Olaaaf: A General Adaptation Prototype

Erwan Diebold, Yan Kabrit, Axel Kril, Jean Lieber, Paul Malvaud, Emmanuel Nauer, Jules Sipp

Olaaaf: a General Adaptation Prototype

Erwan Diebold[1], Yan Kabrit[2], Axel Kril[2],
Jean Lieber[3], Paul Malvaud[2], Emmanuel Nauer[3], and Jules Sipp[2]

[1] Université de Lorraine, master informatique
[2] Université de Lorraine, master sciences cognitives
[3] LORIA, Université de Lorraine, CNRS, Inria, F-54000 Nancy, France

Abstract. Adaptation is often considered a complex issue during the design of a case-based reasoning system. Various approaches can be found in the literature, but their scopes are often limited to a relatively narrow range of applications. However, a general approach to adaptation based on belief revision has been developed over the years and applied in several formalisms, these formalisms being chosen for particular needs. This article presents the first version of Olaaaf, a general adaptation prototype based on belief revision whose long-term objective is to cover a wide range of adaptation processes. It is based on a formalism that covers both attribute-value pairs (often used for representing cases) and taxonomies (often used for representing domain knowledge). It is shown through an example how this system works, and it is discussed how it can be used for other complex adaptations.

Keywords: case-based reasoning, adaptation, general prototype

1 Introduction

Case-based reasoning (CBR [27, 1]) aims to solve problems by retrieving and reusing previous problem-solving episodes (or cases). The reuse step consists either (a) in reusing as such the solution of the retrieved case(s) (and this is appropriate for some applications), (b) in leaving to the user the adaptation of this solution (this is often acceptable, since, for a human being, retrieval is often considered as tedious, while adaptation sometimes appears as a simple task for humans while more difficult to implement), or (c) by implementing an automatic adaptation procedure. The approach (c) requires some work to design and implement such a procedure. Indeed, to the best of our knowledge, very little work has been done on the development of domain-independent adaptation engines. In particular, in a recent survey on general frameworks for CBR [28], the adaptation procedures of the described frameworks were, at best, limited to null adaptation, ontology-based substitutions, handling adaptation rules, or adaptation methods defined by the developers using such a framework, hence domain dependent.

Now, a general approach to adaptation has been developed, studied and applied. This approach is based on belief revision operators and requires the implementation of such operators in a formalism in which cases and domain knowledge are defined. The first study on this work was published in 2007 [18] and

it has been developed from there (see e.g. [6]), with applications to adaptation processes for the Taaable system in the cooking domain [8], in three different formalisms [23, 5, 12]. This was the first motivation for developing Olaaaf,[1] a revision-based inference engine in a general formalism.

This article presents Olaaaf, a domain-independent case adaptation prototype in a formalism that is general enough to express various kinds of knowledge, including attribute-value pairs and taxonomies. This has a certain level of generality since, in CBR, cases are often represented in an attribute-value representation, and domain knowledge is often represented by taxonomies, i.e., class hierarchies organized under the subclass-of relation. More precisely, this formalism is based on constraints (e.g. linear numerical constraints or propositional variables) and on the connectives of propositional logic ($\neg$, $\wedge$, $\vee$, etc.).

A running example, in the cooking domain that illustrates the case adaptations in Olaaaf is introduced now. The target problem is "I want a recipe of milkshake with kiwis." The retrieved case is a recipe for a banana milkshake, consisting in mixing the following ingredients: 2 bananas, 4 tablespoons of granulated sugar, 2 packets of vanilla sugar, 1 liter of cow milk, 4 ice cubes. To perform the adaptation, the following domain knowledge is used:

DK1 Bananas and kiwis are fruits.

DK2 For each food type and unit, there is a known correspondence of one unit of this food type to its mass, e.g. the mass of 1 banana and the mass of 1 tablespoon of granulated sugar.

DK3 There are relations between quantities of one type of food and its subtypes in a taxonomy (e.g. the mass of fruits is the sum of the masses of bananas, kiwis, etc.). In the same line of idea, the sweetening power is known for every food type, e.g. 1 for granulated sugar, 0.158 for bananas (1 gram of banana has the same sweetening power as 0.158 gram of granulated sugar), etc.[2]

DK4 Almond milk, cow milk and soy milk are 3 types of milks (and, to make it simpler, it is assumed that there are no other types of milk in my fridge).

DK5 Cow milk and soy milk associated with kiwis give a bitter taste.

DK6 A milkshake is a dessert, and a dessert must not be bitter.

The expected adaptation consists of substituting bananas for kiwis and, to avoid bitterness (forbidden in desserts), substituting cow milk for almond milk which does not taste bitter when associated with kiwis. It also consists in changing quantities (number of fruits, mass of ingredients), in order to conserve the mass of fruits and total mass of the milkshake, and to preserve the sweetening power.

In the Taaable system, this adaptation was performed in two steps, using two adaptation engines, working with two different formalisms: propositional logic to

[1] The Olaaaf system, available at `https://github.com/OlaaafEngine/Olaaaf`, is distributed under a free MIT license. This repository also contains a video demo and the examples presented in this paper.

[2] Numerical information about conversions for DK2 and DK3 has been found on the Web, in particular, at `https://fdc.nal.usda.gov`.

compute substitutions of ingredient types and conjunction of linear constraints to adapt ingredient quantities once the substitutions of ingredient types are known. On the contrary, Olaaaf solves this adaptation problem in a single step.

Section 2 recalls some general notions related to CBR, in particular, to adaptation, and to the general approach to adaptation that is used in our system. Olaaaf is presented in Section 3: how it can be used to solve adaptation problems and the main principles of its algorithm. As Olaaaf has been announced as a general tool for case adaptation, Section 4 discusses its scope: What are the adaptation inferences that it can or cannot (yet) draw? Finally, Section 5 concludes and gives some future direction of work for the development of future versions of Olaaaf.

2 Preliminaries

In Section 2.1, some basic notions about CBR are recalled, together with the notations used throughout the article. The general approach to adaptation by Olaaaf is revision-based adaptation (described in Section 2.4), which is based on belief revision (whose principles are recalled in Section 2.3), which requires a few reminders about formal logics (Section 2.2).

2.1 Case-based reasoning: notions and notations

In this paper, a problem is denoted by $\mathtt{x}$ and a solution by $\mathtt{y}$. In particular, the target problem is denoted by $\mathtt{x}^{\mathtt{tgt}}$ and the solution proposed by the CBR system by $\mathtt{y}^{\mathtt{tgt}}$. A case is the representation of an episode of problem solving, often given only by a problem-solution pair $(\mathtt{x}, \mathtt{y})$ where $\mathtt{y}$ is a solution of $\mathtt{x}$. Note that, in the context of this article (and of the examples), this a priori distinction between the problem part and the solution part of a case is not necessary. Two kinds of cases are distinguished: correct and incorrect cases; a case is correct if it leads to an acceptable solution to the problem it solves. It should be noted that this distinction between correct and incorrect cases is usually incompletely known by the CBR system. A *source case*, denoted by $\mathtt{c}^s$, is a case available to a CBR system, and the set of source cases constitutes the *case base* $\mathtt{CB}$. The source cases are supposed to be correct.

In this article, only single case reuse is considered, that is, the process model of CBR consists in (1) retrieving a $\mathtt{c}^s \in \mathtt{CB}$ deemed similar to $\mathtt{x}^{\mathtt{tgt}}$, (2) adapting $\mathtt{c}^s$ in order to propose a solution $\mathtt{y}^{\mathtt{tgt}}$ to $\mathtt{x}^{\mathtt{tgt}}$. Other CBR steps are not considered in this article. In fact, this article is primarily concerned with step (2), adaptation.

The knowledge model of a CBR system usually consists of four containers [26]: the case base $\mathtt{CB}$, the *domain knowledge* $\mathtt{DK}$, the *adaptation knowledge* $\mathtt{AK}$, and the *retrieval knowledge* $\mathtt{RK}$. $\mathtt{CB}$ has been introduced above and $\mathtt{RK}$, is not considered in this article.

$\mathtt{DK}$ concerns cases and may be seen as a set of integrity constraints. In particular, $\mathtt{DK}$ often contains a taxonomy, i.e., a set of ordered pairs (A, B) where A is a subclass of B and such a pair corresponds to the constraint that prohibits

having an instance of A that is not an instance of B: written in propositional logic, the constraint $A \rightarrow B$ means that $A \wedge \neg B$ cannot be true. Thus, DK can be seen as a set of necessary conditions for the correctness of a case.

AK concerns the way cases relate in the case space: given a case that is supposed to be correct (e.g. a source case), it gives knowledge about other cases that *should* be correct, with some uncertainty. This uncertainty is often measured by a number. In this article, this number is called a *cost*: given $\mathtt{c}^1$ and $\mathtt{c}^2$ two cases, the former known to be correct (e.g. $\mathtt{c}^1 \in \mathtt{CB}$), AK gives an estimation of this cost c, which can be interpreted, e.g., in a probabilistic way: $c = -\log P$ where P estimates the probability that $\mathtt{c}^2$ is correct (knowing that $\mathtt{c}^1$ is). For example, AK may contain adaptation rules. Such an adaptation rule states that if a case $\mathtt{c}^1$ and a problem $\mathtt{x}^2$ are related according to the condition of the rule, then the rule is triggered and gives a modification of $\mathtt{c}^1$ into $\mathtt{c}^2$, such that $\mathtt{c}^2$ represents a solving of the problem $\mathtt{x}^2$ and the correctness of $\mathtt{c}^2$ is estimated according to the cost associated with the adaptation rule.

It is important to distinguish the roles of DK and of AK. Indeed, DK is about cases considered in isolation, whereas AK concerns variations between cases.

2.2 Some basic notions about some monotonic logics

A monotonic logic, in all generality, is an ordered pair $(\mathcal{L}, \models)$ where $\mathcal{L}$ is a language defined by a formal syntax (a *formula* is by definition an element of $\mathcal{L}$) and $\models$ is a binary relation that relates a set of formulas B to a formula α where $\mathcal{B} \models \alpha$ reads "$\mathcal{B}$ entails α."

For some logics (including propositional logic and the formalism of the Olaaaf system), the relation $\models$ is defined as follows. A fixed set Ω is given, and an *interpretation* is an element of Ω. A fixed mapping $\mathcal{M}$ is given, which associates to a formula α a subset $\mathcal{M}(\alpha)$ of Ω (the definitions of Ω and of $\mathcal{M}$ depend on the chosen logic). For a set of formulas $\mathcal{B} = \{\alpha_1, \alpha_2, \ldots, \alpha_p\}$, $\mathcal{M}(\mathcal{B})$ is defined by $\mathcal{M}(\mathcal{B}) = \mathcal{M}(\alpha_1) \cap \mathcal{M}(\alpha_2) \cap \ldots \cap \mathcal{M}(\alpha_p)$ (if $\mathcal{B} = \emptyset$, $\mathcal{M}(\mathcal{B}) = \Omega$). Then $\models$ can be defined for $\mathcal{B}$, a set of formulas, and $\alpha \in \mathcal{L}$ by $\mathcal{B} \models \alpha$ if $\mathcal{M}(\mathcal{B}) \subseteq \mathcal{M}(\alpha)$. Given $\alpha, \beta \in \mathcal{L}$, α is equivalent to β, denoted by $\alpha \equiv \beta$, if $\mathcal{M}(\alpha) = \mathcal{M}(\beta)$. A formula α is inconsistent if $\mathcal{M}(\alpha) = \emptyset$. A formula α is inconsistent with a formula β if $\mathcal{M}(\alpha) \cap \mathcal{M}(\beta) = \emptyset$. The principle of irrelevance of syntax means that two equivalent formulas are interchangeable within an inference.

Finally, in the following, it is assumed that $\mathcal{L}$ uses the unary connective $\neg$ and the binary connectives $\wedge$ and $\vee$, and that these connectives have the following semantics: for $\alpha, \alpha_1, \alpha_2 \in \mathcal{L}$, $\mathcal{M}(\neg\alpha) = \Omega \backslash \mathcal{M}(\alpha)$, $\mathcal{M}(\alpha_1 \wedge \alpha_2) = \mathcal{M}(\alpha_1) \cap \mathcal{M}(\alpha_2)$ and $\mathcal{M}(\alpha_1 \vee \alpha_2) = \mathcal{M}(\alpha_1) \cup \mathcal{M}(\alpha_2)$. So, in particular, α_1 is inconsistent with α_2 iff $\alpha_1 \wedge \alpha_2$ is insatisfiable.

2.3 Belief revision

Consider an agent whose set of beliefs represented by a formula ψ is confronted with new beliefs represented by a formula μ, and that these new beliefs are supposed to take priority over the old ones. If ψ is consistent with μ (i.e.,

$\mathcal{M}(\psi \wedge \mu) \neq \emptyset$), then the revised set of agent beliefs is the union of old and new beliefs, represented by $\psi \wedge \mu$. Otherwise, $\psi \wedge \mu$ is insatisfiable, so some changes must be made in ψ (not in μ since it takes priority over ψ) in ψ' so that $\psi' \wedge \mu$ is consistent. Then, the result of belief revision of ψ by μ is $\psi \dot{+} \mu = \psi' \wedge \mu$. According to the principle of minimal change defended in the so-called AGM theory of belief revision [2], the modification $\psi \mapsto \psi'$ must be "minimal" in some sense, and, since the measure of change to assess this minimality is not unique, there are in general many *revision operators* $\dot{+}$.

Let $(\mathcal{L}, \models)$ be a monotonic logic whose semantics is given by the set of interpretations Ω and the mapping $\mathcal{M}$, and let $\texttt{dist}$ be a distance function on Ω. Technically, $\texttt{dist}$ is only assumed (1) to return a nonnegative real number and (2) to verify, for $\mathcal{I}, \mathcal{J} \in \Omega$, that $\texttt{dist}(\mathcal{I}, \mathcal{J}) = 0$ iff $\mathcal{I} = \mathcal{J}$. Thus, neither symmetry nor triangular inequality is required for $\texttt{dist}$. A revision operator $\dot{+}^{\texttt{dist}}$ can be semantically defined, up to syntax, as follows: $\mathcal{J} \in \mathcal{M}\left(\psi \dot{+}^{\texttt{dist}} \mu\right)$ if $\mathcal{J} \in \mathcal{M}(\mu)$ and $\mathcal{J}$ is maximally close to elements of $\mathcal{M}(\psi)$ according to $\texttt{dist}$. Formally:

$$\mathcal{M}\left(\psi \dot{+}^{\texttt{dist}} \mu\right) = \{\mathcal{J} \in \mathcal{M}(\mu) \mid \texttt{dist}(\mathcal{M}(\psi), \mathcal{J}) = \texttt{dist}(\mathcal{M}(\psi), \mathcal{M}(\mu))\}$$

where $\texttt{dist}(\mathcal{M}(\psi), \mathcal{J}) = \inf_{\mathcal{I} \in \mathcal{M}(\psi)} \texttt{dist}(\mathcal{I}, \mathcal{J})$ and $\texttt{dist}(\mathcal{M}(\psi), \mathcal{M}(\mu)) = \inf_{\mathcal{I} \in \mathcal{M}(\psi), \mathcal{J} \in \mathcal{M}(\mu)} \texttt{dist}(\mathcal{I}, \mathcal{J})$.

2.4 Revision-based adaptation

In [18], an approach to adaptation based on belief revision has been introduced. It presupposes that the cases and the domain knowledge are represented in the same monotonic logic $(\mathcal{L}, \models)$. The idea is to use a revision operator $\dot{+}$ on this logic to modify the retrieved source case $\texttt{c}^s$ so that it becomes consistent with the target problem. Now, both $\texttt{c}^s$ and $\texttt{x}^{\texttt{tgt}}$ must be considered with the domain knowledge, so the revision to be performed is of $\psi = \texttt{DK} \wedge \texttt{c}^s$ by $\mu = \texttt{DK} \wedge \texttt{x}^{\texttt{tgt}}$. Finally, the choice of the revision operator for that purpose is of importance in the same way as the choice of a similarity measure is important for case retrieval: a basic choice often gives interesting results, but a better choice gives better results. The idea is to choose a revision operator $\dot{+}^{\texttt{AK}}$ parametrized by the available adaptation knowledge $\texttt{AK}$. Therefore, the revision-based adaptation of $\texttt{c}^s$ to a problem $\texttt{x}^{\texttt{tgt}}$ is defined by:

$$\texttt{c}^{\texttt{tgt}} \quad = \quad (\texttt{DK} \wedge \texttt{c}^s) \quad \dot{+}^{\texttt{AK}} \quad (\texttt{DK} \wedge \texttt{x}^{\texttt{tgt}}) \tag{1}$$

where $\texttt{c}^{\texttt{tgt}}$ is a solving of $\texttt{x}^{\texttt{tgt}}$ that leads to the solution $\texttt{y}^{\texttt{tgt}}$. It can be noted that an operator $\dot{+}^{\texttt{AK}}$ can often be written as an operator $\dot{+}^{\texttt{dist}}$, for a distance function that takes into account adaptation knowledge, for example, adaptation rules.

Various distance-based revision engines have been defined for revision-based adaptation in several formalisms, in particular, to be used in different versions of the Taaable system, a contestant of every edition of the Computer Cooking Contest (see [8] for a first synthesis of Taaable). The first engines were developed

for propositional logic: first with no adaptation knowledge, using an arbitrary distance function `dist` [18], then for adaptation knowledge using adaptation rules [23]. This was used in Taaable to adapt a recipe represented at the level of its food types and recipe types. To be able to adapt the quantities of ingredients in Taaable, a revision engine has been developed in the formalism of conjunctions of linear constraints. It should be noted that both propositional logic and this latter formalism are fragments of the formalism of Olaaaf as presented in the next section. Then, a belief revision engine was developed for qualitative algebras (i.e. a family of formalisms including Allen algebra [3] and RCC8 [24], for reasoning on temporal or spatial constraints) and applied, in particular, to the preparation part of the recipes in Taaable [12]. Then, this has been extended to belief revision in the propositional closure of qualitative algebras [10]. Finally, it has been shown that analogical extrapolation (that is, case-based inference based on analogical proportion) can be related to revision-based adaptation [20].

3 Olaaaf: how it can be used and how it is implemented

The formalism used by Olaaaf is described in Section 3.1. Section 3.2 explains how this adaptation system can be used on the running example and Section 3.3 specifies it. The complete description of Olaaaf algorithm is beyond the scope of this article (and its description is long), but its principles are given in Section 3.4.

3.1 Olaaaf formalism for representing cases and domain knowledge

Syntax. Let $\mathbb{B} = \{\mathtt{F}, \mathtt{T}\}$ be the set of Booleans. Let $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ be the set of integers, of rational numbers and of real numbers. A *variable* is a symbol x associated with a type $\tau(x)$, where $\tau(x)$ is either $\mathbb{Z}$, $\mathbb{R}$ or a finite set S defined in an enumerated way (e.g. $S = \{a, b, c\}$ or $S = \mathbb{B}$). The finite set of variables is denoted by $\mathcal{V}$.

The atoms of Olaaaf's formalism, called *constraints*, are:

- Linear constraints: expressions of the form $a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$ where $a_1, a_2, \ldots, a_n \in \mathbb{Q}$ and, for each $k \in [\![1, n]\!]$, x_k is a numerical variable (i.e. either $\tau(x_k) = \mathbb{Z}$ or $\tau(x_k) = \mathbb{R}$) if $a_k = 0$ then the term a_kx_k can be omitted;
- Constraints of the form $x = v$ where $\tau(x)$ is a finite set S and $v \in S$.

A formula of Olaaaf's formalism is either a constraint or an expression of one of the forms $\neg\alpha$, $\alpha \wedge \beta$ and $\alpha \vee \beta$ where α and β are formulas.

The following abbreviations are made (where $\underline{iaaf}$ stands for "is an abbreviation for"):

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b \ \underline{iaaf} \ (-a_1)x_1 + (-a_2)x_2 + \ldots + (-a_n)x_n \leq -b$$
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n < b \ \underline{iaaf} \ \neg(a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b)$$
$$a_1x_1 + a_2x_2 + \ldots + a_nx_n > b \ \underline{iaaf} \ \neg(a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b)$$

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n = b \ \underline{iaaf} \ \left| \begin{array}{c} (a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b) \\ \wedge\, (a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b) \end{array} \right.$$

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \ \boldsymbol{R} \ a'_1x_1 + a'_2x_2 + \ldots + a'_nx_n + b$$

$$\underline{iaaf} \ (a_1 - a'_1)x_1 + (a_2 - a'_2)x_2 + (a_n - a'_n)x_n \ \boldsymbol{R} \ b \qquad (\boldsymbol{R} \in \{<, \leq, \geq, >, =\})$$

$$\alpha \rightarrow \beta \ \underline{iaaf} \ \neg\alpha \vee \beta$$

$$\alpha \leftrightarrow \beta \ \underline{iaaf} \ (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$$\left. \begin{array}{r} x \ \underline{iaaf} \ x = \mathtt{T} \\ \neg x \ \underline{iaaf} \ x = \mathtt{F} \end{array} \right| \text{ for a variable } x \text{ such that } \tau(x) = \mathbb{B}$$

Semantics. An *Interpretation* $\mathcal{I}$ is a mapping of every variable x to a value $\mathcal{I}(x) \in \tau(x)$. For an interpretation $\mathcal{I}$ and a formula α, the relation $\mathcal{I} \models \alpha$ ($\mathcal{I}$ satisfies α) is defined inductively as follows:

- $\mathcal{I} \models a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$ if $a_1\mathcal{I}(x_1) + a_2\mathcal{I}(x_2) + \ldots + a_n\mathcal{I}(x_n) \leq b$;
- $\mathcal{I} \models x = v$ for $\tau(x)$ be a finite set if $\mathcal{I}(x) = v$;
- $\mathcal{I} \models \neg\alpha$ if $\mathcal{I} \not\models \alpha$;
- $\mathcal{I} \models \alpha \wedge \beta$ if $\mathcal{I} \models \alpha$ and $\mathcal{I} \models \beta$;
- $\mathcal{I} \models \alpha \vee \beta$ if $\mathcal{I} \models \alpha$ or $\mathcal{I} \models \beta$.

The set of all interpretations is denoted by Ω and $\mathcal{M}$ is a mapping defined by $\mathcal{M}(\alpha) = \{\mathcal{I} \in \Omega \mid \mathcal{I} \models \alpha\}$ for any formula α of Olaaaf's formalism. Then, the notions related to deductive reasoning (entailment, (in)satisfiability, contradiction, etc.) are defined as in Section 2.2.

The running example can be formalized in the Olaaaf formalism with the following naming conventions:

- Given a food type f, say kiwi, the fact that f is an ingredient of the recipe is represented by a Boolean variable whose name is just f (e.g. $\textit{kiwi}$), its mass in grams in the recipe is given by the real variable of name f_{g} (e.g. $\textit{kiwi}_{\mathrm{g}}$), its number of units is given by an integer variable of name f_{u} (e.g. $\textit{kiwi}_{\mathrm{u}}$).
- In a similar way, variables corresponding to volumes of ingredients can be defined in L (liter) or tbsp (tablespoon), e.g. $\textit{granulatedSugar}_{\mathrm{tbsp}}$.
- The Boolean variable $\textit{milkshake}$ states that this is a milkshake recipe.

The target problem $\mathtt{x}^{\mathtt{tgt}}$ and the retrieved case $\mathtt{c}^{\mathtt{s}}$ are represented as follows:

$$\begin{aligned} \mathtt{x}^{\mathtt{tgt}} &= \textit{milkshake} \wedge \textit{kiwi} \\ \mathtt{c}^{\mathtt{s}} &= \textit{milkshake} \wedge (\textit{banana}_{\mathrm{u}} = 2) \wedge (\textit{granulatedSugar}_{\mathrm{tbsp}} = 4) \\ &\quad \wedge (\textit{vanillaSugarBag}_{\mathrm{u}} = 2) \wedge (\textit{cowMilk}_{\mathrm{L}} = 1.) \wedge (\textit{iceCube}_{\mathrm{u}} = 4) \wedge \ldots \end{aligned}$$

The "..." indicates that for every ingredient not stated in the recipe, its mass is interpreted as 0 (e.g., $\mathtt{c}^{\mathtt{s}} \models (\textit{kiwi}_{\mathrm{g}} = 0)$). The domain knowledge considered in the example is $\mathtt{DK} = \mathtt{DK}_1 \wedge \mathtt{DK}_2 \wedge \mathtt{DK}_3 \wedge \mathtt{DK}_4 \wedge \mathtt{DK}_5 \wedge \mathtt{DK}_6 \wedge \mathtt{DK}_7$ where $\mathtt{DK}_1$ to

$\texttt{DK}_6$ corresponds to the DK1 to DK6 in Section 1 and $\texttt{DK}_7$ relates propositional variables and numerical variables.[3]

$$\texttt{DK}_1 = (\mathit{banana} \rightarrow \mathit{fruit}) \wedge (\mathit{kiwi} \rightarrow \mathit{fruit})$$

$$\texttt{DK}_2 = (\mathit{banana}_{\,\mathrm{g}} = 115\,\mathit{banana}_{\,\mathrm{u}}) \wedge (\mathit{granulatedSugar}_{\,\mathrm{g}} = 15\,\mathit{granulatedSugar}_{\,\mathrm{tbsp}})$$
$$\wedge\, (\mathit{kiwi}_{\,\mathrm{g}} = 100\,\mathit{kiwi}_{\,\mathrm{u}}) \wedge (\mathit{cowMilk}_{\,\mathrm{g}} = 1030\,\mathit{cowMilk}_{\,\mathrm{L}}) \wedge \ldots$$

$$\texttt{DK}_3 = (\mathit{fruit}_{\,\mathrm{g}} = \mathit{banana}_{\,\mathrm{g}} + \mathit{kiwi}_{\,\mathrm{g}} + \ldots)$$
$$\wedge\, (\mathit{milk}_{\,\mathrm{g}} = \mathit{almondMilk}_{\,\mathrm{g}} + \mathit{cowMilk}_{\,\mathrm{g}} + \mathit{soyMilk}_{\,\mathrm{g}})$$
$$\wedge \begin{pmatrix} \mathit{sweeteningPower}_{\,\mathrm{g}} & = 0.158\,\mathit{banana}_{\,\mathrm{g}} & + 0.0899\,\mathit{kiwi}_{\,\mathrm{g}} \\ & + 1.\,\mathit{granulatedSugar}_{\,\mathrm{g}} & + 0.98\,\mathit{vanillaSugar}_{\,\mathrm{g}} \\ & + \ldots & \end{pmatrix}$$

$$\texttt{DK}_4 = (\mathit{almondMilk} \vee \mathit{soyMilk} \vee \mathit{cowMilk}) \leftrightarrow \mathit{milk}$$

$$\texttt{DK}_5 = (\mathit{soyMilk} \vee \mathit{cowMilk}) \wedge \mathit{kiwi} \rightarrow \mathit{bitter}$$

$$\texttt{DK}_6 = (\mathit{milkshake} \rightarrow \mathit{dessert}) \wedge (\mathit{dessert} \rightarrow \neg \mathit{bitter})$$

$$\texttt{DK}_7 = (\mathit{banana} \leftrightarrow \mathit{banana}_{\,\mathrm{g}} > 0) \wedge (\mathit{cowMilk} \leftrightarrow \mathit{cowMilk}_{\,\mathrm{g}} > 0) \wedge \ldots$$

3.2 Using Olaaaf through an example

Once $\texttt{c}^s$, $\texttt{x}^{\texttt{tgt}}$, DK and the parameters of the distance used (see Section 3.3 below), with respective Python variable names `srce_case`, `tgt_problem` and `dk`, the adaptation can be computed by Olaaaf:

```
min_dist, tgt_case = adaptator.execute(srce_case, tgt_problem, dk)
```

where `min_dist` is the minimal distance between source and target cases and `tgt_case` corresponds to $\texttt{c}^{\texttt{tgt}}$.

With the running example, the computation time of Olaaaf took about 40 seconds on a simple laptop. An excerpt of the display of the result is as follows:

```
                cowmilk_L = 0            kiwi_u = 1
           almondMilk_L = 0.95       banana_u = 1
granulatedSugar_tbsp = 6
```

For the sake of readability, the lines of the variables whose values can be deduced from other variables, such as $\mathit{almondMilk}_{\,\mathrm{g}}$, and the variables whose values have not changed from the source case to the target case, such as $\mathit{iceCube}_{\,\mathrm{u}}$, have been removed. Furthermore, the order of the lines have been changed. This adaptation shows that 1 banana has been substituted with 1 kiwi, that cow milk has been substituted with almond milk (to avoid the bitterness of this dessert), and that the amount of some ingredients (in particular, granulated sugar) has been changed in order to preserve the sweetening power.

[3] Complete Python example available at `https://github.com/OlaaafEngine/Olaaaf/tree/main/examplesICCBR2024/example1.KiwiMilkshake.py`

Now, the result of this adaptation can be debated: some persons would expect to have all the bananas removed and replaced by kiwis. This result is consistent with the minimal change principle of the revision process, though: it has been required that there is some kiwi in the recipe (hence the minimal non null number of units of kiwis) but nothing is said about bananas. In order to achieve an adaptation removing all the bananas and replacing them with a similar amount of kiwis (in term of masses, while respecting the fact that the number of kiwis is an integer), one quick fix is to add the conjunct $\neg$*banana* to the target problem.[4] However, this may be an unsatisfactory way to solve the problem, as it appears to be an a posteriori adjustment of the target problem. Another way to formulate that all the bananas must be replaced could be to add a variable that counts the number of different fruit types, variable whose preservation would make a substitution of all the bananas by kiwis. More precisely, every variable x such that $\tau(x) = \mathbb{B}$ is associated with an integer variable *b2i_x* that is bound to 0 (resp., 1) whenever x is bound to `F` (resp., `T`): *b2i_* stands for "Boolean to integer". Then the following formula is added to the domain knowledge:[5]

$$\mathtt{DK}_8 = (\mathit{numberOfFruitTypes} = \mathit{b2i}\_\mathtt{banana} + \mathit{b2i}\_\mathtt{kiwi} + \ldots)$$

Both variations of the example give the same result:

```
kiwi_u = 2          banana_u = 0          (etc.)
```

3.3 Olaaaf's specification

Olaaaf takes as input the domain knowledge `DK`, a case $\mathtt{c}^{\mathtt{s}}$, a problem $\mathtt{x}^{\mathtt{tgt}}$ and returns a solved case $\mathtt{c}^{\mathtt{tgt}}$. It is based on revision-based adaptation (cf. section 2.4, equality (1)) and the revision operator used is distance-based, so, it is specified by a distance function $\mathtt{dist}_w^{\varepsilon}$ described below.

Distance functions between interpretations. Let $\mathcal{I}, \mathcal{J} \in \Omega$ and $x \in \mathcal{V}$. Let

$$\Delta_{\mathcal{I}\mathcal{J}} x = \begin{cases} \mathcal{J}(x) - \mathcal{I}(x) & \text{if } \tau(x) = \mathbb{Z} \text{ or } \tau(x) = \mathbb{R} \\ 0 & \text{if } \tau(x) \text{ is an enumerated type and } \mathcal{I}(x) = \mathcal{J}(x) \\ 1 & \text{if } \tau(x) \text{ is an enumerated type and } \mathcal{I}(x) \neq \mathcal{J}(x) \end{cases}$$

Now, let w be a mapping from $\mathcal{V}$ to the positive real number: w is called a *weighting function*. For $x \in \mathcal{V}$, $w(x)$ is denoted by w_x. The distance function $\mathtt{dist}_w$ is defined as follows, for $\mathcal{I}, \mathcal{J} \in \Omega$:

$$\mathtt{dist}_w(\mathcal{I}, \mathcal{J}) = \sum_{x \in \mathcal{V}} w_x \cdot |\Delta_{\mathcal{I}\mathcal{J}} x| \tag{2}$$

[4] `https://github.com/OlaaafEngine/Olaaaf/blob/main/examplesICCBR2024/example2.KiwiMilkshakeNoBanana.py`

[5] `https://github.com/OlaaafEngine/Olaaaf/blob/main/examplesICCBR2024/example3.KiwiMilkshakeSameNumberOfFruitTypes.py`

An issue with this distance function is that the revision operator $\dot{+}^{\mathtt{dist}_w}$ associated with it does not satisfy all the postulates of the AGM theory, in particular, the postulate stating that if μ is satisfiable then $\psi \dot{+} \mu$ has also to be satisfiable. For example if $x \in \mathcal{V}$ with $\tau(x) = \mathbb{R}$, $\psi = (x \leq 0)$ and $\mu = (x > 1)$, then there is no interpretation $\mathcal{J} \in \mathcal{M}(\mu)$ that is the closest one to $\mathcal{M}(\psi)$ according to $\mathtt{dist}_w$, due to the fact that $\mathcal{M}(\mu)$ is not a closed set of the metric space $(\Omega, \mathtt{dist}_w)$. Since this postulate is important for revision-based adaptation, another distance function $\mathtt{dist}_w^\varepsilon$ is defined as the upper approximation of $\mathtt{dist}_w$ to multiples of ε, where ε is a positive real number:

$$\mathtt{dist}_w^\varepsilon(\mathcal{I}, \mathcal{J}) = \varepsilon \cdot \left\lceil \frac{\mathtt{dist}_w(\mathcal{I}, \mathcal{J})}{\varepsilon} \right\rceil \qquad \text{(for } \mathcal{I}, \mathcal{J} \in \Omega\text{)}$$

where $\lceil r \rceil$ is the ceil function applied to $r \in \mathbb{R}$.[6]

Remark. According to the specification of distance-based revision, the output of revision is a satisfiable formula ϱ such that $\mathcal{M}(\varrho)$ may be infinite. By default, the adaptation process of Olaaaf provides only one model of ϱ, thus ϱ can be written as a conjunction of formulas of the form $x = v$ where $x \in \mathcal{V}$ and $v \in \tau(x)$, which is in general more readable than any formula ϱ. However, it is possible to ask for a formula ϱ with potentially many models, at the costs of an important additional computing time and a less readable result.

3.4 Algorithmic principles of Olaaaf

According to the specification of Olaaaf given below, Olaaaf consists in finding the solution(s) of the following optimisation problem:

$$\mathcal{I} \in \mathcal{M}(\mathtt{DK} \wedge \mathtt{c}^s) \tag{3}$$
$$\mathcal{J} \in \mathcal{M}(\mathtt{DK} \wedge \mathtt{x}^{\mathtt{tgt}}) \tag{4}$$
$$\text{minimize } \mathtt{dist}_w^\varepsilon(\mathcal{I}, \mathcal{J})$$

Such a solution is a pair $(\mathcal{I}^*, \mathcal{J}^*)$ and the output of Olaaaf is given by a formula equivalent to $\mathtt{DK} \wedge \mathtt{x}^{\mathtt{tgt}} \wedge \mathtt{y}^{\mathtt{tgt}}$ where $\mathtt{y}^{\mathtt{tgt}}$ is a proposed solution to $\mathtt{x}^{\mathtt{tgt}}$ and $\mathcal{J}^* \in \mathcal{M}(\mathtt{DK} \wedge \mathtt{x}^{\mathtt{tgt}} \wedge \mathtt{y}^{\mathtt{tgt}})$.

Technically, the above optimization problem is based on (a) transforming the formulas in the constraints (3) and (4), in particular, putting them in disjunctive normal form, and (b) calling a mixed integer linear optimization system.[7] A comprehensive version of Olaaaf's algorithm is given in [19].

3.5 Towards a methodology for choosing the weights of $\mathtt{dist}_w^\varepsilon$

The appropriate choice of weights for similarity measures for retrieval is a long-standing issue in CBR research (see e.g. [29]). A similar issue occurs here: how

[6] Since $\mathtt{dist}_w^\varepsilon$ is a discrete distance function, every subset of Ω is a closed set of $(\Omega, \mathtt{dist}_w^\varepsilon)$.

[7] We have used the scipy wrapper for HiGHS [16], `https://highs.dev`.

can the weights w_x be adequately chosen? This issue deserves some future work, though we have applied two heuristic principles that led us to a choice of weights consistent with our expectation of the adaptation results of the examples without having to perform many trial and error tests. These heuristics are:

- Strongly weighting more "abstract" variables, e.g. $w_{\mathit{fruit}} \gg w_{\mathit{kiwi}}$.
- Giving a weight for variables associated to a unit conversion, which takes into account this conversion, e.g. since 1 liter of cow milk has a mass of 1030 g, so $w_{\mathit{cowMilk}_{\mathrm{L}}} = 1030\, w_{\mathit{cowMilk}_{\mathrm{g}}}$.

4 Discussion and Related Work

Building a general adaptation system such as Olaaaf requires studying the adaptation processes proposed in previous CBR works. Various types of adaptation have been studied in the past, in addition to the revision-based adaptation approach used in this article. Is Olaaaf able to integrate other adaptation approaches? Section 4.1 shows how Olaaaf could be helpful for generic CBR systems. Section 4.2 discusses how Olaaaf could take into account common adaptation approaches. Section 4.3 discusses this issue for various formalisms of case representation.

4.1 Adaptation in generic tools

For many years, the CBR community has developed and distributed some generic tools. In [28], Schulteis et al. compare five systems that are still currently available: CloodCBR, eXiT*CBR, jColibri, myCBR and ProCAKE. For these systems, many features are analyzed, including the reuse step feature. The null adaptation (aka adaptation by copy) is provided by all these systems; it consists simply in using as such the solution of the retrieved case as a solution to the target problem. However, even if some of these systems provide some other adaptation possibilities, these adaptation processes are still limited and could benefit from the addition of Olaaaf.

jColibri [25] computes ontology-based adaptations and adaptations based on numerical proportions. myCBR [4] uses adaptation rules. CloodCBR and ProCAKE use adaptation methods defined outside of these generic systems. The ontology-based adaptation of jColibri is similar to the one introduced in our running example, exploiting the hierarchical organization that can be expressed using Olaaaf logical expressions of the form $a \to b$ (where a and b are two propositional variables). Numerical proportions, as implemented in jColibri, are used to adapt numerical values exploiting dependencies within cases. For example, if the cost of a hotel 1 night is 80 €, the cost of 2 nights will be 160 €. This type of dependency can be expressed in Olaaaf according to a logarithmic change of variables (turning equalities of proportions into equalities of differences, which are linear constraints). Adaptation rules used, in particular, by myCBR are studied below in the context of Olaaaf.

4.2 Common adaptation approaches

Rule-based adaptation, used e.g. by myCBR and in the Taaable system [8], is a way to apply adaptation knowledge in the form of adaptation rules (that are acquired either from experts or using adaptation learning methods generally based on differences between source case pairs [14, 22]). Revision-based adaptation using such rules has already been studied in the past, in a propositional logic setting [23], but can it be applied with the Olaaaf formalism? Although answering precisely this question would require the specification of an adaptation rule formalism, an example can be used to give an idea of how Olaaaf could integrate such an adaptation rule. Let us consider the adaptation rule $\texttt{AR}$ informally defined by "In a salad dish recipe, 1 g of vinegar can be substituted by 0.5 g of lemon juice and 0.5 g of water, and reciprocally." Now, assume that the source case recipe $\texttt{c}^s$ contains 4 g of vinegar and that vinegar is forbidden in the target problem $\texttt{x}^{\texttt{tgt}}$ ($\texttt{c}^s \models_{\texttt{DK}} (\mathit{vinegar}_{\text{g}} = 4)$ and $\texttt{x}^{\texttt{tgt}} \models_{\texttt{DK}} \neg \mathit{vinegar}$). A first idea to use $\texttt{AR}$ to adapt $\texttt{c}^s$ would consist of modifying the distance $\texttt{dist}_w(\mathcal{I}, \mathcal{J})$ defined in equation (2), Section 3.3, by adding the following terms:

$$\begin{array}{rl} & w_{\texttt{AR}} \left| \Delta_{\mathcal{I}\mathcal{J}} \mathit{vinegar}_{\text{g}} + \Delta_{\mathcal{I}\mathcal{J}} \mathit{lemonJuice}_{\text{g}} + \Delta_{\mathcal{I}\mathcal{J}} \mathit{water}_{\text{g}} \right| \\ + & w_{\texttt{AR}} \left| \Delta \mathit{lemonJuice}_{\text{g}} - \Delta \mathit{water}_{\text{g}} \right| \end{array} \tag{5}$$

The addition of these terms is conditioned by the fact that the source case is a salad dish ($\texttt{c}^s \models_{\texttt{DK}} \mathit{saladDish}$). In this situation, if the weight $w_{\texttt{AR}}$ is large enough, these two terms are equal to 0 when $\texttt{dist}(\mathcal{I}, \mathcal{J})$ is minimal, so the result of Olaaaf adaptation will give 2 g of lemon juice and 2 g of water (or 2 additional grams of each, if there is already lemon juice or water in $\texttt{c}^s$).

Since adaptation rules are not (yet) integrated into Olaaaf, this has been performed in the current version of this engine by introducing two additional real variables AR_1 and AR_2 with weights both equal to $w_{\texttt{AR}}$, with the following additional domain knowledge:

$$\mathit{saladDish} \rightarrow \begin{pmatrix} (\mathit{AR}_1 = \mathit{vinegar}_{\text{g}} + \mathit{lemonJuice}_{\text{g}} + \mathit{water}_{\text{g}}) \\ \wedge\ (\mathit{AR}_2 = \mathit{lemonJuice}_{\text{g}} - \mathit{water}_{\text{g}}) \end{pmatrix}$$

Olaaaf adaptation using this additional piece of domain knowledge translating the adaptation rule $\texttt{AR}$ has given the expected result on an example of carrot and cabbage salad recipe.[8]

Now, beyond these examples, two issues remain. First, what is the language of adaptation rules that Olaaaf could manage in a similar way as in this example? Second, how would such adaptation rules interact? This second issue is related to the way adaptation rules can be composed (one after the other) and to the way they interact (when two adaptation rules applicable to the same adaptation problem are based on common variables). These complex issues are left for future work.

[8] `https://github.com/OlaaafEngine/Olaaaf/tree/main/examplesICCBR2024/example4.CarrotCabbageSalad.py`

Case-based adaptation, introduced in [17] and reused further in [9], consists in applying CBR to adaptation, using *adaptation cases*, which capture previous successful adaptation episodes. One might consider adaptation cases as specific adaptation rules (at least in some applications). Thus, the use of adaptation cases in Olaaaf might be related to the previous issue of integrating adaptation rules into this engine, though this issue deserves much more work to be studied in a proper way.

Differential adaptation [13] is a methodology for acquiring adaptation knowledge as well as a numerical adaptation approach. It relies on linear relations between the variations of the problem variables and the variations of the solution variables. Using the notation of the current article, with $\mathtt{x}_1, \ldots, \mathtt{x}_m$ problem variables, and $\mathtt{y}_\ell$ a solution variable, it relies on pieces of adaptation knowledge given by linear constraints of the form $\Delta_{\mathcal{TJ}}\mathtt{y}_\ell = \sum_{k=1}^{m} a_{k\ell}\Delta_{\mathcal{TJ}}\mathtt{x}_k$ (with $a_{k\ell}$, numerical constants). This is similar to the example given above of the adaptation rule `AR` and this can be managed in a similar way. Actually, it has been shown in [6] how such piece of adaptation knowledge can be taken into account by a revision-based adaptation approach in a setting such as the Olaaaf formalism.

4.3 Adaptation in various case representation formalisms

Adaptation in process-oriented CBR (PO-CBR) systems. In PO-CBR systems, cases represent processes. The most common representation of process cases in PO-CBR is the formalism of workflows. For example, in [21], adaptation cases are used to adapt workflows. Unfortunately, no obvious way has been found (yet) to translate workflows into the Olaaaf formalism. Now, in [11], revision-based adaptation is studied in qualitative algebra formalisms, and one of these algebras, $\mathcal{INDU}$, is used to represent and adapt process cases (this has been applied to the adaptation of recipe preparations within the Taaable system). Since there are translations of qualitative algebras into propositional logic that can be used for belief revision and other belief change operations (see [7]), and since Olaaaf formalisms contains propositional logic, adapting processes represented in a qualitative algebra can theoretically be computed by Olaaaf.

Geometric adaptation. The CADRE system [15], aims at assisting computer-aided design, using CBR, to adapt architectural plans. In this system, the goal is to optimize a set of constraints that represent different geometric views. The variables used to express the constraints are continuous numerical variables. However, some of the constraints are quadratic (hence nonlinear). Therefore, the current approach of Olaaaf is insufficient to fully simulate the adaptations of CADRE. Taking into account some nonlinear constraints within Olaaaf is a future work.

Textual adaptation. Textual CBR is CBR where cases are (at least partially) composed of texts in a natural language. When textual cases are handled directly, there is little hope that Olaaaf can be used for their adaptations. By contrast,

when textual cases are handled through a structured representation, then Olaaaf might be used to adapt them. For example, in Taaable, textual preparations of recipes have been formalized using the qualitative algebra $\mathcal{INDU}$ (see above) and the adaptation is made at that representation level and then applied to the text of the source case. Other textual CBR systems use structured representation associated with texts (see e.g. [30]) and thus, a revision-based adaptation for them might be usable in these contexts.

5 Conclusion

This article has presented the first version of Olaaaf, an adaptation engine for CBR in a formalism that contains both attribute-value pair representations for cases and taxonomies for domain knowledge (and beyond). It should be noted that Olaaaf uses the available domain and adaptation knowledge: if there is little of this knowledge, then it behaves as a knowledge-light approach, otherwise it behaves as a knowledge-intensive one. The generality of Olaaaf has been discussed in Section 4. This work contributes to the development of generic tools for CBR, specifically, for a phase of CBR whose development is often little considered or considered ad hoc.

One strength of CBR compared to more fashionable AI approaches (deep learning, LLMs) is that it requires much less electrical energy and much less data than these approaches. On the downside, designing a CBR system is often more complex than using such a heavy machine learning approach, which often requires only to pick up some tools "from the shelf". Therefore, the development of general systems for CBR or for some CBR steps, such as adaptation in Olaaaf, contributes to reducing this weakness of CBR and thus making it more attractive, which should reduce the negative ecological impact of AI technologies.

A first direction of future work on Olaaaf is to reduce its computing time. For this purpose, some ongoing work aims to use optimization techniques from the field of algorithmics for logic. Another optimization approach will aim to circumscribe the set of variables required for a given adaptation problem.

Another direction of future work is the extension of Olaaaf. In particular, as explained in Section 4.2, we are looking now at ways to take into account adaptation rules, which can be seen as shortcuts in the case space. An example has shown that this idea is promising, though it requires some work to be adequately generalized. Another extension of Olaaaf is motivated by the discussion about the CADRE system (Section 4.3). In this system, some nonlinear constraints are used to represent cases, so the question is whether Olaaaf could be extended for a formalism with such constraints. From a theoretical point of view, as soon as an optimization system is available to solve optimization problems working with these kinds of constraints, this should be feasible, and this would further extend the scope of Olaaaf.

References

1. Aamodt, A., Plaza, E.: Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications **7**(1) (1994) 39–59
2. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the Logic of Theory Change: partial meet functions for contraction and revision. Journal of Symbolic Logic **50** (1985) 510–530
3. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM **26**(11) (1983) 832–843
4. Bach, K., Althoff, K.D.: Developing case-based reasoning applications using mycbr 3. In Agudo, B.D., Watson, I., eds.: Case-Based Reasoning Research and Development, Berlin, Heidelberg, Springer Berlin Heidelberg (2012) 17–31
5. Cojan, J., Lieber, J.: Conservative Adaptation in Metric Spaces. In: Advances in Case-Based Reasoning, 9th European Conference, ECCBR-2008, Trier, Germany. Proceedings. LNAI 5239 (2008) 135–149
6. Cojan, J., Lieber, J.: Applying Belief Revision to Case-Based Reasoning. In: Computational Approaches to Analogical Reasoning: Current Trends. Volume 548 of Studies in Computational Intelligence. Springer (2014) 133 – 161
7. Condotta, J.F., Kaci, S., Marquis, P., Schwind, N.: Merging qualitative constraints networks using propositional logic. In: European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Springer (2009) 347–358
8. Cordier, A., Dufour-Lussier, V., Lieber, J., Nauer, E., Badra, F., Cojan, J., Gaillard, E., Infante-Blanco, L., Molli, P., Napoli, A., Skaf-Molli, H.: Taaable: a Case-Based System for personalized Cooking. In Montani, S., Jain, L.C., eds.: Successful Case-based Reasoning Applications-2. Volume 494 of Studies in Computational Intelligence. Springer (2014) 121–162
9. Craw, S., Jarmulak, J., Rowe, R.: Learning and Applying Case-Based Adaptation Knowledge. In Aha, D.W., Watson, I., eds.: Case-Based Reasoning Research and Development — Fourth International Conference on Case-Based Reasoning (ICCBR-01). LNCS 2080 (2001)
10. Dufour-Lussier, V., Hermann, A., Le Ber, F., Lieber, J.: Belief revision in the propositional closure of a qualitative algebra. In: 14th International Conference on Principles of Knowledge Representation and Reasoning, Vienna, Austria, AAAI Press (July 2014) 4
11. Dufour-Lussier, V., Le Ber, F., Lieber, J., Martin, L.: Case Adaptation with Qualitative Algebras. In Rossi, F., ed.: International Joint Conferences on Artificial Intelligence (IJCAI-2013), Pékin, Chine, AAAI Press (August 2013) 3002–3006
12. Dufour-Lussier, V., Le Ber, F., Lieber, J., Martin, L.: Adapting Spatial and Temporal Cases. In Ian Watson, B.D.A., ed.: International Conference for Case-Based Reasoning. Volume 7466 of LNCS., Lyon, France, Amélie Cordier, Marie Lefèvre, Springer (September 2012) 77–91
13. Fuchs, B., Lieber, J., Mille, A., Napoli, A.: Differential adaptation: An operational approach to adaptation for solving numerical problems with CBR. Knowledge-Based Systems **68** (2014) 103 – 114
14. Hanney, K., Keane, M.T.: Learning Adaptation Rules From a Case-Base. In Smith, I., Faltings, B., eds.: Advances in Case-Based Reasoning – Third European Workshop, EWCBR'96. LNAI 1168, Springer Verlag, Berlin (1996) 179–192
15. Hua, K., faltings, B., Smith, I.: CADRE: case-based geometric design. Artificial Intelligence in Engineering **10**(2) (1996) 171–183

16. Huangfu, Q., Hall, J.A.J.: Parallelizing the dual revised simplex method. Mathematical Programming Computation **10**(1) (2018) 119–142
17. Leake, D.B., Kinley, A., Wilson, D.: Acquiring case adaptation knowledge: a hybrid approach. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 1. AAAI'96, AAAI Press (1996) 684–689
18. Lieber, J.: Application of the Revision Theory to Adaptation in Case-Based Reasoning: the Conservative Adaptation. In: Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR-07). LNCS 4626. Springer, Belfast (2007) 239–253
19. Lieber, J.: Révision des croyances dans une clôture propositionnelle de contraintes linéaires. In Maudet, N., Zanuttini, B., eds.: Journées d'intelligence artificielle fondamentale, plate-forme intelligence artificielle, Rennes, France (June 2015)
20. Lieber, J., Nauer, E., Prade, H.: When Revision-Based Case Adaptation Meets Analogical Extrapolation. In: 29th International Conference on Case-Based Reasoning (ICCBR 2021). Volume 12877 of Lecture Notes in Computer Science book series (LNCS)., Salamanca (virtual), Spain (September 2021) 156–170
21. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards Case-Based Adaptation of Workflows. In: Proceedings of the 7th International Conference on Case-Based Reasoning. LNCS 4626, Belfast, Springer (2007) 421–435
22. Nauer, E., Lieber, J., d'Aquin, M.: Lazy Adaptation Knowledge Learning Based on Frequent Closed Itemsets. In: International Conference on Cased-Based Reasoning (ICCBR 2023). Volume 14141 of Lecture Notes in Computer Science., Aberdeen, United Kingdom, Springer Nature Switzerland (July 2023) 309–324
23. Personeni, G., Hermann, A., Lieber, J.: Adapting Propositional Cases Based on Tableaux Repairs Using Adaptation Knowledge. In Lamontagne, L., Plaza, E., eds.: Case-Based Reasoning Research and Development, Proceedings of ICCBR-2014. Number 8765 in Lecture Notes in Computer Science, Cork, Ireland, Springer International Publishing (September 2014) 390–404
24. Randell, D.A., Cui, Z., Cohn, A.G.: A spatial logic based on regions and connection. In Nebel, B., Rich, C., Swartout, W.R., eds.: 3rd Int. Conf. on Knowledge Representation and Reasoning (KR-92), Morgan Kaufmann (1992) 165–176
25. Recio-García, J.A., González-Calero, P.A., Díaz-Agudo, B.: jcolibri2: A framework for building case-based reasoning systems. Volume 79. (2014) 126–145 Experimental Software and Toolkits (EST 4): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT-3 2010).
26. Richter, M.M., Michael, M.: Knowledge containers. Readings in case-based reasoning. Morgan Kaufmann Publishers (2003)
27. Riesbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey (1989)
28. Schultheis, A., Zeyen, C., Bergmann, R.: An Overview and Comparison of Case-Based Reasoning Frameworks. In: 31th International Conference on Case-Based Reasoning (ICCBR 2023), Aberdeen, Springer (2023) 327–343
29. Stahl, A.: Learning Similarity Measures: A Formal View Based on a Generalized CBR Model. In: Case-Based Reasoning, Research and Development, 6th International Conference, on Case-Based Reasoning, ICCBR 2005, Chicago, IL, USA, August 23-26, 2005, Proceedings. (2005) 507–521
30. Upadhyay, A., Massie, S., Clogher, S. In: Case-Based Approach to Automated Natural Language Generation for Obituaries. (10 2020) 279–294