remark 3.8. Let us note that adding rule $r = (l{:}\alpha) \leftarrow (l{:}\alpha)$ to P has no effect on consistent models of P. Indeed, let B be a consistent model of P. Clearly, B is a model of $\{r\}$. Hence, by Theorem 3.4, B is an s-model of P, and an s-model of $\{r\}$. Therefore, Theorem 3.5 implies that B is an s-model of $P \cup \{r\}$.

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 [10]. The reader is referred to [10,11] 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 are dropped. Next, main properties of justified revisions studied in [9,11] should have their counterparts in the case of justified revisions of annotated programs. In particular, justified revisions of an annotated revision program should be models of the program.

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 $(l{:}\beta_1 \vee \beta_2)$.

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

We will refer to this operation as the *join transformation*.

It is clear, that a $\mathcal{T}^2$-valuation 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 vice 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*.

Now we introduce the notion of a justified revision of an annotated revision program and contrast it with an earlier proposal by Fitting [4]. 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 [10,11], 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 these two valuations. The construction of the reduct consists of two steps and mirrors the original definition of the reduct of an unannotated revision program [11]. 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 $\alpha \vee \gamma \geqslant \beta$. The least such element exists (due to the fact that $\mathcal{T}$ is complete and infinitely distributive). Let us denote this value 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 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.

[2] The operation $pcomp(\cdot, \cdot)$ is known in the lattice theory as the *relative pseudocomplement*, see [14].

Definition 4.1. 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 this change 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 4.2. B_R is a *P-justified revision* of B_I if $B_R = (B_I \otimes -C) \oplus C$, where $C = NC(P_{B_R}|B_I)$ is the necessary change for $P_{B_R}|B_I$.

We will now contrast this approach with the one proposed by Fitting in [4]. In order to do so, we recall the definitions introduced in [4]. 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 4.3 (*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 it uses the necessary change of the F-reduct (instead of the necessary change of the reduct defined above in Definition 4.1). We call the justified revision based on 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 4.4. 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 a 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 4.5. Let P be the same revision program as before, and let P' consist of the rules

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

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 $(\mathbf{in}(b){:}\{p,q\})$ 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 theorem provides a complete characterization of those cases (let us recall that a lattice $\mathcal{T}$ is *linear* if for any two elements $\alpha,\beta \in \mathcal{T}$ either $\alpha \leqslant \beta$ or $\beta \leqslant \alpha$).

Theorem 4.6. *F-justified revisions and justified revisions coincide if and only if the lattice $\mathcal{T}$ is linear.*

Proof. $(\Rightarrow)$ Assume that F-justified revisions and justified revisions coincide for a lattice $\mathcal{T}$. Let $\alpha,\beta \in \mathcal{T}$. We will show that either $\alpha \leqslant \beta$ or $\beta \leqslant \alpha$. Indeed, let P be annotated revision program consisting of the following rules.

$$(\mathbf{in}(a){:}\alpha) \leftarrow (\mathbf{in}(b){:}\alpha \vee \beta) \quad \text{and} \quad (\mathbf{in}(b){:}\beta) \leftarrow .$$

Let B_I be given by $B_I(a) = \langle\bot,\bot\rangle$ and $B_I(b) = \langle\alpha,\bot\rangle$. Let B_R be given by $B_R(a) = \langle\alpha,\bot\rangle$ and $B_R(b) = \langle\alpha \vee \beta,\bot\rangle$. It is easy to see that B_R is a justified revision of B_I (with respect to P). By our assumption, B_R is also an F-justified revision of B_I. There are only two possible cases.

Case 1. $\alpha \vee \beta \leqslant \alpha$. Then, $\beta \leqslant \alpha$.

Case 2. $\alpha \vee \beta \nleqslant \alpha$. Then, $P^F_{B_R}|B_I = P$. Let $C = NC(P^F_{B_R}|B_I)$. By the definition of the necessary change,

$$C(a) = NC\big(P^F_{B_R}|B_I\big)(a) = NC(P)(a) = \begin{cases} \langle\bot,\bot\rangle, & \text{when } \alpha \vee \beta \nleqslant \beta, \\ \langle\alpha,\bot\rangle, & \text{when } \alpha \vee \beta \leqslant \beta. \end{cases}$$

By the definition of an F-justified revision, $B_R = (B_I \otimes -C) \oplus C$. From the facts that $B_R(a) = \langle\alpha,\bot\rangle$ and $B_I(a) = \langle\bot,\bot\rangle$ it follows that $C(a) = \langle\alpha,\bot\rangle$. Therefore, it is the case that $\alpha \vee \beta \leqslant \beta$. That is, $\alpha \leqslant \beta$.

$(\Leftarrow)$ Assume that lattice $\mathcal{T}$ is linear. Then, for any $\alpha,\beta \in \mathcal{T}$

$$pcomp(\alpha,\beta) = \begin{cases} \bot, & \text{when } \alpha \geqslant \beta, \\ \beta, & \text{otherwise (when } \alpha < \beta). \end{cases}$$

Let P be an annotated revision program. Let B_I and B_R be any $\mathcal{T}^2$-valuations. Let us see what is the difference between $P_{B_R}|B_I$ and $P^F_{B_R}|B_I$. The first steps in the definitions of reduct and F-reduct are the same. During the second step of the definition of an F-reduct each annotated atom $(l{:}\beta)$ such that $\beta \leqslant B_I(l)$ is deleted from bodies of rules. In the second step of the definition of the reduct such annotated atom is replaced by $(l{:}\bot)$. If $\beta > B_I(l)$, then in the reduct $P_{B_R}|B_I$ annotated atom $(l{:}\beta)$ is replaced by $(l{:}\mathit{pcomp}(B_I(l), \beta)) = (l{:}\beta)$, that is, it remains as it is. In the F-reduct, $(l{:}\beta)$ also remains in the bodies for $\beta > B_I(l)$. Thus, the only difference between $P_{B_R}|B_I$ and $P^F_{B_R}|B_I$ is that bodies of the rules from $P_{B_R}|B_I$ may contain atoms of the form $(l{:}\bot)$, where $l \in U$, that are not present in the bodies of the corresponding rules in $P^F_{B_R}|B_I$. However, annotated atoms of the form $(l{:}\bot)$ are always satisfied. Therefore, the necessary changes of $P_{B_R}|B_I$ and $P^F_{B_R}|B_I$, as well as justified and F-justified revisions of B_I coincide. $\square$

Theorem 4.6 explains why the difference between the justified revisions and F-justified revisions is not seen when we limit our attention to revision programs as considered in [11]. Namely, the lattice $\mathcal{TWO} = \{\boldsymbol{f}, \boldsymbol{t}\}$ 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 observe that revision programs as defined in [10] can be encoded as annotated revision programs (with annotations taken from the lattice $\mathcal{TWO} = \{\boldsymbol{f}, \boldsymbol{t}\}$). Namely, a revision rule

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

(where p and all q_i's are revision atoms) can be encoded as

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

We will denote by P^a the result of applying this transformation to a revision program P (rule by rule). Second, let us represent a set of atoms B by a $\mathcal{TWO}^2$-valuation B^v as follows: $B^v(a) = \langle \boldsymbol{t}, \boldsymbol{f} \rangle$, if $a \in B$, and $B^v(a) = \langle \boldsymbol{f}, \boldsymbol{t} \rangle$, otherwise.

Fitting [4] argued that under such encodings the semantics of F-justified revisions generalizes the semantics of justified revisions introduced in [10]. 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 [10]. Specifically, we have the following result.

Theorem 5.1. *Let P be an ordinary revision program and let B_I and B_R be two sets of atoms. Then, B_R is a P-justified revision of B_I if and only if the necessary change of $P^a_{B^v_R}|B^v_I$ is consistent and B^v_R is a P^a-justified revision of B^v_I.*

Before we study how properties of justified revisions generalize to the case with annotations, we prove the following auxiliary results.

Lemma 5.2. *Let P be an annotated revision program. Let B be a $\mathcal{T}^2$-valuation. Then, $NC(P_B|B) = T_P^b(B)$.*

Proof. The assertion follows from definitions of a necessary change and operator T_P^b. □

Lemma 5.3. *Let P be an annotated revision program. Let B_I, B_R, and C be $\mathcal{T}^2$-valuations, such that $B_R \leqslant B_I \oplus C$. Then, C satisfies the bodies of all rules in $P_{B_R}|B_I$.*

Proof. Let $r' \in P_{B_R}|B_I$. Let $(l{:}\gamma)$ be an annotated revision atom from the body of r'. Let $(\theta^{-1}(B_I))(l) = \alpha$. By the definition of the reduct, r' was obtained from some rule $r \in P$, such that the body of r is satisfied by B_R, and $\gamma = pcomp(\alpha, \beta)$, where $(l{:}\beta)$ is in the body of r. Since the body of r is satisfied by B_R, we have $\beta \leqslant (\theta^{-1}(B_R))(l)$. From $B_R \leqslant_k B_I \oplus C$ it follows that

$$
\begin{aligned}
\big(\theta^{-1}(B_R)\big)(l) &\leqslant \big(\theta^{-1}(B_I \oplus C)\big)(l) \\
&= \big(\theta^{-1}(B_I)\big)(l) \vee \big(\theta^{-1}(C)\big)(l) \\
&= \alpha \vee \big(\theta^{-1}(C)\big)(l).
\end{aligned}
$$

Combining this inequality with our previous observation that $\beta \leqslant (\theta^{-1}(B_R))(l)$, we get $\beta \leqslant \alpha \vee (\theta^{-1}(C))(l)$. By the definition of $pcomp(\alpha, \beta)$, we get $\gamma \leqslant (\theta^{-1}(C))(l)$. That is, C satisfies $(l{:}\gamma)$. Since $(l{:}\gamma)$ was arbitrary, C satisfies all annotated revision atoms in the body of r'. As r' was an arbitrary rule from $P_{B_R}|B_I$, we conclude that C satisfies the bodies of all rules in $P_{B_R}|B_I$. □

Lemma 5.4. *Let B_R be a P-justified revision of B_I. Then, $NC(P_{B_R}|B_I) = T_P^b(B_R)$.*

Proof. By the definition of a justified revision $B_R = (B_I \otimes -C) \oplus C$, where $C = NC(P_{B_R}|B_I)$. Hence, $B_R \leqslant B_I \oplus C$. By Lemma 5.3, C satisfies the bodies of all rules in $P_{B_R}|B_I$. Since C is a model of $P_{B_R}|B_I$, C satisfies all heads of clauses in $P_{B_R}|B_I$.

Let D be a valuation satisfying all heads of rules in $P_{B_R}|B_I$. Then D is a model of $P_{B_R}|B_I$. Since C is the least model of the reduct $P_{B_R}|B_I$, we find that $C \leqslant_k D$. Consequently, C is the least valuation that satisfies all heads of the rules in $P_{B_R}|B_I$. The rules in P_{B_R} are all those rules from P whose bodies are satisfied by B_R. Thus, by the definition of the operator T_P^b, $C = T_P^b(B_R)$. □

We will now look at properties of the semantics of justified revisions. We will present a series of results generalizing properties of revision programs to the case with annotations. We will show that the concept of an s-model is a useful notion in the investigations of justified revisions of annotated programs.

Our first result relates justified revisions to models and s-models. 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 an analogous result.

Theorem 5.5. *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 an s-model of P (and, hence, a model of P).*

Proof. By the definition of a P-justified revision, $B_R = (B_I \otimes -C) \oplus C$, where C is the necessary change for $P_{B_R}|B_I$. From Lemma 5.4 it follows that $C = T_P^b(B_R)$. Therefore,

$$B_R = \big(B_I \otimes -T_P^b(B_R)\big) \oplus T_P^b(B_R) \leqslant_k -T_P^b(B_R) \oplus T_P^b(B_R).$$

Also,

$$B_R = \big(B_I \otimes -T_P^b(B_R)\big) \oplus T_P^b(B_R) \geqslant T_P^b(B_R).$$

Hence, B_R is an s-model of P. □

In the previous section we showed an example demonstrating that F-justified revisions do not satisfy the property of invariance under joins. In contrast, justified revisions in the sense of our paper do have this property.

Theorem 5.6. *Let P_2 be the result of simplification of an annotated revision program P_1 by means of the join transformation. Then for every initial database B_I, P_1-justified revisions of B_I coincide with P_2-justified revisions of B_I.*

The proof follows directly from the definition of P-justified revisions and from the following distributivity property of pseudocomplement: $pcomp(\alpha, \beta_1) \vee pcomp(\alpha, \beta_2) = pcomp(\alpha, \beta_1 \vee \beta_2)$.

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 several results provide a complete description of justified revisions of models of annotated revision programs. First, we characterize those models that are their own justified revisions. This result provides additional support for the importance of the notion of an s-model in the study of annotated revision programs.

Theorem 5.7. *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 an s-model of P.*

Proof. Let us denote $C = NC(P_{B_I}|B_I)$. By the definition, B_I is a P-justified revision of itself if and only if $B_I = (B_I \otimes -C) \oplus C$. Since B_I satisfies P, Theorem 3.1 and Lemma 5.2 imply that $B_I \geqslant_k C$. Thus, $B_I \oplus C = B_I$. Distributivity of the product lattice $\mathcal{T}^2$ implies that

$$(B_I \otimes -C) \oplus C = (B_I \oplus C) \otimes (-C \oplus C) = B_I \otimes (-C \oplus C).$$

Clearly, $B_I = B_I \otimes (-C \oplus C)$ if and only if $B_I \leqslant_k (-C \oplus C)$.

By Lemma 5.2, $C = NC(P_{B_I}|B_I) = T_P^b(B_I)$. Thus, B_I is a P-justified revision of itself if and only if $B_I \leqslant_k T_P^b(B_I) \oplus (-T_P^b(B_I))$. But this latter condition is precisely the one that distinguishes s-models among models. Thus, under the assumptions of the theorem, B_I is a P-justified revision of itself if and only if it is an s-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. In other words, s-models, if they are inconsistent, may have other revisions besides themselves (by Theorem 5.7 they always are their own revisions).

The following example shows that an *inconsistent* s-model may have no revisions other than itself, may have only one consistent justified revision, or may have incomparable (with respect to the knowledge ordering) consistent revisions.

Example 5.8. Let the lattice of annotations be $\mathcal{T}_{\{p,q\}}$. Consider an inconsistent $\mathcal{T}^2$-valuation B_I such that $B_I(a) = \langle\{q\},\{q\}\rangle$.

(1) Consider annotated revision program P_1 consisting of the clauses:

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

It is easy to see that B_I is an s-model of P_1 and the only justified revision of itself.

(2) Let an annotated revision program P_2 consist of the clauses:

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

Clearly, B_I is an s-model of P_2. Hence, B_I is its own justified revision (under P_2). However, B_I is not the only P_2-justified revision of B_I. Consider the $\mathcal{T}^2$-valuation B_R such that $B_R(a) = \langle\emptyset,\{q\}\rangle$. We have $P_{2B_R}|B_I = \{(\mathbf{out}(a){:}\{q\}) \leftarrow\}$. Let us denote the corresponding necessary change, $NC(P_{2B_R}|B_I)$, by C. Then, $C(a) = \langle\emptyset,\{q\}\rangle$. Hence, $-C = \langle\{p\},\{p,q\}\rangle$ and $((B_I \otimes -C) \oplus C)(a) = \langle\emptyset,\{q\}\rangle = B_R(a)$. Consequently, B_R is a P_2-justified revision of B_I. It is the only consistent P_2-justified revision of B_I.

(3) Let an annotated revision program P_3 be the following:

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

Then, B_I is an s-model of P_3 and its own P_3-justified revision. In addition, it is straightforward to check that B_I has two consistent revisions B_R and B'_R, where $B_R(a) = \langle\emptyset,\{q\}\rangle$ and $B'_R(a) = \langle\{q\},\emptyset\rangle$. The revisions B_R and B'_R are incomparable with respect to the knowledge ordering.

The same behavior can be observed in the case of programs annotated with elements from other lattices. The following example is analogous to the second case in Example 5.8, but the lattice is $\mathcal{T}_{[0,1]}$.

Example 5.9. 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 an s-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 $-C = \langle 0, 1\rangle$. Thus, $((B_I \otimes -C) \oplus C)(a) = \langle 0, 1\rangle = B_R(a)$. 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 $\leqslant_k$. It is not coincidental as demonstrated by our next result.

Theorem 5.10. *Let B_I be a model of an annotated revision program P. Let B_R be a P-justified revision of B_I. Then, $B_R \leqslant_k B_I$.*

Proof. By the definition of a P-justified revision, $B_R = (B_I \otimes -C) \oplus C$, where C is the necessary change of $P_{B_R}|B_I$. By the definition of the reduct $P_{B_R}|B_I$ and the fact that B_I is a model of P, it follows that B_I is a model of $P_{B_R}|B_I$. The necessary change C is the least fixpoint of $T^b_{P_{B_R}|B_I}$, therefore, $C \leqslant B_I$. Hence,

$$B_R = (B_I \otimes -C) \oplus C \leqslant_k B_I \oplus C \leqslant_k B_I \oplus B_I = B_I. \qquad \square$$

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

Theorem 5.11. *Let B_I be a consistent model of an annotated revision program P. Then, B_I is the only P-justified revision of itself.*

Proof. Theorem 3.4 implies that B_I is an s-model of P. Then, from Theorem 5.7 we get that B_I is a P-justified revision of itself. We need to show that there are no other P-justified revisions of B_I.

Let B_R be a P-justified revision of B_I. Then, $B_R \leqslant_k B_I$ (Theorem 5.10). Therefore, $T^b_P(B_R) \leqslant_k T^b_P(B_I)$. Hence, $-T^b_P(B_I) \leqslant_k -T^b_P(B_R)$. Theorem 3.1 implies that $B_I \geqslant_k T^b_P(B_I)$. Thus, $-B_I \leqslant_k -T^b_P(B_I)$. Since B_I is consistent, $B_I \leqslant_k -B_I$. Combining the above inequalities, we get

$$B_I \leqslant_k -B_I \leqslant_k -T^b_P(B_I) \leqslant_k -T^b_P(B_R).$$

That is, $B_I \leqslant_k -T^b_P(B_R)$. Hence, $B_I \otimes -T^b_P(B_R) = B_I$.

From definition of justified revision and Lemma 5.4,

$$B_R = \bigl(B_I \otimes -T^b_P(B_R)\bigr) \oplus T^b_P(B_R) = B_I \oplus T^b_P(B_R) \geqslant_k B_I.$$

Therefore, $B_R = B_I$. $\square$

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 an s-model of a program, may have different justified revisions. However, all these additional revisions must be $\leqslant_k$-less inconsistent 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.

In [11] we proved that, in the case of ordinary revision programs, "additional evidence does not destroy justified revisions". More precisely, we proved that if B_R is a P-justified revision of B_I and B_R is a model of P' then B_R is a $P \cup P'$-justified revision of B_I. We will now prove a generalization of this property to the case of annotated revision programs. However, as before, we need to replace the notion of a model with that of an s-model.

Theorem 5.12. *Let P, P' be annotated revision programs. Let B_R be a P-justified revision of B_I. Let B_R be an s-model of P'. Then, B_R is a $P \cup P'$-justified revision of B_I.*

Proof. Let $C = NC(P_{B_R}|B_I)$. Let $C' = NC((P \cup P')_{B_R}|B_I)$. Clearly, $C \leqslant_k C'$. By the definition of a justified revision $B_R = (B_I \otimes -C) \oplus C$. Hence,

$$B_R \leqslant_k B_I \oplus C \leqslant_k B_I \oplus C'.$$

By Lemma 5.3 it follows that C' satisfies the bodies of all rules in $(P \cup P')_{B_R}|B_I$. Since C' is the necessary change of $(P \cup P')_{B_R}|B_I$ we conclude that C' satisfies the heads of all rules in $(P \cup P')_{B_R}|B_I$. Reasoning as in the proof of Lemma 5.4 we find that $C' = T^b_{P \cup P'}(B_R)$.

By Theorem 5.5, B_R is an s-model of P. Therefore, by Theorem 3.5, B_R is a s-model of $P \cup P'$. Theorem 5.7 implies that B_R is a $P \cup P'$-justified revision of itself. In other words,

$$B_R = \big(B_R \otimes -NC\big((P \cup P')_{B_R}|B_R\big)\big) \oplus NC\big((P \cup P')_{B_R}|B_R\big).$$

From Lemma 5.2 it follows that $NC((P \cup P')_{B_R}|B_R) = T^b_{P \cup P'}(B_R)$. Hence,

$$B_R = (B_R \otimes -C') \oplus C'.$$

Next, let us recall that $B_R = (B_I \otimes -C) \oplus C$. Hence,

$$B_R = \big(\big((B_I \otimes -C) \oplus C\big) \otimes -C'\big) \oplus C'.$$

Now, using the facts that $C \leqslant_k C'$ and $-C' \leqslant_k -C$, we get the following equalities:

$$\begin{aligned} B_R &= \big(\big((B_I \otimes -C) \oplus C\big) \otimes -C'\big) \oplus C' \\ &= \big((B_I \otimes -C) \otimes -C'\big) \oplus (C \otimes -C') \oplus C' \\ &= \big(B_I \otimes (-C \otimes -C')\big) \oplus C' = (B_I \otimes -C') \oplus C'. \end{aligned}$$

Thus, $B_R = (B_I \otimes -C') \oplus C'$. By the definition of justified revisions, B_R is a $P \cup P'$-justified revision of B_I. □

In case of revision programs without annotations, justified revisions satisfy the minimality principle (see [11]). Namely, P-justified revisions of a database differ from the database by as little as possible. Recall, that in the case of revision programs without annotations, databases are sets of atoms, and the difference between databases R and I is their symmetric difference $R \div I = (R \setminus I) \cup (I \setminus R)$. The minimality principle states that if R is a P-justified revision of I, then, $R \div I$ is minimal in the family $\{B \div I : B$ is a model of $P\}$ (Theorem 3.6 in [11]).

Before generalizing the minimality principle to the case of annotated revision programs we need to specify what we mean by the difference between $\mathcal{T}^2$-valuations.

Definition 5.13. Let R, B be $\mathcal{T}^2$-valuations. We say that B *can be transformed into* R *via* a $\mathcal{T}^2$-valuation C if $R = (B \otimes -C) \oplus C$. We say that B *can be transformed* into R if there exists $\mathcal{T}^2$-valuation C such that B can be transformed into R via C.

Given two $\mathcal{T}^2$-valuations, it is not necessarily the case that one of them can be transformed into the other. Indeed, let $V_\top$ be a $\mathcal{T}^2$-valuation that assigns to each atom annotation $\top$. Let $V_\perp$ be a $\mathcal{T}^2$-valuation that assigns to each atom annotation $\perp$. Then, if a lattice consists of more than one element, then we have $\top \neq \perp$, and $V_\top$ cannot be transformed into $V_\perp$.

Definition 5.14. Let R, B be $\mathcal{T}^2$-valuations. Let $S = \{C \mid B$ can be transformed into R via $C\}$. The *difference* $\mathrm{diff}(R, B)$ is

$$\mathrm{diff}(R, B) = \begin{cases} \prod S, & \text{when } S \neq \emptyset, \\ V_\top, & \text{otherwise (when } S = \emptyset). \end{cases}$$

The following lemma describes a useful property of a difference between $\mathcal{T}^2$-valuations. Namely, the difference between $\mathcal{T}^2$-valuations R and B is the least (in $\leqslant_k$ ordering) $\mathcal{T}^2$-valuation among all C such that $R = (B \otimes -C) \oplus C$.

Lemma 5.15. *Let R, B be $\mathcal{T}^2$-valuations. Let $S = \{C \mid B$ can be transformed into R via $C\}$. If $S \neq \emptyset$, then* $\mathrm{diff}(R, B) \in S$.

Proof. Let $S = \{C \mid B$ can be transformed into R via $C\} \neq \emptyset$. Then, $\mathrm{diff}(R, B) = \prod S$. First, let us show that $-\prod S = \sum\{-C\colon C \in S\}$. On the one hand, $\prod S \leqslant_k C$ for all $C \in S$. Thus, $-\prod S \geqslant_k -C$ for all $C \in S$. Hence,

$$-\prod S \geqslant_k \sum\{-C\colon C \in S\}. \tag{8}$$

On the other hand, $\sum\{-C\colon C \in S\} \geqslant_k -C$ for all $C \in S$. Thus, $-\sum\{-C\colon C \in S\} \leqslant_k C$ for all $C \in S$. Hence, $-\sum\{-C\colon C \in S\} \leqslant_k \prod S$. That is,

$$\sum\{-C\colon C \in S\} \geqslant_k -\prod S. \tag{9}$$

From (8) and (9) it follows that $-\prod S = \sum\{-C\colon C \in S\}$.

Since $\mathcal{T}$ is complete and infinitely distributive, we get the following.

$$\begin{aligned} \Big(B \otimes -\prod S\Big) \oplus \prod S &= \Big(B \otimes \sum\{-C\colon C \in S\}\Big) \oplus \prod S \\ &= \sum\{(B \otimes -C)\colon C \in S\} \oplus \prod S \\ &= \prod\Big\{\sum\{(B \otimes -C)\colon C \in S\} \oplus C'\colon C' \in S\Big\} \\ &\geqslant_k \prod\{(B \otimes -C') \oplus C'\colon C' \in S\} = \prod\{R\} = R. \end{aligned}$$

That is,

$$\Big(B \otimes -\prod S\Big) \oplus \prod S \geqslant_k R. \tag{10}$$

By definition of S, for each $C \in S$, $R = (B \otimes -C) \oplus C$. Therefore, for each $C \in S$, $C \leqslant_k R$ and $B \otimes -C \leqslant_k R$. Thus, $\prod S \leqslant_k R$ and

$$B \otimes -\prod S = B \otimes \sum\{-C\colon C \in S\} = \sum\{(B \otimes -C)\colon C \in S\} \leqslant_k R.$$

Hence, $(B \otimes -\prod S) \oplus \prod S \leqslant_k R$. This together with (10) imply that

$$\Big(B \otimes -\prod S\Big) \oplus \prod S = R.$$

That is, $\prod S \in S$. $\square$

Now we will show that the minimality principle can be generalized to the case of annotated revision programs. We will have, however, to assume that $\mathcal{T}$ is a Boolean algebra and restrict ourselves to consistent $\mathcal{T}^2$-valuations.

Let $\mathcal{T}$ be a Boolean algebra with De Morgan complement being the complement. Let us define the *negation* operation on $\mathcal{T}^2$ as $\neg\langle\alpha,\beta\rangle = \langle\bar{\alpha},\bar{\beta}\rangle$ $(\alpha,\beta \in \mathcal{T})$. Then, the lattice $\mathcal{T}^2$ with operations $\oplus$, $\otimes$, $\neg$, and elements $\bot$, $\top$ is a Boolean algebra, too. Operations on $\mathcal{T}^2$ lift pointwise to the space of $\mathcal{T}^2$-valuations. It is easy to see that the space of $\mathcal{T}^2$-valuations with operations $\oplus$, $\otimes$, $\neg$, and elements $V_\bot$, $V_\top$ is again a Boolean algebra.

Lemma 5.16. *Let $\mathcal{T}$ be a Boolean algebra. Let R, B, I be $\mathcal{T}^2$-valuations. Let R and I be consistent. Let* $\mathrm{diff}(R,B) \leqslant_k \mathrm{diff}(R,I)$. *Then, $R \otimes B \geqslant_k R \otimes I$.*

Proof. Let $C = \mathrm{diff}(R,I)$, $C' = \mathrm{diff}(R,B)$. Since I is consistent, $I \leqslant_k -I$. Thus,

$$I \otimes -(\neg I) \leqslant_k -I \otimes -(\neg I) = -(I \oplus \neg I) = -V_\top = V_\bot. \tag{11}$$

Since R is consistent, C is consistent, too. That is, $C \leqslant_k -C$. Hence,

$$I \otimes -C = (I \otimes -C) \oplus (I \otimes C). \tag{12}$$

Consider valuation $C \otimes \neg I$. Using (11) and (12) we get:

$$\begin{aligned}
\big(I \otimes -(C \otimes \neg I)\big) \oplus (C \otimes \neg I) &= \big(I \otimes \big(-C \oplus -(\neg I)\big)\big) \oplus (C \otimes \neg I)\\
&= (I \otimes -C) \oplus \big(I \otimes -(\neg I)\big) \oplus (C \otimes \neg I)\\
&= (I \otimes -C) \oplus (I \otimes C) \oplus V_\bot \oplus (C \otimes \neg I)\\
&= (I \otimes -C) \oplus (I \otimes C) \oplus (C \otimes \neg I)\\
&= (I \otimes -C) \oplus \big(C \otimes (I \oplus \neg I)\big)\\
&= (I \otimes -C) \oplus (C \otimes V_\top) = (I \otimes -C) \oplus C = R.
\end{aligned}$$

Consequently, $C \leqslant_k C \otimes \neg I$ (by definition of $\mathrm{diff}(R,I)$). Hence, $C \otimes I \leqslant_k C \otimes \neg I \otimes I = V_\bot$. That is, $C \otimes I = V_\bot$. Since $C' \leqslant_k C$, it follows that $C' \otimes I = V_\bot$. We have:

$$I \otimes -C \leqslant_k R = (B \otimes -C') \oplus C'.$$

Thus,

$$\begin{aligned} I \otimes -C &= (I \otimes -C) \otimes I \leqslant_k \big((B \otimes -C') \oplus C'\big) \otimes I \\ &= \big((B \otimes -C') \otimes I\big) \oplus (C' \otimes I) \\ &= \big((B \otimes -C') \otimes I\big) \oplus V_\perp = (B \otimes -C') \otimes I \leqslant_k B \otimes -C'. \end{aligned}$$

That is,

$$I \otimes -C \leqslant_k B \otimes -C'. \tag{13}$$

Since R is consistent, C' is consistent, too. It means that $C' \leqslant_k -C'$. Hence, $B \otimes -C' \geqslant_k B \otimes C'$. Therefore,

$$\begin{aligned} R \otimes B &= \big((B \otimes -C') \oplus C'\big) \otimes B = \big((B \otimes -C') \otimes B\big) \oplus (C' \otimes B) \\ &= (B \otimes -C') \oplus (B \otimes C') = B \otimes -C'. \end{aligned}$$

That is,

$$R \otimes B = B \otimes -C'. \tag{14}$$

Similarly,

$$R \otimes I = I \otimes -C. \tag{15}$$

Combining (13), (14), and (15) we get $R \otimes I \leqslant_k R \otimes B$. □

If $\mathcal{T}$ is *not* a Boolean algebra, then the statement of the above lemma does not necessarily hold, as illustrated by the following example.

Example 5.17. Let $\mathcal{T} = \mathcal{T}_{[0,1]}$, $U = \{a\}$. Let $R(a) = \langle 0.3, 0.7\rangle$, $B(a) = \langle 0.2, 0.5\rangle$, and $I(a) = \langle 0.1, 0.6\rangle$. Clearly, R and I are consistent. It is easy to see that $(\mathrm{diff}(R, B))(a) = (\mathrm{diff}(R, I))(a) = \langle 0.3, 0.7\rangle$. Hence, $\mathrm{diff}(R, B) \leqslant_k \mathrm{diff}(R, I)$. However, $R \otimes B \ngeqslant_k R \otimes I$. Indeed, $(R \otimes B)(a) = \langle 0.2, 0.5\rangle$, and $(R \otimes I)(a) = \langle 0.1, 0.6\rangle$.

Theorem 5.18. *Let $\mathcal{T}$ be a Boolean algebra. Let R be a consistent P-justified revision of a consistent I. Let $C = \mathrm{diff}(R, I)$. Let B be such that $\mathrm{diff}(R, B) = C' \leqslant_k C$. Then, R is a P-justified revision of B.*

Proof. Consider two reducts $P_R|I$ and $P_R|B$. Let $r' \in P_R$. Let $(l{:}\beta)$ be an annotated revision atom from the body of r'. Let $(\theta^{-1}(I))(l) = \delta_I$, $(\theta^{-1}(B))(l) = \delta_B$, and $(\theta^{-1}(R))(l) = \delta_R$. By the definition of a reduct, the corresponding rule in $P_R|I$ contains in the body the annotated revision literal $(l{:}\gamma_I)$, where $\gamma_I = \mathit{pcomp}(\delta_I, \beta)$. The corresponding rule in $P_R|B$ contains in the body the annotated revision literal $(l{:}\gamma_B)$, where $\gamma_B = \mathit{pcomp}(\delta_B, \beta)$. By the definition of pseudocomplement,

$$\delta_I \vee \gamma_I \geqslant \beta. \tag{16}$$

Since $r' \in P_R$, $\beta \leqslant \delta_R$. Hence, $\beta \wedge \delta_R = \beta$. Also, from the definition of *pcomp* we get $\gamma_I \leqslant \beta$, which implies $\gamma_I \wedge \delta_R = \gamma_I$. From (16) we get

$$(\delta_I \vee \gamma_I) \wedge \delta_R \geqslant \beta \wedge \delta_R.$$

That is,

$$(\delta_I \wedge \delta_R) \vee \gamma_I \geqslant \beta.$$

From Lemma 5.16 it follows that $\delta_B \wedge \delta_R \geqslant \delta_I \wedge \delta_R$. Therefore,

$$\delta_B \vee \gamma_I \geqslant (\delta_B \wedge \delta_R) \vee \gamma_I \geqslant \beta.$$

From definition of $pcomp(\delta_B, \beta)$ it follows that $\gamma_B \leqslant \gamma_I$. This means that the only difference between reducts $P_R|I$ and $P_R|B$ is that annotations of literals in the bodies of rules from $P_R|B$ are less than annotations of corresponding literals in $P_R|I$. Consequently, $NC(P_R|B) \geqslant_k NC(P_R|I)$.

Since R is consistent,

$$\begin{aligned} C' &\leqslant_k C \leqslant_k NC(P_R|I) \leqslant_k NC(P_R|B) \leqslant_k R \\ &\leqslant_k -R \leqslant_k -NC(P_R|B) \leqslant_k -C \leqslant_k -C'. \end{aligned}$$

Also, $R = (B \otimes -C') \oplus C'$ implies that $B \otimes -C' \leqslant_k R$, and $B \oplus C' \geqslant_k R$. Then, on one hand,

$$\big(B \otimes -NC(P_R|B)\big) \oplus NC(P_R|B) \leqslant_k (B \otimes -C') \oplus R \leqslant_k R \oplus R = R.$$

On the other hand,

$$\begin{aligned} \big(B \otimes -NC(P_R|B)\big) \oplus NC(P_R|B) &= \big(B \oplus NC(P_R|B)\big) \otimes -NC(P_R|B) \\ &\geqslant_k (B \oplus C') \otimes R \geqslant_k R \otimes R = R. \end{aligned}$$

Therefore, $(B \otimes -NC(P_R|B)) \oplus NC(P_R|B) = R$. That is, R is a P-justified revision of B. $\square$

Theorem 5.19. *Let $\mathcal{T}$ be a Boolean algebra. Let R be a consistent P-justified revision of a consistent I. Then, $\mathrm{diff}(R, I)$ is minimal in the family $\{\mathrm{diff}(B, I)$: B is a consistent model of $P\}$.*

Proof. Let $C = \mathrm{diff}(R, I)$. Then, $R = (I \otimes -C) \oplus C$. Since R is consistent, C is also consistent. That is, $C \leqslant_k -C$. Let B be a consistent model of P, and let $\mathrm{diff}(B, I) = C' \leqslant_k C$. We have $B = (I \otimes -C') \oplus C'$. Inequality $C' \leqslant_k C$ implies $C' \leqslant_k C \leqslant_k -C \leqslant_k -C'$. Therefore,

$$\begin{aligned} (B \otimes -C) \oplus C &= \Big(\big((I \otimes -C') \oplus C'\big) \otimes -C\Big) \oplus C \\ &= (I \otimes -C' \otimes -C) \oplus (C' \otimes -C) \oplus C \\ &= (I \otimes -C) \oplus C' \oplus C \\ &= (I \otimes -C) \oplus C = R. \end{aligned}$$

Consequently, $\mathrm{diff}(R, B) \leqslant_k C$. By Theorem 5.18, R is a P-justified revision of B. However, B is a consistent model of P. By Theorem 5.11, B is the only P-justified revision of itself. Therefore, $R = B$. $\square$

The condition in the above theorem that revision is consistent is important. For inconsistent revisions the minimality principle does not hold, as shown in the following example.

Example 5.20. Let $\mathcal{T} = \mathcal{T}_{\{p\}}$ with the De Morgan complement being the set-theoretic complement. Let P be an annotated revision program consisting of the following rules:

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

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

Let $I(a) = \langle \emptyset, \{p\}\rangle$. Clearly, I is consistent. Let $R_1(a) = \langle \{p\}, \{p\}\rangle$ and $R_2(a) = \langle \{p\}, \emptyset\rangle$. Both R_1 and R_2 are P-justified revisions of I. Thus, R_1 is inconsistent s-model of P, and R_2 is consistent model of P. We have: $\mathrm{diff}(R_1, I) = \langle \{p\}, \{p\}\rangle$, and $\mathrm{diff}(R_2, I) = \langle \{p\}, \emptyset\rangle$. Clearly, $\mathrm{diff}(R_2, I) \leqslant_k \mathrm{diff}(R_1, I)$. Therefore, R_1 is a P-justified revision of a consistent I, but $\mathrm{diff}(R_1, I)$ is *not* minimal in the family $\{\mathrm{diff}(B, I){:}\ B$ is a consistent model of $P\}$.

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 in $\mathcal{T}$ we will evaluate atoms in $\mathcal{T}^2$. 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 "shifting theorem" of [9].

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 the conjunction of $(\mathbf{in}(a){:}\alpha)$ 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 $\langle \alpha, \beta\rangle \leqslant_k B(a)$. 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 programs. Let P be an annotated program. Let B_I and B_R be two $\mathcal{T}^2$-valuations. The reduct of a program P with respect to two valuations B_I and B_R is defined in a manner similar to Definition 4.1. 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!).

Definition 6.1. 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 $\gamma = pcomp(B_I(l), \beta)$ (here $\beta, \gamma \in \mathcal{T}^2$).

Next, we compute the least fixpoint of the operator associated with the reduced program. Finally, as in Definition 4.2, we define the concept of justified revision of a valuation B_I with respect to a revision program P.

Definition 6.2. B_R is a *P-justified revision* of B_I if $B_R = (B_I \otimes -C) \oplus C$, where $C = NC(P_{B_R}|B_I)$ is the necessary change for $P_{B_R}|B_I$.

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

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

a "new" rule

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

An "old" out-rule

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

is encoded in analogous way:

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

Translation tr_2, in the other direction, replaces a "new" revision 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)

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

and

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

The translations tr_1 and tr_2 can be extended to programs. We then have the following theorem that states that the new syntax and semantics of annotated revision programs presented in this section are equivalent to the syntax and semantics introduced and studied earlier in the paper.

Theorem 6.3. *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 [9] 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 $\mathcal{T}^2$. For instance, the mapping $\psi : \mathcal{T}^2 \to \mathcal{T}^2$ defined by $\psi(\langle\alpha, \beta\rangle) = \langle\beta, \alpha\rangle$ is an order isomorphism of $\mathcal{T}^2$. In the case of the lattice $\mathcal{T}_X$, order isomorphisms of $\mathcal{T}_X^2$ can also be generated by permutations of the set X.

Let ψ be an order isomorphism on $\mathcal{T}^2$. It can be extended to annotated atoms, annotated rules, and $\mathcal{T}^2$-valuations as follows:

$$\psi(a{:}\delta) = a{:}\psi(\delta),$$
$$\psi(a{:}\delta \leftarrow a_1{:}\delta_1, \ldots, a_n{:}\delta_n) = \psi(a{:}\delta) \leftarrow \psi(a_1{:}\delta_1), \ldots, \psi(a_n{:}\delta_n),$$
$$\big(\psi(B)\big)(a) = \psi\big(B(a)\big),$$

where $a, a_1, \ldots, a_n \in U$, $\delta, \delta_1, \ldots, \delta_n \in \mathcal{T}^2$, and B is a $\mathcal{T}^2$-valuation.

The extension of an order isomorphism on $\mathcal{T}^2$ to $\mathcal{T}^2$-valuations is again an order isomorphism, this time on the lattice of all $\mathcal{T}^2$-valuations. We say that an order isomorphism ψ on a lattice *preserves conflation* if $\psi(-\delta) = -\psi(\delta)$ for all elements δ from the lattice. We now have the following result that generalizes the shifting theorem of [9].

Theorem 6.4. *Let ψ be an order isomorphism on the set of $\mathcal{T}^2$-valuations. Let ψ preserve conflation. 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 $\psi(B_I)$.*

Proof. By definition, B_R is a P-justified revision of B_I if and only if $B_R = (B_I \otimes -C) \oplus C$, where $C = NC(P_{B_R}|B_I)$. Since ψ is an order isomorphism, it preserves meet and join operations. Therefore,

$$\begin{aligned}\psi(B_R) &= \psi\big((B_I \otimes -C) \oplus C\big) = \psi(B_I \otimes -C) \oplus \psi(C)\\ &= \big(\psi(B_I) \otimes \psi(-C)\big) \oplus \psi(C) = \big(\psi(B_I) \otimes -\psi(C)\big) \oplus \psi(C).\end{aligned}$$

At the same time, $\psi(P_{B_R}|B_I) = (\psi(P))_{\psi(B_R)}|\psi(B_I)$, and $NC(\psi(P_{B_R}|B_I)) = \psi(NC(P_{B_R}|B_I))$. Thus, B_R is a P-justified revision of B_I if and only if $\psi(B_R)$ is a $\psi(P)$-justified revision of $\psi(B_I)$. $\square$

Shifting theorem of [9], that applies to ordinary revision programs, is just a particular case of Theorem 6.4. In order to derive it from Theorem 6.4, we take $\mathcal{T} = \mathcal{TWO}$. Next, we consider an ordinary revision program P and two databases B_1 and B_2 (let us recall that in the case of ordinary revision programs, databases are *sets of atoms* and not valuations).

Let P^a and B_1^v and B_2^v be defined as in Theorem 5.1. It is easy to see that the operator ψ, defined by

$$\big(\psi(v)\big)(a) = \begin{cases} \langle \beta, \alpha\rangle, & \text{when } B_1^v(a) \neq B_2^v(a), \\ \langle \alpha, \beta\rangle, & \text{when } B_1^v(a) = B_2^v(a) \end{cases}$$

is an order-isomorphism on $\mathcal{TWO}^2$-valuations and that $\psi(B_1^v) = B_2^v$. Let C_1 and C_2 be two sets of atoms such that $C_2^v = \psi(C_1^v)$. By Theorem 6.4, C_1^v is a P^a-justified revision of B_1^v if and only if C_2^v is a $\psi(P^a)$-justified revision of B_2^v. Theorem 5.1 and the observation that the necessary change of $P^a_{C_1^v}|B_1^v$ is consistent if and only if the necessary change of $\psi(P^a)_{C_2^v}|B_2^v$ is consistent together imply now the shifting theorem of [9].

The requirement in Theorem 6.4 that ψ preserves conflation is essential. If it is not the case, the statement of the theorem may not hold as illustrated by the following example.

Example 6.5. Let $\mathcal{T} = \mathcal{T}_{\{p,q,r\}}$ with the De Morgan complement defined as follows:

$$\begin{array}{llll} \overline{\{\}} = \{p,q,r\}, & \overline{\{p\}} = \{p,r\}, & \overline{\{q\}} = \{q,r\}, & \overline{\{r\}} = \{p,q\}, \\ \overline{\{p,q,r\}} = \{\}, & \overline{\{p,r\}} = \{p\}, & \overline{\{q,r\}} = \{q\}, & \overline{\{p,q\}} = \{r\}. \end{array}$$

Let ψ be order isomorphism on $\mathcal{T}$ such that $\psi(\{p\}) = \{p\}$, $\psi(\{q\}) = \{r\}$, and $\psi(\{r\}) = \{q\}$. Clearly, ψ does not preserve conflation, because

$$\psi(-\langle\{p\},\{\}\rangle) = \psi(\langle\{p,q,r\},\{p,r\}\rangle) = \langle\{p,q,r\},\{p,q\}\rangle, \quad \text{but}$$
$$-\psi(\langle\{p\},\{\}\rangle) = -\langle\{p\},\{\}\rangle = \langle\{p,q,r\},\{p,r\}\rangle.$$

Let an annotated program be the following:

$$P: \quad a{:}\langle\{p\},\{\}\rangle \leftarrow$$

It determines the necessary change $C(a) = \langle\{p\},\{\}\rangle$.

Then, $-C(a) = \langle\{p,q,r\},\{p,r\}\rangle$. Let $B_I(a) = \langle\{\},\{r\}\rangle$. The P-justified revision of B_I is

$$B_R(a) = (\langle\{\},\{r\}\rangle \otimes \langle\{p,q,r\},\{p,r\}\rangle) \oplus \langle\{p\},\{\}\rangle = \langle\{p\},\{r\}\rangle.$$

The annotated program $\psi(P)$ is the same as P. We have $\psi(B_I)(a) = \langle\{\},\{q\}\rangle$, $\psi(B_R)(a) = \langle\{p\},\{q\}\rangle$. The reduct $(\psi(P))_{\psi(B_R)}|\psi(B_I) = \psi(P) = P$. The necessary change determined by the reduct is C. However,

$$\big(\big(\psi(B_I) \otimes -C\big) \oplus C\big)(a) = \langle\{p\},\{\}\rangle \neq \psi(B_R)(a).$$

Therefore, $\psi(B_R)$ is *not* 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 a justified revision) for annotated programs considered by Fitting in [4]. This new definition eliminates some anomalies arising in the approach by Fitting. Specifically, in Fitting's approach, justified revisions are not, in general, models of a program. In addition, they do not satisfy the invariance-under-join property. In our approach, both properties hold. Moreover, as we show in Sections 5 and 6, many key properties of ordinary revision programs extend to the case of annotated revision programs under our definition of justified revisions.

Several research topics need to be further pursued. First, the concepts of an annotated revision program and of a justified revision can be generalized to the disjunctive case, where a program may have “nonstandard disjunctions” in the head. One can show that this extension indeed reduces back to the ordinary concept of annotated revision programming, as discussed here, if no rule of a program contains a disjunction in its head. However, an in-depth study of annotated disjunctive revision programming has yet to be conducted.

Second, in this paper we focused on the case when the lattice of annotations is distributive. This assumption can be dropped and a reasonable notion of a justified revision can still be defined. However, the corresponding theory is so far less understood and it seems to be much less regular than the one studied in this paper.

Finally, we did not study here the complexity of reasoning tasks for annotated revision programs. Assuming that the lattice is finite and fixed (is not part of the input), the complexity results obtained in [11] can be extended to the annotated case. The complexity of reasoning tasks when the lattice of annotations is a part of an input still needs to be studied. Clearly, any such study would have to take into account the complexity of evaluating lattice operations.

Acknowledgements

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

References

[1] J.J. Alferes, J.A. Leite, L.M. Pereira, H. Przymusinska, T.C. Przymusinski, Dynamic logic programming, in: Proc. 6th International Conference on Principles of Knowledge Representation and Reasoning (KR-98), Trento, Italy, Morgan Kaufmann, San Mateo, CA, 1998, pp. 98–110.

[2] M.H. van Emden, Quantitative deduction and its fixpoint theory, J. Logic Programming 3 (1) (1986) 37–53.

[3] M.H. van Emden, R.A. Kowalski, The semantics of predicate logic as a programming language, J. ACM 23 (4) (1976) 733–742.

[4] M.C. Fitting, Annotated revision specification programs, in: Logic Programming and Nonmonotonic Reasoning (Lexington, KY, 1995), Lecture Notes in Computer Science, Vol. 928, Springer, Berlin, 1995, pp. 143–155.

[5] M.C. Fitting, Fixpoint semantics for logic programming—A survey, Theoret. Comput. Sci. (2002), to appear.

[6] M.L. Ginsberg, Multivalued logics: A uniform approach to reasoning in artificial intelligence, Comput. Intelligence 4 (1988) 265–316.

[7] M. Kifer, V.S. Subrahmanian, Theory of generalized annotated logic programs and its applications, J. Logic Programming 12 (1992) 335–367.

[8] V. Lifschitz, T.Y.C. Woo, Answer sets in general nonmonotonic reasoning, in: Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR-92), Cambridge, MA, Morgan Kaufmann, San Mateo, CA, 1992, pp. 603–614.

[9] W. Marek, I. Pivkina, M. Truszczyński, Revision programming = logic programming + integrity constraints, in: Computer Science Logic, 12th International Workshop, CSL-98, Lecture Notes in Computer Science, Vol. 1584, Springer, Berlin, 1999, pp. 73–89.
[10] W. Marek, M. Truszczyński, Revision programming, database updates and integrity constraints, in: Proc. 5th International Conference on Database Theory—ICDT-95, Lecture Notes in Computer Science, Vol. 893, Springer, Berlin, 1995, pp. 368–382.
[11] W. Marek, M. Truszczyński, Revision programming, Theoret. Comput. Sci. 190 (2) (1998) 241–277.
[12] R. Ng, V.S. Subrahmanian, Stable semantics for probabilistic deductive databases, Inform. and Comput. 110 (1) (1994) 42–83.
[13] T.C. Przymusinski, H. Turner, Update by means of inference rules, J. Logic Programming 30 (2) (1997) 125–143.
[14] H. Rasiowa, R. Sikorski, The Mathematics of Metamathematics, PWN—Polish Scientific Publishers, Warsaw, 1970.
[15] A. Tarski, Logic, Semantics, Metamathematics, Oxford at the Clarendon Press, Oxford, 1956.