Defining an Adaptable Framework for Behaviour Support Agents in Default Logic

Johanna Wolff[1,*], Victor de Boer[2], Dirk Heylen[1] and M. Birna van Riemsdijk[1]

[1]University of Twente, Drienerlolaan 5, 7522 NB Enschede, The Netherlands
[2]Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Abstract

In order to provide personalised advice, behaviour support agents need to consider the user's needs and preferences. This user model should be easily adaptable as the user's requirements will change during the long-term use of the agent. We propose a formal framework for such an agent in which the knowledge and the beliefs of the agent are represented explicitly and can be updated directly. Our framework is based on ordered default logic as defeasible reasoning allows the agent to infer additional information based on possibly incomplete knowledge about the world and the user. We also define updates on each component of the agent's framework and demonstrate how these updates can be used to resolve potential misalignments between the agent and the user. Throughout the paper we illustrate our work using a simplified example of a behaviour support agent intended to assist the user in finding a suitable form of exercise.

Keywords

Default Logic, Belief Revision, Behaviour Support Agent

1. Introduction

The rise of artificial assistants has lead to an increased interest in behaviour change support agents [1], which can support the user in establishing new routines and finding ways to achieve their goals consistently. In order for these agents to support each user as effectively as possible, the agents need to model the user's desires, needs and preferences as accurately as possible [2]. Since the agent should offer support over longer periods of time, it is likely that both the user and the surrounding context will go through changes throughout the agent's use [3]. Based on the emerging design principles of hybrid intelligence [4],we propose that the agent and the user should be able to collaborate in order to identify and implement the updates that are necessary to adapt the agent over time. This means that the user is in control of the agent's knowledge and beliefs [5], but the agent should be able to assist the user in determining how each change can be realised and explaining the effects that this will have.

While data-driven approaches such as machine learning, can be used to create a detailed and accurate user model [6], these models can be hard to adapt when the user's needs change [7]. The concepts captured in these user models are often not explicitly represented, which in turn means that they cannot be updated directly. This also makes it difficult for the user to understand exactly how their changes will affect the agent's output [8]. By using knowledge-driven methods, we can formalise changes to the user model within the framework, similarly for example to the work in [7]. In particular, we use default logic to model an agent with both dynamic knowledge and beliefs.

In this paper, we introduce a formal framework for a behaviour support agent which includes a model of the world and the user (Section 3.1). We use this framework to represent the agent's knowledge and beliefs explicitly within a default theory (Section 3.2). We use the defeasible nature of default logic to express beliefs about both the user and the world, which allows the agent to reason with incomplete knowledge and provide advice based on this. In order to make changes to the agent's knowledge and beliefs possible, we define updates to our formal framework (Section 4). These updates are based on existing work on belief revision updates for default logic [9]. We then compare this to previous work on user-agent misalignment [10, 11] and showcase how the formal updates can be used to resolve potential misalignment scenarios (Section 5) . Throughout the paper we will illustrate the framework using a simple running example of a support agent intended to assist the user in finding a suitable exercise based on their needs.

2. Preliminaries

We begin by introducing some preliminaries about the ordered default logic that we will be using for our agent

22nd International Workshop on Nonmonotonic Reasoning, November 2-4, 2024, Hanoi, Vietnam
*Corresponding author.
j.d.wolff@utwente.nl (J. Wolff); v.de.boer@vu.nl (V. d. Boer); d.k.j.heylen@utwente.nl (D. Heylen); m.b.vanriemsdijk@utwente.nl (M. B. v. Riemsdijk)
0009-0005-0178-9570 (J. Wolff); 0000-0001-9079-039X (V. d. Boer); 0000-0003-4288-3334 (D. Heylen); 0000-0001-9089-5271 (M. B. v. Riemsdijk)

framework in Section 3.2. We also present the belief revision operators that we will be using in Section 4.

2.1. Ordered Default Logic

Default logic was first introduced in [12] to formalise inference rules which are usually true but allow for exceptions. This is done using default rules of the form

$$\frac{\text{Prerequisite} \quad : \text{ Justification}}{\text{Consequent}} \ \delta.$$

This rule states that if the prerequisite is proven and it is consistent to assume the justification, then the consequent is inferred.

In the work of [13] there is additionally a strict partial ordering $\delta_1 < \delta_2$ on these default rules which expresses that δ_1 should only be applied if δ_2 has already been applied or is inapplicable. This results in an ordered default theory of the form $(K, D, <)$ in which K is a set of sentences, D is a set of default rules and $<$ is an ordering on the default rules in D. Intuitively, we understand the sentences in K to describe our, possibly incomplete, knowledge of the world while the default rules in D allow us to derive additional information based on our beliefs. The ordering $<$ may be used to express either preferences or priorities between these beliefs. A theory of this ordered default logic can be translated into standard default logic, allowing for an implementation in theorem provers for standard default logic [13].

When working with default theories, we are interested in the complete views of the world that are consistent with the initial theory, which we refer to as extensions. For an ordered default theory $T = (K, D, <)$ and any set of sentences S, we define $\Gamma(S)$ to be the smallest set satisfying the following properties:

1. $K \subseteq \Gamma(S)$
2. $Th(\Gamma(S)) = \Gamma(S)$
3. for all default rules $\frac{\alpha : \beta}{\gamma} \in D$, if $\alpha \in \Gamma(S)$ and $\neg\beta \notin S$ then $\gamma \in \Gamma(S)$

Here $Th(\Gamma(S))$ stands for the deductive closure of $\Gamma(S)$. We call a set of sentences E an extension of the theory T if $E = \Gamma(E)$. In the following, we will discuss only the consistent extensions of a theory. If we restrict ourselves to normal default rules where the justification and the consequent are the same, [12] has shown that this ensures the existence of a consistent extension. In the following, we will only consider default rules of this form.

Definition 1. *We define $\mathcal{E}(T)$ to be the set of all consistent extensions of the default theory $T = (K, D, <)$.*

The consistent extensions we have defined above do not yet take the ordering $<$ into account. To include this we use the notion of $<$-preserving extensions from [13].

Definition 2. *We define $Prereq(\Delta), Justif(\Delta)$ and $Conseq(\Delta)$ to be the set of prerequisites, justifications and consequents of the default rules δ in Δ. We take $GD(D, E)$ to be the set of default rules which generate the extension E and a grounded enumeration $(\delta_i)_{i\in I}$ of $GD(D, E)$ to be an order in which these rules can be applied.*

For a theory $T = (K, D, <)$, an extension $E \in \mathcal{E}(T)$ is $<$-preserving if there is a grounded enumeration $(\delta_i)_{i\in I}$ of $GD(D, E)$ so that for all $i, j \in I$ and $\delta \in D \setminus GD(D, E)$ it holds that

1. *if $\delta_i < \delta_j$ then $j < i$ and*
2. *if $\delta_i < \delta$ then $Prereq(\delta) \notin E$ or $K \cup Conseq(\{\delta_0, \dots, \delta_{i-1}\}) \vdash \neg Justif(\delta)$.*

Even if we know that $\mathcal{E}(T)$ is not empty, this does not ensure that a $<$-preserving extension of $T = (K, D, <)$ exists. Intuitively, this is because lower ranked default rules may have a consequent which can be used to infer the prerequisite of otherwise inapplicable, higher ranked default rules. This means that a higher ranked rule may be applied after the application of a lower ranked rule. As a result, the grounded enumeration of $GD(D, E)$ will not satisfy the first condition from Definition 2.

In [14] these inference relationships between the default rules of a theory are formalised using the dependency graph of the theory. The dependency graph $\mathcal{G}(D, K)$ captures whether default rules influence the applicability of other default rules, either positively by inferring the prerequisite or negatively by inferring the negation of the justification. We take $\mathcal{G}(D, K)$ to be the set of directed edges between the default rules in D.

In [13] this is used to specify conditions under which an order default theory has a $<$-preserving extension. For this, a default theory is considered *even* if all cycles of the dependency graph have an even number of negative relations. Intuitively this means that the application of a default rules does not negatively influence its own applicability. The ordering $<$ specifies that a lower ranked rule is only applicable after all higher ranked rules have been applied. This means that for each relation $(\delta < \delta')$, we want to ensure that δ does not affect the applicability of δ'.

Proposition 1. *As proven in [13], an ordered default theory $T = (K, D, <)$ is guaranteed to have a $<$-preserving extension if the dependency graph $\mathcal{G}(D, K)$*

1. *is even and*
2. *including the ordering $<$ does not create new cycles, so for all cycles $\mathcal{C}$ of $\mathcal{G}(D, K) \cup \{(\delta', \delta) \mid \delta < \delta'\}$, $\mathcal{C}$ is a cycle of $\mathcal{G}(D, K)$.*

Since the ordering $<$ is not necessarily total, it is possible that there are multiple $<$-preserving extensions.

2.2. Belief Revision

The field of belief revision is concerned with formalising changes to knowledge and belief bases. Since the knowledge and beliefs of a behaviour support agent are subject to change over time, we want to use update operations from belief revision to reflect this.

In general, belief revision is used to update a set of sentences S. We will be working with theory base revision operators [15], which do not require S to be deductively closed, as opposed to AGM operators [16]. Specifically we will use the operator $S * \varphi$ to add a sentence φ to S while ensuring the resulting set remains consistent and the operator $S \div \varphi$ to remove sentences from S until φ can no longer be inferred.

There is a range of work specifically concerned with integrating belief revision methods into default logic such as [17, 18, 19]. In our work we will use the operators defined in [9], which includes updates to the knowledge base and the default rules of a default theory.

If we use theory base revision operators, $K * \varphi$ and $K \div \varphi$ on the knowledge base K of a default theory T, [9] shows that this can be used to either add φ to all extensions of T or remove φ from $Th(K)$.

Additionally, [9] introduces updates on the set of default rules D. We use $D \div \delta = D \setminus \{\delta\}$ as an operator which removes the default rule δ from D and $D * \delta = D \cup \{\delta\}$ which adds a default rule δ to D.

3. Behaviour Support Agent

In the following section we introduce our framework that can be used to formalise a behaviour support agent. The agent will be able to select a suitable goal for the user to pursue based on the context that the user is currently in. The agent will then recommend actions which result in this goal being achieved, based on the user's preferences.

3.1. Syntax

We define a support agent for a set of possible actions A that the agent can recommend, the set of goals G the user may have and a set of contexts C that may affect the user's goals and actions.

Definition 3 (Atoms)**.** *We define the following sets of propositional atoms:*

- $A = \{a_1, \dots, a_n\}$ *describing the possible actions,*
- $G = \{g_1, \dots, g_m\}$ *describing the goals,*
- $C = \{c_1, \dots, c_l\}$ *describing different contexts and*
- $Atoms = A \cup G \cup C$.

Definition 4 (Language)**.** *Let $O = \{\top, \neg, \wedge, \vee, \rightarrow\}$ be a standard set of logical operators. We introduce the following propositional languages, defined over the operators O and sets of atoms in the usual way:*

- *The action language $\mathscr{L}_A$ over O and atoms A*
- *The goal language $\mathscr{L}_G$ over O and atoms G*
- *The context language $\mathscr{L}_C$ over O and atoms C*
- *The agent language $\mathscr{L}$ over O and atoms $Atoms$*

A plan for a goal $g \in G$ is a tuple of the form (g, φ), in which φ is a formula from $\mathscr{L}_A$ describing the actions that must be taken or avoided to achieve the goal g.

Definition 5. *The set of all possible plans LP is defined as follows: $LP = \{(g, \varphi) \mid g \in G, \varphi \in \mathscr{L}_A, \varphi \not\equiv \bot\}$.*

We introduce several types of rules which allow the agent to infer information based on its initial knowledge. Each rule is represented as a tuple (φ, ψ) in which φ is the prerequisite and ψ is the consequent. These rules will capture a form of defeasible reasoning in which we only infer the consequent if it is consistent with all other information. This means that if φ is true and nothing suggests otherwise, then ψ is inferred. For all types of rules φ may be $\top$ to signify that the rule has no prerequisite.

Context assumption rules are of the form (φ, ψ) with $\varphi, \psi \in \mathscr{L}_C$ describing aspects of the context. We can use these rules to make assumptions about the standard context that the user is in or to represent the beliefs the agent has about the relation between different contexts.

Definition 6. *The set of all possible context assumption rules is defined as $\mathscr{R}_C = \{(\varphi, \psi) \mid \varphi, \psi \in \mathscr{L}_C\}$.*

Goal selection rules are of the form (φ, g) with $\varphi \in \mathscr{L}_C$ describing the context and $g \in G$ describing the goal that should be achieved in this context. These are used to describe which goal the user should strive for in a certain context, if possible.

Definition 7. *The set of all possible goal selection rules is defined as $\mathscr{R}_G = \{(\varphi, g) \mid \varphi \in \mathscr{L}_c, g \in G\}$.*

Action selection rules are of the form (φ, ψ) with $\varphi \in \mathscr{L}$ and $\psi \in \mathscr{L}_A$. Here φ describes the circumstances in which the actions described in ψ may be taken, if they are possible. These circumstances can include certain context factors, the selected goals and other selected actions depending on the application.

Definition 8. *The set of action selection rules is defined as $\mathscr{R}_A = \{(\phi, \psi) \mid \phi \in \mathscr{L}, \psi \in \mathscr{L}_A\}$.*

We use $\mathscr{R} = \mathscr{R}_C \cup \mathscr{R}_G \cup \mathscr{R}_A$ to refer to all rules collectively. In order to be able to reason with these rules, we assign each rule $r \in \mathscr{R}$ a unique name $n(r)$. For this, we define an injective naming function n from the set of all rules $\mathscr{R}$ to a set of names N. We use these names to define an ordering on the rules. For simplicity of notation we will use the name and the rule itself interchangeably.

We represent the current state of the agent through its configuration, a tuple which specifies the formulas, rules and orderings that the agent reasons with.

Definition 9. *The configuration of an agent is a tuple* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *where* $W \subseteq \mathscr{L}$ *is the world knowledge,* $CC \subseteq L_C$ *are literals describing the current context,* $P \subseteq LP$ *is a set of plans,* $D_C \subseteq \mathscr{R}_C$ *is a set of context assumption rules,* $D_G \subseteq \mathscr{R}_G$ *is a set of goal selection rules,* $D_A \subseteq \mathscr{R}_A$ *is a set of action selection rules and* $<_C, <_G, <_A$ *are acyclic orderings on* D_C, D_G *and* D_A.

We also specify that for each goal $g \in G$, *there is only one plan* $p = (g, \varphi) \in P$. *If there are multiple ways to achieve a goal this should be specified through disjunctions in* φ, *rather than separate plans.*

To illustrate the use of each component of the agent's configuration we introduce a simplified example. We consider an agent which can give the user advice on how to lead a healthier lifestyle based on the user's medical data. For our purposes we assume that the agent should recommend one exercise for the user each day, but if the user's blood pressure is elevated, this should be a higher-intensity workout. The agent knows of two types of low-intensity exercises, namely walking and yoga, and two types of higher-intensity exercises, namely going for a run and weight training.

Example 1. *The agent is defined for the context factor* $C = \{BP\}$ *which indicates that the user's blood pressure is elevated, the set of goals* $G = \{LI, HI\}$ *which stand for low-intensity or higher-intensity workouts and the set of actions* $A = \{Walk, Yoga, Run, Weights\}$ *which are available.*

The world knowledge is given by $W = \{\varphi_{1g}, \varphi_{1a}\}$, *in which the formulas* $\varphi_{1g}, \varphi_{1a}$ *express that at most one goal and one action proposition can be true at the same time and therefore included in the agent's advice. This is needed to ensure that the agent only gives one recommended action each day. The current context CC contains the blood pressure information of the user. In this example we will assume that the blood pressure is high, so* $CC = \{BP\}$. *The plans corresponding to the goals are* $P = \{(LI, Walk \vee Yoga), (HI, Run \vee Weights)\}$.

We assume that if we have no information suggesting otherwise, the user's blood pressure is normal. Therefore $D_C = \{(\top, \neg BP)\}$. *The goal of a higher intensity workout should only be selected if BP is true, but the goal of a lower intensity workout can be selected in any situation so* $D_G = \{(BP, HI), (\top, LI)\}$. *For the sake of this example we assume that all the considered actions can be done in any context, which gives us the action selection rules* $\{(\top, Walk), (\top, Yoga), (\top, Run), (\top, Weights)\}$.

Since we only have one context assumption rule, we do not specify any ordering on this type of rule. The goal of a higher intensity workout, if applicable, is more important than the lower intensity workout so we have $(\top, LI) <_G (BP, HI)$ *The user has expressed that they prefer Yoga over Walking and Running over Weights so we specify* $(\top, Walk) <_A (\top, Yoga)$ *and* $(\top, Weights) <_A (\top, Run)$.

This results in an agent configuration $Ex = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$

3.2. Determining the Agent's Advice

For a given configuration *Conf* of the agent, we define a corresponding theory of ordered default logic $T = (K, D, <)$. We define the knowledge base K based on W, CC and P, the set of default rules based on D_C, D_G and D_A and the ordering $<$ based on $<_C$, $<_G$ and $<_A$. We take the sentences in K to describe the agent's, possibly incomplete, knowledge of the world while the default rules in D allow the agent to derive additional information based on its beliefs. The ordering $<$ provides a way to prioritise between these beliefs, either based on other beliefs about the world or based on the the user's preferences.

For this we translate every plan $p \in P$ of the form $p = (g, \varphi)$ into a formula $g \to \varphi \in \mathscr{L}$. We write $Tr(P) = \{g \to \varphi \mid (g, \varphi) \in P\}$ for the set of all such translated plans. We also translate each rule $r = (\varphi, \psi) \in D_i$ for $i \in \{C, G, A\}$ in the agent's configuration to a default rule of the form

$$\frac{\varphi \ : \ \psi}{\psi} \ r.$$

We take the transitive closures $<_A^+, <_G^+, <_C^+$ of the orderings to obtain strict partial orderings and define $<$ as the series composition partial order of D_C, D_G and D_A. This means in addition to ordering given in *Conf*, we also consider all rules regarding the context to be ranked higher than goal and action selection rules and we rank all goal selection rules higher than the action selection rules. We do this to make sure that the agent first considers the context it is in, then selects a goal for the user to pursue and then selects actions based on this.

Definition 10. *We define the ordered default theory* $DL(Conf)$ *corresponding to the agent whose configuration is given by* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *to be* $DL(Conf) = (K, D, <)$ *where* $K = W \cup CC \cup Tr(P)$ *is the knowledge base,* $D = \{\varphi \ : \ \psi / \psi \mid (\varphi, \psi) \in D_C \cup D_G \cup D_A\}$ *is the set of default rules that make up the belief base and* $<$ *is the series partial order of* D_C, D_G *and* D_A.

Based on our definition of a configuration, it is not yet guaranteed that this ordered default theory has a consistent extension. We define valid configurations to be those which do.

Definition 11. *A configuration Conf is valid if* $W \cup CC \cup Tr(P)$ *is consistent.*

Proposition 2. *For an ordered default theory* $DL(Conf) = (K, D, <)$ *based on a valid configuration Conf of an agent, the unordered default theory* (K, D) *has at least one consistent extension.*

This follows directly from [12] as, by definition, K is consistent and all default rules in D are normal. However, as discussed in Section 2, this does not yet guarantee the existence of a $<$-preserving extension. For this we define the notion of an effective configuration.

Definition 12. *An agent configuration is effective if it is valid and the ordered default theory $DL(Conf)$ fulfils the requirements from Proposition 1.*

We argue that an agent which is defined in an intuitively sensible way, will fulfil these conditions. If the dependency graph of the theory $DL(Conf)$ is not even, or goes against the ordering $<$, then this signifies an implicit inconsistency in the reasoning formalised in the agent. However, these conditions are difficult to formalise for the configuration of the agent, as they require us to consider the default theory. In future work we hope to determine clear requirements for agent configurations which guarantee for the existence of a $<$-preserving extension.

We use the $<$-preserving extensions of the default theory $DL(Conf)$ based on the agent's configuration to determine the advice that the agent should give the user. If there are multiple suitable extensions of $DL(Conf)$ then the agent requires a way to choose one of these extensions. This requires a meta-logic above the default logic that we have specified, so we will simply assume that such a selection can be made.

Definition 13. *For an agent with the effective configuration $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ which is translated into the ordered default theory $DL(Conf)$ with the $<$-preserving extension E, the agent's advice consists of the set of selected goals $\mathscr{A}_G = G \cap E$ and the set of recommended actions $\mathscr{A}_A = L(A) \cap E$.*

We showcase how the advice is obtained from the configuration of an agent by going through the configuration from Example 1.

Example 2. *The configuration Ex as defined in Example 1 is translated into the ordered default theory $T = (K, D, <)$ with*

- $K = \{\varphi_{1g}, \varphi_{1a}, BP, HI \rightarrow (Run \vee Weights), LI \rightarrow (Walk \vee Yoga)\}$,
- $D = \{\frac{\top : \neg BP}{\neg BP}\delta_1, \frac{BP : HI}{HI}\delta_2, \frac{\top : LI}{LI}\delta_3, \frac{\top : Walk}{Walk}\delta_4, \frac{\top : Yoga}{Yoga}\delta_5, \frac{\top : Run}{Run}\delta_6, \frac{\top : Weights}{Weights}\delta_7\}$ *and*
- $< = D_C; D_G; D_A = \{(\delta_i, \delta_1) \mid i = 2, \dots, 7\} \cup \{(\delta_i, \delta_j) \mid i = 4 \dots, 7; j = 2, 3\} \cup \{(\delta_3, \delta_2)\} \cup \{(\delta_4, \delta_5), (\delta_7, \delta_6)\}$

The default theory (K, D) has four possible extensions. We write only the relevant parts of each extension. These are $E_1 = \{HI, Run\}, E_2 = \{HI, Weights\}, E_3 = \{LI, Yoga\}$ and $E_4 = \{LI, Walk\}$. However, only E_1 is $<$-preserving. Therefore the agent's advice consists of the selected goal HI and the recommended action Run.

4. Agent Updates

In the previous section we have defined the configuration of a behaviour support agent and detailed how this determines the advice that the agent gives. In practice, the knowledge and beliefs of the agent change over time, so we need to be able to adapt the configuration of the agent. In this section, we define update operations on the agent's configuration which will allow us to add or remove information from each component individually.

For each of these components, we want the updates to be defined in such a way that the knowledge base $W \cup CC \cup Tr(P)$ remains consistent. This is necessary to ensure that we obtain a valid configuration as the result of the update. Unfortunately, we cannot always guarantee that the new configuration will also be effective due to the requirements from Definition 12. We will formally define the updates and also highlight such possible problems.

4.1. Updates to Knowledge Base

The knowledge base of the agent is made up of the world knowledge W, the current context information CC and the set of plans P. We want to be able to update these parts individually, but as explained above we have to consider them all to ensure the updates yield a valid configuration.

We can update the world knowledge W of the agent by adding a sentence $\varphi \in \mathscr{L}$ using the following update.

Definition 14. *For a configuration $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ and a formula $\varphi \in \mathscr{L}$ with $\{\varphi\} \cup CC \cup Tr(P)$ consistent, we define the update operation $Conf *_W \varphi = (W', CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ with $W' = (W * (\{\varphi\} \cup CC \cup Tr(P))) \setminus (CC \cup Tr(P))$.*

This means we use the theory base revision operator on W and update it with φ but also with $Tr(P)$ and CC. While we remove $Tr(P)$ and CC again afterwards, this approach guarantees that $W' \cup CC \cup Tr(P)$ will be consistent.

If we want to remove a formula from the world knowledge W, this is unproblematic for the consistency of the knowledge base.

Definition 15. *For a configuration $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ and a formula $\varphi \in \mathscr{L}$, we define the update operation $Conf \div_W \varphi = ((W \div \varphi), CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ to remove φ from W and its deductive closure $Th(W)$.*

We note that by defining the operator in this way, it is possible that φ is still contained in an extension E of $DL(Conf')$ due to the information in $CC \cup Tr(P)$ and the rules in D_C, D_G and D_A.

In order to update the current context CC we use the following operators, similarly to the ones for W.

Definition 16. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and context information* $\varphi \in L(C)$ *so that* $\{\varphi\} \cup W \cup Tr(P)$ *consistent, we define the update operation* $Conf *_{CC} \varphi = (W, CC', P, D_C, D_G, D_A, <_C, <_G, <_A)$ *with* $CC' = (CC * (\varphi \cup CC \cup Tr(P))) \setminus (W \cup Tr(P))$

Definition 17. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and context information* $\varphi \in C \cup \neg C$*, we define the update operations* $Conf \div_{CC} \varphi = (W, (CC \div \varphi), P, D_C, D_G, D_A, <_C, <_G, <_A)$ *to remove* φ *from CC.*

In order to update the plans P by adding or removing a plan $\pi = (g, \varphi)$ we use similar updates as for the knowledge base. When adding a new plan to P we need to ensure that the resulting set of plans only contains at most one plan per goal. This means we have to remove any previous plan (g, ψ) for the goal g before adding (g, φ) to P. We also make sure that the new information is consistent with the other components of the knowledge base.

Definition 18. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and a plan* $\pi = (g, \varphi) \in LP$ *so that* $\{Tr(\pi)\} \cup W \cup CC$ *consistent, we define the update operation* $Conf *_P \pi = (W, CC, P', D_C, D_G, D_A, <_C, <_G, <_A)$ *with* $P' = (P \setminus \{(g, \psi)\} \cup \{\pi\})$ *to add* π *to P.*

If we remove a plan $\pi = (g, \varphi)$ from P, this will result in a valid configuration. However, it is possible that the goal g is still the result of a goal selection rule and may be contained in an extension of $DL(Conf)$. This means the agent may advise the user to pursue the goal, despite there not being any action recommendation which corresponds to this. For this reason, this update should usually not be performed in isolation in practice.

Definition 19. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and a plan* $\pi = (g, \varphi) \in LP$*, we define the update operations* $Conf \div_P \pi = (W, CC, P', D_C, D_G, D_A, <_C, <_G, <_A)$ *with* $P' = P \setminus \pi$ *to remove* π *from P.*

Lemma 1. *The updates* $Conf *_K \varphi$,$Conf *_W \varphi$, $Conf \div_W \varphi$, $Conf *_{CC} \varphi$, $Conf \div_{CC} \varphi$ $Conf *_P \pi$ and $Conf \div_P \pi$, are well-defined. Additionally, if the default theory* $DL(Conf)$ *has a consistent extension, then the updated default theory* $DL(Conf')$ *will also have a consistent extension.*

Proof. This follows directly from the definitions of $*\varphi$ and $\div\varphi$ and Proposition 2. □

Unfortunately we cannot make the same claim regarding <-preserving consistent extensions. This is because any update to the knowledge base of a theory will affect the dependency graph $\mathcal{G}(D, K)$ of the theory $DL(Conf)$.

4.2. Updates to the Beliefs

The beliefs of the agent are made up of the context assumption rules D_C, the goal selection rules D_G and the action selection rules D_A. Since all types of rules and their respective orderings are defined and translated in the same way, we will only go through the updates of the context assumption rules in detail, the rest are analogous.

When adding a new context assumption rule to the agent's belief base, it is likely that this belief should also be integrated into the ordering $<_C$. However, this is not mandatory and can be done separately with the update operator on $<_C$ that we introduce below.

Definition 20. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and a context assumption rule* $r = (\varphi, \psi) \in \mathcal{R}_C$ *we define the update* $Conf *_{D_C} r = (W, CC, P, D'_C, D_G, D_A, <_C, <_G, <_A)$ *where* $D'_C = D_C \cup \{r\}$.

When an existing context assumption rule r needs to be removed from D_C, then we have to remove it from the ordering $<_C$ as well. This follows from the requirement that the ordering $<_C$ should be defined on the set D_C.

Definition 21. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and a context assumption rule* $r = (\varphi, \psi) \in \mathcal{R}_C$ *we define the update* $Conf \div_{D_C} r = (W, CC, P, D'_C, D_G, D_A, <'_C, <_G, <_A)$ *where* $D'_C = D_C \setminus \{r\}$ *and* $<'_C = <_C|_{D'_C}$.

Lemma 2. *The update operators* $Conf *_{D_C} r$ *and* $Conf \div_{D_C} r$ *are well-defined. If the default theory* $DL(Conf)$ *has a consistent extension, then the updated theory will also have a consistent extension.*

Proof. This follows directly from the definition and Proposition 2, since all default rules are still normal and the knowledge base is still consistent. □

Unfortunately, when adding a new rule we cannot guarantee the existence of a <-preserving extension as this rule could generate new cycles in the dependency graph that might not be even. However, when removing a rule this does result in a configuration $Conf'$ for which $DL(Conf)$ has a <-preserving extension.

Proposition 3. *For an effective configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *where* $DL(Conf)$ *has a <-preserving extension and a rule* $r \in \mathcal{R}_C$*, the ordered default theory* $DL(Conf')$ *with* $Conf' = Conf \div_{D_C} r$*, also has a <-preserving extension.*

Proof. To see this we check the conditions of Proposition 1. Removing a rule from the configuration and thereby removing a default rule from the default theory, cannot create any new cycles in the dependency graph. Since

Conf is an effective configuration, we know that all existing cycles are even, which means that the dependency graph of $DL(Conf')$ is even too. Additionally, any cycles that are removed from the dependency graph by removing r are also removed from the ordering $<$, so the ordering cannot introduce any new cycles. □

4.3. Updates to the Ordering

The ordering of the agent is made up of the orderings $<_C, <_G$ and $<_A$ on D_C, D_G and D_A respectively. We can update each of these orderings individually and only need to ensure that the resulting ordering is acyclic. Since all three orderings of the agent are defined in the same way, we will only go through the updates to the context ordering in detail; the others are analogous.

We can add a relation to $<_C$ using the following update.

Definition 22. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and a relation* (r_1, r_2) *with* $r_1, r_2 \in D_C$ *and* $(r_2, r_1) \notin <_C^+$ *we define the update* $Conf *_{<_C} (r_1, r_2) = (W, CC, P, D_C, D_G, D_A, <'_C, <_G, <_A)$ *where* $<'_C = <_C \cup \{(r_1, r_2)\}$.

When removing a relation (r_1, r_2) from $<_C$, we ideally want to remove the relation from $<_C^+$ to make sure it does not appear in $DL(Conf)$. However, this may require removing multiple relations from $<_C$ in the process. Since we have multiple choices for this, we will not include this in the update. If necessary, the ordering will need to be updated multiple times to fully remove the relation from $<_C^+$.

Definition 23. *For a configuration* $Conf = (W, CC, P, D_C, D_G, D_A, <_C, <_G, <_A)$ *and a relation* (r_1, r_2) *with* $r_1, r_2 \in D_C$ *we define the update* $Conf \div_{<_C} (r_1, r_2) = (W, CC, P, D_C, D_G, D_A, <'_C, <_G, <_A)$ *where* $<'_C = <_C \setminus \{(r_1, r_2)\}$.

Lemma 3. *The update operators* $Conf * <_C (r_1, r_2)$ *and* $Conf \div_{<_C} (r_1, r_2)$ *are well-defined. The resulting ordering* $<'_C$ *is acyclic. If the default theory* $DL(Conf)$ *has a consistent extension, then the updated theory will also have a consistent extension.*

Proof. This follows directly from definition as the knowledge base is still consistent and the default rules are still normal. □

When adding a new relation to the ordering $<_C$, this may create new cycles when combined with the dependency graph of $DL(Conf)$, which means we cannot guarantee that the resulting configuration will be effective. On the other hand, it is obvious that removing a relation does not have this issue, meaning that a useful configuration will be updated to another useful configuration.

5. Resolving Misalignments

With the framework we introduced, the agent is able to reason about a user model and a world model in order to provide personalised support to the user. By representing this explicitly, the user can interact with and adapt the agent's reasoning process directly using the updates that we have defined in the previous section. We chose to use default logic for this purpose because this allows the user to interact with and adapt the agent's reasoning process directly using the updates that we have defined in the previous section. A revision of the agent's reasoning process is necessary if the agent's advice does not align with the needs and wants of the user because the agent's advice contains either an action a or a goal g that the user does not agree with. In the following, we refer to these situations as misalignment scenarios, based on [10, 11]. In this section we will discuss the causes of misalignments that are identified in [10] and discuss how these can be resolved using the update operators defined in the previous section.

The three causes for these misalignments that are differentiated in [10] are the reasoning process of the agent being wrong, the agent's user model being wrong or something having changed in such a way that the agent needs to adapt to the new situation. For our purposes, we do not need to differentiate whether the misalignments occur due to a change or because of a mistake in the initialisation of the agent. Formally, these are handled the same way in this framework. We will discuss how each of the scenarios can be addressed by updating the configuration of the agent. We will give examples of potential misalignments with the advice provided by the agent we introduced in Example 1 and showcase how the realignment updates affect that configuration.

For the sake of this section we will assume that the agent and the user are able to identify the exact cause of the misalignment together. While this is not a trivial assumption and still a topic of active research, this is not something that can be achieved purely within the logical framework of an agent, making it out of the scope of this paper. For simplicity, we also assume that there is only one misalignment at a time.

5.1. Incorrect Reasoning

The reasoning process of the agent is based on logical inference, which cannot be incorrect by itself. However, if the world model of the agent is incorrect, then the agent may draw the wrong conclusions even if the user model is correct. This may refer to either the knowledge or the beliefs about the world, the latter including the prioritisation of these beliefs.

5.1.1. Incorrect World Knowledge

The first misalignment scenario we consider is the situation in which the agent has incorrect knowledge about the world. This means that there is either a sentence $\varphi \notin W$ that the agent does not know or the agent incorrectly accepts $\varphi \in W$ as known.

If the agent is missing the information φ, we can update the configuration $Conf$ of the agent using $Conf *_W \varphi$. By definition, this update requires φ to be consistent with $CC \cup Tr(P)$. If we assume that φ is the only cause of misalignment then this requirement also makes sense intuitively. In order for the agent to also be able to give advice, there are the additional requirements of the effective configuration. While we have explained above that these requirements are reasonable, they might be hard for the user to understand, especially if the agent becomes more complex. In future work we hope to look into ways to identify problematic cycles in the agents configuration and assist the user in resolving them.

Example 3. *In Example 2, we have identified that the agent's advice would be to pursue higher intensity exercising and specifically to go for a run. However, the user may be unable to go for a run because their regular running route is under construction. While this is related to a certain context in a way, which we discuss later on, we can treat this as direct information about the world. This means we update the agent's configuration with* $Ex' = Ex *_W \neg Run$. *As a result,* $\{HI, Run\}$ *is no longer an extension of* Ex', *and the agent's advice will instead be based on the extension* $E' = \{HI, Weights\}$.

Next we consider that the agent has wrongly identified the current context CC. If the mistake concerns a context factor c that is already in C, this can easily be resolved using the updates $Conf *_{CC} c$ and $Conf \div_{CC} c$, similarly to the updates to the knowledge base we described above. However, if the user thinks that a new context factor should be considered which is not yet in the language of the agent, then simply adding it to the current context CC is not enough. We likely need to add a context assumption rule which specifies whether this context factor is normally assumed to be true or false. Additionally, we probably want to include this context factor in the relevant goal and action selection rules. Since we do not have an update that can modify individual rules, this needs to be achieved by deleting the original rule, then including the modified rule and lastly reinstating the relevant orderings.

Example 4. *We consider that the user does not want to go for a run because it is raining. The original configuration of the agent did not account for the context of rain, so we need to perform a series of updates to include this. We begin by adding Rain to the description of the current context using* $Ex^1 = Ex *_{CC} Rain$. *We add a context assumption rule to specify that unless we have other knowledge, we assume it is not raining. We use the update* $Ex^2 = Ex^1 *_{D_C} \frac{\top : \neg Rain}{\neg Rain} \delta_8$. *We remove the action selection rule* δ_6 *which is concerned with running through* $Ex^3 = Ex^2 \div_{D_A} \delta 6$. *We add the modified action selection rule and obtain* $Ex^4 = Ex^3 *_{D_A} \frac{\neg Rain : Run}{Run}\}$. *We now restore the ordering by including* $\delta_7 < \delta_9$. *This gives us the final updated configuration* $Ex' = Ex^5 *_{<_A} (\delta_7, \delta_9)$

The resulting default theory $DL(Ex')$ *only has one* $<$*-preserving extension* $E' = \{HI, Weights\}$.

5.1.2. Incorrect Beliefs about the world

The agent's beliefs about the world, modelled as context assumption rules in D_C, may also be incorrect.

If a new belief δ needs to be adopted, then this can be done using the update $Conf *_{D_C} \delta$. This will cause the dependency graph of the agent to change, which may mean that the resulting configuration is not effective. This brings similar problems as a change in the world knowledge.

A belief δ can be removed from the agent's configuration with the update $Conf \div_{D_C} \delta$. While this will produce an effective configuration, this may lead to the agent's advice being less specific towards the user's context.

The beliefs about the world may also need to be prioritised differently, by updating the ordering $<_C$. While we have introduced updates $Conf *_{<_C} (\delta_i, \delta_j)$ and $Conf \div_{<_C} (\delta_i, \delta_j)$ to add or remove a relation to the ordering, in practice we will likely want to make more complex changes. While these can all be broken down into multiple applications of the two updates we have defined, this may be too complicated for the user to oversee themselves. Additionally, we also need to make sure that the relation remains acyclic and does not contradict the implicit ordering of the default theory modelled in the dependency graph.

Example 5. *So far the agent's configuration has included the context assumption rule* δ_1 *that unless other information is available, the user's blood pressure is normal. For this we remove the original rule* δ_1 *using* $Ex^1 = Ex \div_{D_C} \delta_1$ *and then add the new rule through* $Ex' = Ex^1 *_{D_C} \frac{\top : BP}{BP}$. *This means that even if the agent does not know the blood pressure levels of the user, so* $BP \notin CC$, *it will still recommend higher intensity exercises.*

5.2. Incorrect User Model

The user model of our agent contains information about the user's goals, the user's possible actions and the preferences regarding these. While humans may have goals and preferences that are not strictly logical, the formal framework of the agent requires the goals to be consistent with the current context and the knowledge about the world

and the dependency graph to fulfil the requirements from Proposition 1. The agent will need to collaborate with the user to ensure that the user model is as accurate as possible while still meeting the formal requirements.

5.2.1. Incorrect Goals

The goals of the user are what motivates the advice that the agent gives, but they are also subject to change as the needs and desires of the user develop. Each goal $g \in G$ has to correspond to a plan $\pi \in P$, so that the agent knows how each goal can be achieved. Additionally, a goal should occur in the consequent of a goal selection rule, otherwise it cannot be considered in the agent's advice. Any changes to the goals of the user therefore have to be captured in the set of plans and the goal selection rules.

If a new goal g is added, this goal needs a corresponding plan π, which can be added with the update $Conf *_P \pi$. Usually we will also add a goal selection rule $(\varphi : g/g)\,\delta_g$, for a sentence $\varphi \in \mathscr{L}_C$ describing the context in which the goal can be selected, by $Conf *_{D_G} \delta_g$. The goal selection rule will then likely need to prioritised adequately, by adding relations to the ordering $<_G$ using $Conf *_{<_G}$ and $Conf \div_{<_G}$. Each of these updates will affect the dependency graph, which means there is a risk that the resulting agent is not effective.

If the user no longer wants to pursue a goal g, then the relevant goal selection rules as well as their orderings need to be removed using the appropriate update.

If a plan or a goal selection rule needs to be changed, the original rule has to be deleted and the new version needs to be added in separate updates as for updates to the world model.

Example 6. *We want the agent to consider the additional goal Rest when giving advice. This goal is achieved if no exercise is done, so the plan is* $\pi = (Rest, \neg Walk \wedge \neg Yoga \wedge \neg Run \wedge \neg Weights)$. *For now we do not have any context requirements for this goal to be selected but we prioritise it above the other goal selection rules. We begin by including the plan p in the set of plans using* $Ex^1 = Ex *_P p$. *We then include the goal selection rule* $\top : Rest/Rest\,\delta_g$ *with the update* $Ex^2 = Ex^1 *_{D_G} \delta_g$. *Lastly, we include the relations* (δ_2, δ_g) *and* (δ_3, δ_g) *in the ordering* $<_G$ *through the update* $Ex' = Ex^2 *_{<_G} \{(\delta_2, \delta_g), (\delta_3, \delta_g)\}$. *This results in an effective configuration* Ex' *with the* $<$*-preserving extension* $E' = \{Rest, \neg Walk, \neg Yoga, \neg Run, \neg Weights\}$.

5.2.2. Incorrect Actions

The actions that are recommended by the agent are determined by the plans for the selected goals and the action selection rules. The user may either want to change the context prerequisites for selecting certain actions, add a new action selection rule or remove an existing action selection rule. These can each be achieved using the updates $Conf *_{D_A}$ and $Conf \div_{D_A}$. If the preferences of the user regarding the actions need to be changed, this can be handled analogous to the change of the ordering $<_C$, using the updates $Conf *_{<_A}$ and $Conf \div_{<_C}$.

Example 7. *In Example 3, the user was not able to go for a run and we added* $\neg Run$ *to the agent's world knowledge. This time we will remove the action selection rule for Run instead. By performing the update* $Ex' = Ex \div_{D_A} \delta_6$, *the action selection rule and its corresponding ordering in* $<_A$ *are removed. This leads to the same* $<$ $-$*preserving extension as in Example 3,* $E' = \{HI, Weights\}$. *While these updates formally have the same result, they intuitively mean different things. In Example 3 the user is not able to run due to outside circumstances from the world that may be resolved at some point. We can remove* $\neg Run$ *from the knowledge base W and are still able to use the previous user model. The update in this example on the other hand removes running as a possible action from the user model, indicating that the user no longer views this as an option.*

6. Discussion

We have introduced a formal framework which can be used to specify the configuration of a behaviour support agent. The configuration can be translated into a theory of ordered default logic and the $<$-preserving expansions of this theory determine the advice that the agent presents to the user. We have also defined updates on the configuration of the agent which can add or remove information from each of the components. These updates can be used to resolve misalignments between the user and the agent.

In order to use the updates for realignment, it is necessary for the agent and the user to accurately identify the precise cause of the misalignment. While this problem needs to be addressed through communication between the agent and the user [10], we want to facilitate this process using the formal framework of the agent. In future work we hope to study whether the structure of the framework is understandable to users, how we can formally identify potential causes of misalignment and how we can explain problematic cycles in the dependency graph to assist the user in resolving these.

So far we have only included the basic updates which add or remove information from each component of the agent's configuration. However, in [9] there are also other updates on default theories, such as introducing a possibility by ensuring that there is at least one consistent extension which contains a sentence φ. It may be interesting to see whether these updates can be adapted for ordered default logic and what they would mean for the agent's configuration. The updates we have used so far have also each been permanent changes of the agent's configuration. In practice there may be situations which

require different advice in the moment but should not be considered in the future. These might require different types of updates to optimise the computational complexity of the agent.

In order to further demonstrate the potential of our framework, we also hope to implement the example agent we have presented in this paper. Since ordered default logic can be translated into regular default logic using the process described in [13], we can use existing solvers for default logic to implement the reasoning of the agent. By combining this with implementations of belief revision operators, we can study how our framework behaves in a real application. For this, we will likely also need to optimise the agent to reduce the computational complexity. A first step for this is to consider the work of [14] which discusses specific types of default theories for which an extension can be found in polynomial time.

Acknowledgments

This research was partly funded by the Hybrid Intelligence Center, a 10-year programme funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, grant number 024.004.022.

References

[1] H. Oinas-Kukkonen, Behavior change support systems: A research model and agenda, in: T. Ploug, P. Hasle, H. Oinas-Kukkonen (Eds.), Persuasive Technology, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 4–14. doi:10.1007/978-3-642-13226-1_3.

[2] N. Albers, M. A. Neerincx, W.-P. Brinkman, Persuading to prepare for quitting smoking with a virtual coach: Using states and user characteristics to predict behavior, in: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23, International Foundation for Autonomous Agents and Multiagent Systems, 2023, p. 717–726.

[3] M. B. van Riemsdijk, C. M. Jonker, V. Lesser, Creating socially adaptive electronic partners: Interaction, reasoning and ethical challenges, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, p. 1201–1206.

[4] Z. Akata, D. Balliet, M. de Rijke, F. Dignum, V. Dignum, G. Eiben, A. Fokkens, D. Grossi, K. Hindriks, H. Hoos, H. Hung, C. Jonker, C. Monz, M. Neerincx, F. Oliehoek, H. Prakken, S. Schlobach, L. van der Gaag, F. van Harmelen, H. van Hoof, B. van Riemsdijk, A. van Wynsberghe, R. Verbrugge, B. Verheij, P. Vossen, M. Welling, A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence, Computer 53 (2020) 18–28. doi:10.1109/MC.2020.2996587.

[5] S. S. Sundar, Rise of Machine Agency: A Framework for Studying the Psychology of Human–AI Interaction (HAII), Journal of Computer-Mediated Communication 25 (2020) 74–88. doi:10.1093/jcmc/zmz026.

[6] D. Goldenberg, K. Kofman, J. Albert, S. Mizrachi, A. Horowitz, I. Teinemaa, Personalization in practice: Methods and applications, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, WSDM '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1123–1126.

[7] E. Fermé, M. Garapa, M. D. L. Reis, Y. Almeida, T. Paulino, M. Rodrigues, Knowledge-driven profile dynamics, Artificial Intelligence 331 (2024) 104117. doi:10.1016/j.artint.2024.104117.

[8] P. B. De Laat, Algorithmic decision-making based on machine learning from big data: can transparency restore accountability?, Philosophy & technology 31 (2018) 525–541.

[9] G. Antoniou, On the dynamics of default reasoning, International Journal of Intelligent Systems 17 (2002) 1143–1155. doi:https://doi.org/10.1002/int.10065.

[10] P.-Y. Chen, M. Tielman, D. Heylen, C. Jonker, M. Riemsdijk, Acquiring semantic knowledge for user model updates via human-agent alignment dialogues: An exploratory focus group study, in: HHAI 2023: Augmenting Human Intellect - Proceedings of the 2nd International Conference on Hybrid Human-Artificial Intelligence, IOS Press, 2023, pp. 93–108. doi:10.3233/FAIA230077.

[11] J. Wolff, V. De Boer, D. Heylen, M. B. Van Riemsdijk, Using default logic to create adaptable user models for behavior support agents, HHAI 2024: HYBRID HUMAN AI SYSTEMS FOR THE SOCIAL GOOD (2024) 350.

[12] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132. doi:https://doi.org/10.1016/0004-3702(80)90014-4, special Issue on Non-Monotonic Logic.

[13] J. P. Delgrande, T. Schaub, Expressing preferences in default logic, Artificial Intelligence 123 (2000) 41–87. doi:https://doi.org/10.1016/S0004-3702(00)00049-7.

[14] C. H. Papadimitriou, M. Sideri, Default theories that always have extensions, Artificial Intelligence 69 (1994) 347–357. doi:https://doi.org/10.1016/

0004-3702(94)90087-6.

[15] M.-A. Williams, Iterated theory base change: A computational model., Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (1995) 1541–1549.

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

[17] T. I. Aravanis, P. Peppas, Belief revision in answer set programming, in: Proceedings of the 21st Pan-Hellenic Conference on Informatics, PCI '17, Association for Computing Machinery, New York, NY, USA, 2017. doi:10.1145/3139367.3139387.

[18] P. Krümpelmann, G. Kern-Isberner, Belief base change operations for answer set programming, in: L. F. del Cerro, A. Herzig, J. Mengin (Eds.), Logics in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 294–306.

[19] S. Pandžić, On the dynamics of structured argumentation: Modeling changes in default justification logic, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12012 LNCS (2020) 222 – 241. doi:10.1007/978-3-030-39951-1_14.