Lazy Knowledge Base Update

Witold Łukaszewicz and Ewa Madalińska-Bugaj *
Institute of Informatics, Warsaw University
Warsaw, Poland

Abstract

Knowledge base update has been given much attention in the AI literature. The best-known solution to this problem is Winslett's PMA formalism. In this paper we propose different intuitions standing behind knowledge base update. Roughly speaking, we consider an update formula α not as an effect of an action to be performed, but rather as an observation about dynamically changing world made by an agent. This observation, if it is consistent with the knowledge base, is assumed to be just a new piece of information about the world. Otherwise, it is assumed that a *single* action has been performed and α is its, usually partial, effect.

We formalize our approach using Dijkstra's semantics. We also examine the properties of our update operator with respect to Katsuno and Mendelzon's postulates.

1 Introduction

There are two kinds of modifications of a knowledge base that have been given much attention in the AI literature: *knowledge base revision* and *knowledge base update*.

Knowledge base revision, (KBR, for short) [2, 13] is formulated as follows. We are given a knowledge base KB, representing a static world, and a formula α representing new information. Our task is to determine the resulting knowledge base, $KB * \alpha$. The crucial point is that KB can contain incorrect information, so that α may be inconsistent with KB[1]. The theoretical foundations of KBR have been formulated in [2] by means of *rationality postulates*, referred as the AGM postulates.

Knowledge base update (KBU for short) [14, 15, 11, 9, 3, 10] is similar to KBR with the exception that KB represents dynamic world and α represents the effect of an action to be performed. It is worth noting that under this interpretation KBU is nothing else but a very restricted form of reasoning about action. The formal basis for KBU has been proposed by Katsuno and Mendelzon

*The authors were partially supported by the Wallenberg Foundation and by KBN grant T 11C 009 19.

[1] In this case, it is assumed that α is more reliable than KB.

[11] in the form of eight postulates, referred as the KM postulates. The best-known solution to KBU satisfying the KM postulates is Winslett's PMA [14].

A number of researchers have observed [3, 9] that the KM postulates in the context of classical formulation of KBU can give rise to unintuitive results. The example frequently given is this. Suppose that $KB \models p$ and $\alpha = p \vee q$. Since $KB \models \alpha$, then by postulate 2, we receive $KB * \alpha = p$. This result is clearly problematic. For instance, if α is the effect of tossing a coin, where p is *tails* and q is *heads*, we obviously do not want to conclude that our world does not change after the action.

In recent years, a few KBU formalisms trying to eliminate unintuitive results have been proposed [7]. All of them violate some of the KM postulates.

In this paper we propose different intuitions standing behind knowledge base update. We assume that the knowledge about the world is generally incomplete and that this world can change dynamically as the result of performing actions. Roughly speaking, we consider the update formula α not as an effect of an action to be performed, but rather as an observation about world made by an agent. If this observation is consistent with the knowledge base, there is no reason to conclude that an action has been performed. Accordingly, in this case α is assumed to be just a new piece of information about the world. Otherwise, we conclude that a *single* action has been performed and α is its, usually partial, effect. An example will help to illustrate this.

Suppose I have a neighbour living in Warsaw. Assume first that I have no information about where my neighbour is and where his car is. If I observe the neighbour's car behind his house, I treat this observation as a new piece of information increasing my knowledge about the world. Assume now that I know that yesterday my neighbour went by his car outside. Today, I observed that the car is parked behind his house. The presence of the car clearly suggests that the action of coming back to Warsaw has been performed by the neighbour. It is worth noting that the observation, i.e. the presence of the car, is only a part of the action's effect. Obviously, we should be able to infer more, namely that the neighbour is in Warsaw as well. Note that to make this inference, we must know the full specification of all admissible actions.

The paper is organized as follows. In section 2 we provide a general formalization of the problem and define our update operator referred to as *lazy knowledge base update operator* ($LKBU$, for short). In section 3 we represent $LKBU$ in terms of Dijkstra's semantics. In section 4, we illustrate our update operator with a few examples. Section 5 is devoted to formal properties of $LKBU$ operator. Finally, section 6 contains concluding remarks.

2 Formalization of the problem

We are given a knowledge base KB and an observation α. If α is consistent with KB, we treat it as new information about the world, so the new KB is just the old one augmented with α. Otherwise, we conclude that the world has been changed as the result of performing an action. In this case, our task is the

following: (1) To identify the action that has been performed; (2) To describe the new knowledge base representing the world.

To resolve the above task, we have to know all admissible actions. Suppose that these are $A_1, \ldots, A_n$. Assume further that $\circ$ is an operator that, for a given KB and an action A, returns the new knowledge base representing the world after executing the action[2]. Note that under these assumptions:

1. If the performed action is A_i, the new knowledge base KB' is $KB \circ A_i$.
2. Since α has been observed after executing the action A then KB' should be restricted to $(KB \circ A_i) \wedge \alpha$.
3. If we do not know which action has been performed, all we can conclude is that the new knowledge base is

$$\bigvee_{i=1}^{n} ((KB \circ A_i) \wedge \alpha). \tag{1}$$

It is easy to see that the disjuncts of (1) that are not equivalent to false, specify all possible performed actions.

In conclusion, our update operator, called lazy knowledge base update and abbreviated to *LKBU* is defined by

$$KB * \alpha = \begin{cases} KB \wedge \alpha & \text{if } KB \wedge \alpha \text{ is satisfiable} \\ \bigvee_{i=1}^{n} ((KB \circ A_i) \wedge \alpha) & \text{otherwise} \end{cases}$$

3 *LKBU* in terms of Dijkstra's semantics

In this paper we assume that KB and α are based on classical propositional language, (with boolean constants $\top$ and $\bot$ standing for truth and falsity, respectively) and actions are represented in terms of Dijkstra's semantics, originally developed to reason about programs [6].

In [6] we are provided with a very simple programming language whose semantics is specified in terms of formula transformers. More specifically, with each command S there are associated three formula transformers, called the *weakest precondition*, the *weakest liberal precondition* and the *strongest postcondition*, denoted by *wp*, *wlp* and *sp*, respectively. In this paper we ignore the first two transformers, because they will not be used in the sequel. Before providing the meaning of the *sp* transformer, two remarks are in order.

Firstly, we assume here that the programming language under consideration contains one type of variables only, namely Boolean variables. To be in accord with our earlier terminology, Boolean variables will be referred to as *atoms*. Secondly, in Programming Languages terminology, interpretations are usually referred to as *states*. We shall use this latter term here because it seems to be more appropriate in dynamic settings.

[2]The technical details concerning both the representation of actions and the specification of $\circ$ will be presented later.

If α is a formula, x is a variable and β is a boolean expression, then $\alpha[x \leftarrow \beta]$ denotes a formula obtained from α by replacing all free occurrences of x by β.

Let p be an atom and suppose that α is a formula. By $\exists p.\alpha$ we denote the formula $\alpha[p \leftarrow \top] \vee \alpha[p \leftarrow \bot]$.

The *sp* transformer is to be understood as follows. For each command S and each formula α: $sp(S, \alpha)$ is the formula whose models are precisely all states such that each of them can be reached by starting execution of S in some state satisfying α.

3.1 SDPL: A Simple Dijkstra Programming Language

The considered programming language consists of *skip* command, *assignment* command, *sequential composition* of commands and *alternative* command. Semantics of these commands is specified in terms of *sp* transformer as follows.

1. **The *skip* command.** This is the "empty" command.

   $$sp(skip, \alpha) = \alpha.$$

2. **The *assignment* command.** This command is of the form $x := e$, where x is a (Boolean) variable and e is a (Boolean) expression.

   $$sp(x := e, \alpha) = \exists y.((x \equiv e[x \leftarrow y]) \wedge \alpha[x \leftarrow y]),$$

   where y is a fresh variable. If the variable x does not occur in the expression e, the above equation can be simplified to

   $$sp(x := e, \alpha) = (x \equiv e) \wedge \exists x.\alpha \tag{2}$$

3. **The *sequential composition* command.** This command is of the form $S_1; S_2$, where S_1 and S_2 are any commands.

   $$sp(S_1; S_2, \alpha) \quad = \quad sp(S_2, sp(S_1, \alpha)).$$

4. **The *alternative* command.** This command is of the form

   $$\textbf{if} \quad B_1 \to S_1 \quad [\!] \quad B_2 \to S_2 \quad [\!] \quad \cdots \quad [\!] \quad B_n \to S_n \quad \textbf{fi} \tag{3}$$

   where $B_1, \ldots, B_n$ are Boolean expressions and $S_1, \ldots, S_n$ are any commands. $B_1, \ldots, B_n$ are called *guards* and expressions of the form $B_i \to S_i$ are called *guarded commands*. In the sequel, we refer to the general command (3) as IF. The command is executed as follows. If none of the guards is true, then the execution is equivalent to skip command. Otherwise, one guarded command $B_i \to S_i$ with true B_i is *randomly* selected and S_i is executed.[3]

   $$sp(\text{IF}, \alpha) \quad = \quad \bigvee_{i=1}^{n}(sp(S_i, B_i \wedge \alpha)) \vee (\bigwedge_{i=1}^{n} \neg B_i) \wedge \alpha.$$

[3] Note that when more than one guard is true, the selection of a guarded command is nondeterministic.

In the sequel we shall use the following two properties of the sp transformer: For any command S and any formulae α and β

$$sp(S, \bot) = \bot \tag{4}$$

$$sp(S, \alpha \vee \beta) \equiv sp(S, \alpha) \vee sp(S, \beta). \tag{5}$$

3.2 Computing eliminants

Recall that by $\exists p.\alpha$ we denote the formula $\alpha[p \leftarrow \top] \vee \alpha[p \leftarrow \bot]$. A formula of the form $\exists p.\alpha$ is called an *eliminant of p in α*.[4] Computing eliminants is crucial in computing the sp transformer. In the literature (see [4]), eliminants are computed by first transforming α into its disjunctive normal form. Since transforming a knowledge base into its disjunctive normal form is very expensive, we provide a new algorithm to compute eliminants. This algorithm is based on a DLS algorithm, originally developed to reduce second-order formulae into their first-order equivalents (see [8]). The DLS algorithm is based on Ackermann's lemma [1].

The algorithm consists of three phases.

Phase 1. Preprocessing
The purpose of this phase is to transform the formula $\exists p.A$ into the form

$$\exists p.[(A_1(p) \wedge B_1(p)) \vee \cdots \vee (A_n(p) \wedge B_n(p))] \tag{6}$$

where $A_i(p)$ contains positive and B_i negative ocurrences of the atom p.

1. Eliminate $\Rightarrow$ and $\Leftrightarrow$ and move $\neg$ to the right.

2. Distribute all top-level conjunctions over the disjunctions containing both possitive and negative occurrences of p. For this purpose, apply the equivalences $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ and $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ only if $B \vee C (A \vee B)$ have both positive and negative occurrences of p until the resulting formula is in the form (6).

3. Replace (6) by its equivalent given by

$$\exists p.(A_1(p) \wedge B_1(p)) \vee \cdots \vee \exists p.(A_n(p) \wedge B_n(p)) \tag{7}$$

Phase 2. Preparation to Ackermann's lemma
Repeat this step for each $1 \leq i \leq n$

1. Transform $A_i(p)$ to the form $(p \vee C_i) \wedge D_i$ or $B_i(p)$ to the form $(\neg p \vee C_i) \wedge D_i$, where C_i, D_i do not contain p. This transformation is always possible using thec equivalences $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ and $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.

[4] Intuitively, such an eliminant can be viewed as a formula representing the same knowledge as α about all atoms except p and providing no information about the atom p. The reader interested in a detailed theory of eliminants should consult [4].

Phase 3. Application of Ackermann's lemma
Use Lemma 4.1 (Ackermann's Lemma) from [8].
Repeat this step for each $1 \leq i \leq n$

1. Replace $\exists p(p \vee C_i) \wedge B_i'(p \leftarrow \neg p)$ by $B_i'(p \leftarrow \alpha)$ or $\exists p(\neg p \vee C_i) \wedge A_i'(p)$ by $A_i'(p \leftarrow \alpha)$. ■

4 Examples

In this section we illustrate our update operator with a few examples.

Example 1
There are two fluents, n and c, standing for *my neighbour is in Warsaw* and *his car is in Warsaw*, respectively, and four actions: *leave(Warsaw)-by-car*, *leave(Warsaw)-by-train*, *come-back(to Warsaw)-by-car* and *come-back(to Warsaw)-by train*, specified as follows:

A_1 : *leave-by-car*: **if** $c \wedge n \rightarrow c := \bot; n := \bot [] \neg c \vee \neg n \rightarrow skip$ **fi**

A_2 : *leave-by-train*: **if** $n \rightarrow n := \bot [] \neg n \rightarrow skip$ **fi**

A_3 : *come-back-by-car*: **if** $\neg c \wedge \neg n \rightarrow c := \top; n := \top [] c \vee n \rightarrow skip$ **fi**

A_4 : *come-back-by-train*: **if** $\neg n \rightarrow n := \top [] n \rightarrow skip$ **fi**.

Assume first that $KB{=}\neg n \wedge \neg c$ and our observation α is n. Since $KB{\wedge}\alpha$ is unsatisfiable, $KB{*}\alpha = \bigvee_{i=1}^{4} sp(A_i, KB) \wedge \alpha \equiv [sp(A_3, KB) \wedge \alpha] \vee [sp(A_4, KB) \wedge \alpha]$[5]$\equiv (c \wedge n) \vee (\neg c \wedge n) \equiv n$.

Note that it is impossible to determine how my heighbour came back to Warsaw without an additional observation concerning his car.

Suppose now that KB is as before but α is c. Again $KB{\wedge}\alpha$ is unsatisfiable, so $KB{*}\alpha = \bigvee_{i=1}^{4} sp(A_i, KB) \wedge \alpha \equiv sp(A_3, KB) \wedge \alpha \equiv c \wedge n$.

Note that now we can conclude additional information, namely that my neighbour is in Warsaw. ■

The next example shows that it is sometimes possible to infer new information that is not an effect of a performed action.

Example 2
There are two fluents km, standing for *my son knows undergraduate maths*, and pm, standing for *my son passed undergraduate maths exam*. Suppose that there is one action se (*my son stand to undergraduate maths exam*), defined as

if $km \rightarrow pm := \top [] \neg km \rightarrow pm := \bot$ **fi**.

Assume further that KB is $\neg pm$ and α is pm. Since $KB{\wedge}\alpha$ is unsatisfiable, we infer that the action se has been performed and new KB is $sp(se, KB) \wedge \alpha \equiv (pm \wedge km \vee \neg pm \wedge \neg km) \equiv pm \wedge km$. ■

[5]Remaining disjuncts are reduced to false.

Example 3
There are three fluents denoting a colour of the wall: *blue*, *green*, *red* and three actions: *paint-blue*, *paint-green*, *paint-red* defined as:

paint-blue: $blue := \top; green := \bot; red := \bot$

paint-green: $blue := \bot; green := \top; red := \bot$

paint-red: $blue := \bot; green := \bot; red := \top$

Assume first that $KB = blue \wedge \neg green \wedge \neg red$ and $\alpha = blue \vee green$. Since $KB \wedge \alpha$ is satisfiable, $KB * \alpha = KB \wedge \alpha \equiv KB$.

It is worth noting that the disjunctive formula α is not treated here as an effect of a non-deterministic action but rather as an uncertain observation (possibly due to noisy sensors).

Assume now that KB is as before and α is $green \vee red$. Since $KB \wedge \alpha$ is unsatisfiable, we conclude that an action has been performed.

$$\begin{aligned} KB * \alpha &= sp(\textit{paint-blue}, KB) \wedge (green \vee red) \\ &\vee\ sp(\textit{paint-green}, KB) \wedge (green \vee red) \\ &\vee\ sp(\textit{paint-red}, KB) \wedge (green \vee red) \\ &\equiv \neg blue \wedge (green \wedge \neg red \vee \neg green \wedge red). \end{aligned}$$ ■

5 Formal properties of the *LKBU* operator

As we remarked earlier, Katsuno & Mendelzon [11] proposed a number of postulates (the KM postulates) that they claim every update operator should satisfy. In this section, we check these postulates with respect to *LKBU* operator.

Below, KB is a knowledge base, α is an update formula and $A_1, \ldots, A_n$ is the set of all admissible actions. We assume that this set is *complete* in the following sense: if $KB \wedge \alpha$ is unsatisfiable for satisfiable KB, then there exists at least one action A_i such that $sp(A_i, KB) \wedge \alpha$ is satisfiable. This intuitively means that each observation that is inconsistent with the current world can be explained by at least one action. The KM postulates are the following:

(U1) $KB \star \alpha$ implies α.

(U2) If KB implies α, then $KB \star \alpha$ is equivalent to KB.

(U3) If both KB and α are satisfiable, then $KB \star \alpha$ is also satisfiable.

(U4) If $KB_1 \equiv KB_2$ and $\alpha_1 \equiv \alpha_2$, then $KB_1 \star \alpha_1 \equiv KB_2 \star \alpha_2$.

(U5) $(KB \star \alpha_1) \wedge \alpha_2$ implies $KB \star (\alpha_1 \wedge \alpha_2)$.

(U6) If $KB \star \alpha_1$ implies α_2 and $KB \star \alpha_2$ implies α_1, then $KB \star \alpha_1 \equiv KB \star \alpha_2$.

(U7) If KB is complete, i.e. has at most one model, then $(KB \star \alpha_1) \wedge (KB \star \alpha_2)$ implies $KB \star (\alpha_1 \vee \alpha_2)$.

(U8) $(KB_1 \vee KB_2) \star \alpha \equiv (KB_1 \star \alpha) \vee (KB_2 \star \alpha)$.

Theorem 1
The *LKBU* operator satisfies the postulates (U1) – (U7).
Proof: We start with the following

Lemma 1
If KB is unsatisfiable, then $KB * \alpha$ is equivalent to $\bot$, for any α.
Proof of the lemma: Follows immediately from (4). ■

In view of Lemma 1, it is obvious that the postulates (U1) – (U7) hold if KB is unsatisfiable. We can, therefore, assume that KB is satisfiable.

(U1) and (U4) are straightforward.

(U2) Since KB implies α, KB and $KB \wedge \alpha$ are equivalent, and thus, since KB is satisfiable, $KB \wedge \alpha$ is satisfiable. In conclusion, $KB * \alpha$ is $KB \wedge \alpha$, and so, $KB * \alpha$ is equivalent to KB.

(U3) If $KB \wedge \alpha$ is satisfiable, the postulate obviously holds. Assume therefore, that $KB \wedge \alpha$ is unsatisfiable. Then $KB * \alpha = \bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha$, and so, $KB * \alpha_1$ is satisfiable by the assumption that the set $A_1, \ldots, A_n$ is complete.

(U5) If $KB \wedge \alpha_1$ is satisfiable, the postulate clearly holds. Assume, therefore, that $KB \wedge \alpha_1$ is unsatisfiable. Then $KB * \alpha_1 = \bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_1$, and so, $(KB * \alpha_1) \wedge \alpha_2 = [\bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_1] \wedge \alpha_2 = \bigvee_{i=1}^{n} [sp(A_i, KB) \wedge \alpha_1 \wedge \alpha_2]$. On the other hand, since $KB \wedge \alpha_1$ is unsatisfiable, $KB \wedge \alpha_1 \wedge \alpha_2$ is also unsatisfiable, and hence, $KB * (\alpha_1 \wedge \alpha_2) = \bigvee_{i=1}^{n} [sp(A_i, KB) \wedge \alpha_1 \wedge \alpha_2]$.

(U6) Assume first that $KB \wedge \alpha_1$ is satisfiable. Then, $KB * \alpha_1$ is $KB \wedge \alpha_1$. Hence, since $KB * \alpha_1$ implies α_2, $KB \wedge \alpha_2$ is also satisfiable, and so, $KB * \alpha_2$ is $KB \wedge \alpha_2$. Thus, we know that $KB \wedge \alpha_1$ implies α_2 and $KB \wedge \alpha_2$ implies α_1, what immediately implies that $KB \wedge \alpha_1$ and $KB \wedge \alpha_2$, and so $KB * \alpha_1$ and $KB * \alpha_2$, are equivalent. Assume now that $KB \wedge \alpha_1$ is unsatisfiable. Then, by a similar argument as before, we conclude that $KB \wedge \alpha_2$ is also unsatisfiable. Thus, we infer that $\bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_1$ implies α_2 and $\bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_2$ implies α_1. Hence, by propositional logic alone, we conclude that $\bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_1$ and $\bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_2$, and so, $KB * \alpha_1$ and $KB * \alpha_2$, are equivalent.

(U7) If at least one of $KB \wedge \alpha_1$, $KB \wedge \alpha_2$ is satisfiable, the postulate clearly holds. (Note that in this case $KB \wedge (\alpha_1 \vee \alpha_2)$ is also satisfiable.) Assume, therefore, that $KB \wedge \alpha_1$ and $KB \wedge \alpha_2$ are both unsatisfiable, what implies that $KB \wedge (\alpha_1 \vee \alpha_2)$ is also unsatisfiable. Then $KB * \alpha_1 \wedge KB * \alpha_2$ is equivalent to (*) $\bigvee_{i=1}^{n} sp(A_i, KB) \wedge \alpha_1 \wedge \alpha_2$ and $KB * (\alpha_1 \vee \alpha_2)$ is equivalent to (**) $\bigvee_{i=1}^{n} sp(A_i, KB) \wedge (\alpha_1 \vee \alpha_2)$. Obviously, (*) implies (**). Note that we did not use the assumption that KB is complete. ■

It should be stressed that postulate (U8) need not to hold for the *LKBU* operator. To see this, assume that KB_1 is $\neg p \wedge q$, KB_2 is $p \wedge \neg q$ and α is p. Suppose further that there is one action, A, given by $p{:=}\top$. Then $(KB_1 \vee KB_2) * \alpha$ is equivalent to (*) $p \wedge \neg q$, whereas $(KB_1 * \alpha) \vee (KB_2 * \alpha)$ is equivalent to (**) p. Obviously, (*) and (**) are not equivalent.

The *LKBU* operator behaves similarly like belief revision operator of [2] if α is consistent with *KB*. Accordingly, one may suspect that the *LKBU* operator satisfies some of the AGM postulates that do not hold for the classical update operator. The next theorem, whose proof is straightforward, shows that this is indeed the case.

Theorem 2
The *LKBU* operator satisfies the following postulates:

(U9) If $KB \wedge \alpha$ is satisfiable, then $KB * \alpha \equiv KB \wedge \alpha$.

(U10) If $(KB * \alpha_1) \wedge \alpha_2$ is satisfiable, then $KB * (\alpha_1 \wedge \alpha_2)$ implies $(KB * \alpha_1) \wedge \alpha_2$.

6 Conclusions

In this paper, we have proposed a new approach to knowledge base update, called *LKBU*. The main difference between *LKBU* and other formalisms directed at knowledge base update is that an update formula, if consistent with a knowledge base, is regarded not as an effect of an action but rather as a piece of new information about the considered world.

We have formalized *LKBU* using Dijkstra's semantics, originally developed to reason about programs. Also, we have examined formal properties of our formalism with respect to update postulates of Katsuno and Mendelzon.

In this paper we do not consider integrity constraints. However this extension can be easily incorporated by using a technique proposed in [12].

As we observed (Examples 1, 3), in general, an update formula α do not allow to extract a single performed action. Instead, we receive a number of actions corresponding to α. An interesting topic to consider is to specify a minimal set of additional observations which could eliminate some of these actions. For instance, in Example 1 (first part) the new observation that the car is in Warsaw would eliminate the action *come-back-by-train*. We plan to work on this problem in the future.

References

[1] W. Ackermann. Uutersuchungen über das Eliminantionsproblem der mathematischen Logik. *Mathematische Annalen*, 110, 390-413, 1935.

[2] C. E. Alchourrón, P. Gärdenfors, D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic*, 50, 1985, 510-530.

[3] G. Brewka, J. Hertzberg. How to Do Things with Worlds: on Formalizing Actions and Plans. *Journal of Logic and Computation*. Vol. 3, No. 5, 1993, 517-532.

[4] F. M. Brown. *Boolean Reasoning*. Kluwer Academic Publishers, 1990.

[5] M-O. Cordier, P. Siegel. A Temporal Revision Model for Reasoning about World Change. In *Int'l. Conference on Principles of Knowledge Representation and Reasoning*, 732-739, Morgan Kaufmann, 1992.

[6] E. W. Dijkstra, C. S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[7] P. Doherty, W. Łukaszewicz, E. Madalińska-Bugaj. The PMA and Relativizing Minimal Change for Action Update. *Fundamenta Informaticae* **44** (2000) 95-131.

[8] P. Doherty, W. Łukaszewicz, A. Szałas. Computing Circumscription Revisited: A Reduction Algorithm. *Journal of Automatic Reasoning*, **18**, 1997, 297-336.

[9] A. Herzig. The PMA Revisited. In: *Proc. of the 5th Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1996, 40-50.

[10] G. Grahne. Updates and Counterfactuals. In *Proc. of the 2nd Int'l. Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1991, 269-276.

[11] H. Katsuno, A. O. Mendelzon. On the Difference between Updating a knowledge base and revising it. In: *Proc. of the 2nd Int'l. Conference on Principles of Knowledge Representation and Reasoning*, KR'91, Morgan Kaufmann, 1991, 387-395.

[12] Madalińska-Bugaj, E. (1997) How to Solve Qualification and Ramification Using Dijkstra's Semantics for Programming Languages, in: *AI*IA-97: Advances in Artificial Intelligence, Proceedings of 5th Congress of the Italian Association for Artificial Intelligence*, Springer-Verlag, Lecture Notes in Artificial Intelligence, **1321**, 381-392, 1997.

[13] B. Nebel. A Knowledge Level Analysis of Belief Revision. In: *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, 1989, 301-311.

[14] M. Winslett. *Updating Logical Databases.* Cambridge Tracks in Theoretical Computer Science. Cambridge University Press, 1990.

[15] M. Winslett. Updating Logical Databases. In: D. Gabbay, A. Galton, J. A. Robinson (eds.). *Handbook of Logic in Artificial Intelligence and Logic Programming.* Vol. 4 (Epistemic and Temporal Reasoning), 1995, Oxford University Press, 133-174.