Amalgams: A Formal Approach for Combining Multiple Case Solutions

Santiago Ontañón and Enric Plaza

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain),
{santi,enric}@iiia.csic.es

Abstract. How to reuse or adapt past solutions to new problems is one of the least understood problems in case-based reasoning. In this paper we will focus on the problem of how to combine solutions coming from multiple cases in search-based approaches to reuse. For that purpose, we introduce the notion of *amalgam*. Assuming the solution space can be characterized as a generalization space, an amalgam of two solutions is a third sulution which combines as much as possible from the original two solutions. We define amalgam as a formal operation over terms in a generalization space. We also discuss how amalgams may be applied in search-based reuse techniques to combine case solutions.

1 Introduction

Case-based reasoning systems are based on the hypothesis that "similar problems have similar solutions", and thus they solve new problems by reusing or adapting solutions to past problems. However, how to reuse or adapt past solutions to new problems is one of the least understood problems in case-based reasoning. There are multiple open problems such as which is the knowledge required for adaptation and how to acquire it [15], the relation between solution reuse and case retrieval [13] or solution revision [9]. In this paper we will focus on one of such problems, namely how to reuse solutions coming from multiple cases.

The three most common approaches to reuse are: substitutional adaptation, transformational adaptation and generative adaptation [9]. In this paper we will focus on transformational adaptation techniques, and in particular in search-based approaches. In search-based approaches to reuse, the solution to a problem is seen as a description or term in a search space. The retrieved solution is seen as the starting point of a search process, which explores the search space by applying adaptation operators to the retrieved solution until a satisfactory solution to the problem at hand is found. These techniques are common, for instance, in CBR systems applied to planning domains [10]. These techniques require a starting point for the search process, which is typically the solution of the retrieved case. Imagine a trip planning CBR system that has to generate a trip to go from city A to city C, and has been able to retrieve a case with a solution to go from A to B and another case that has a solution to go from B to C. If search-based techniques are used, the starting point of the search will be one of those two solutions. But, clearly, the solution to the problem at hand could be generated by combining in some way the solutions in the two retrieved cases. Therefore, a better starting point would be one which combines both solutions.

This paper introduces the *amalgam* operation as a formal operation through which, given two terms, a third term which combines as much as possible from the two original terms is generated. We will then show how this operation can be used to define search-based approaches to reuse which can, in a natural way, reuse solutions from more than one case at a time. Our amalgam operation is applicable in situations where the solutions to be generated by a CBR system can be represented as terms in a generalization space, and where a *more general than* (subsumption) relation exists in such space.

The rest of the paper is organized as follows. Section 2 introduces and motivates, in an informal way, the notion of amalgam. Then, Section 3 presents the concepts of generalization space needed in Section 4, where the formal definition of amalgam is presented. Finally, Section 5 discusses how amalgams may be used in the CBR Reuse process. The paper closes with related work and conclusions.

2 The notion of Amalgam

The notion of amalgam can be conceived of as a generalization of the notion of unification over terms. Unification of two terms (or descriptions) *adds* the content of both terms into a new term, the unifier. Thus, if a term ϕ is a unifier of two other terms ($\phi = \psi_a \sqcup \psi_b$), then all that is true for one of these terms is also true for ϕ. For instance, if ψ_a describes "a red vehicle" and ψ_a describes "a German minivan" then their unification ϕ is the description "a red German minivan." Two terms are not unifiable when they possess contradictory information; for instance "a red French vehicle" is not unifiable with "a red German minivan" since being French and German at the same time is not possible.

The strict definition of unification means that any two descriptions with only one item with contradictory information cannot be unified. Now, imagine a scenario where two such descriptions have a large part of complementary information, which a CBR system would be interested in reusing: unification is not useful. Thus, what we would be interested in CBR is considering how to reuse their complementary information setting aside, at least momentarily, the contradictory elements.

This is the idea that we intend to formalize with the notion of *amalgam* of two descriptions (or terms). An amalgam of two terms is a term that contains parts from these two terms forming a new coherent description. For instance, an amalgam of "a red French vehicle" and "a German minivan" is "a red German minivan"; clearly there are always multiple possibilities for amalgams, since "a red French minivan" is another example of amalgam.

Given to terms ψ_a and ψ_b, their possible amalgams constitutes a set of terms, and we propose that Case Reuse can exploit this set as a search space of possible solutions achieved by combining (amalgamating) descriptions ψ_a and ψ_b.

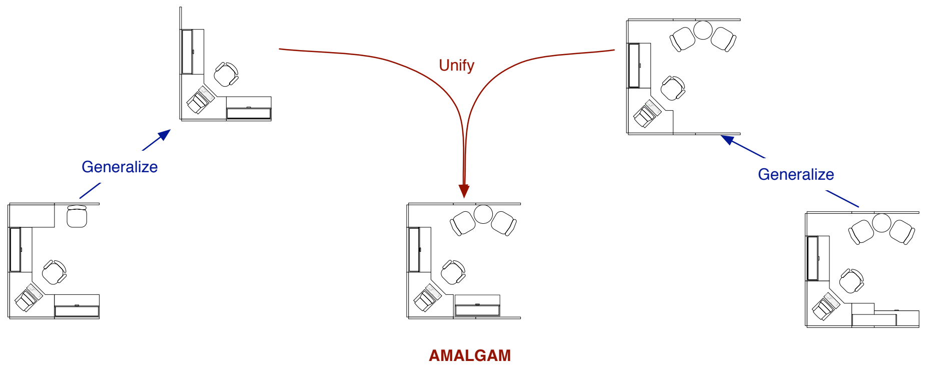


Fig. 1. An example of an amalgam of two cubicles.

In informal terms, since formalization will be presented later, let us consider the content of ψ_a with respect to ψ_b divided in two parts: C_a, and I_a, where C_a is the information that is compatible with ψ_b, and I_a is that content in ψ_a that is not compatible with the content in ψ_b. If, for any two terms ψ_a and ψ_b, we are able to identify their parts ($\langle C_a, I_a\rangle$ and $\langle C_b, I_b\rangle$) there is a clear way to reuse content from both into an amalgam: $C_a \sqcup C_b$, i.e. unifying what is compatible of both terms. Nevertheless, there is part of the content in I_a and I_b that can de reused and added to an amalgam: it's just that some cannot be used together (like "German" and "French" in vehicles). Thus, reusing content in I_a and I_b to form different *coherent* combinations will constitute the space of possible amalgams that can be built from two (non-unifiable) descriptions.

Figure 1 shows an example of an amalgam of two descriptions of cubicles, where the signature "M" shape of the amalgam is made clear: given two cubicles, each one is generalized to one of their possible generalizations, and then they are unified, yielding new cubicle that amalgams aspects of both original cubicles.

3 Generalization Space

In this paper we will make the assumption that solutions to problems are terms in some language $\mathcal{L}$, and that there exists a *subsumption* relation among terms.

We say that a term ψ_1 subsumes another term ψ_2, $\psi_1 \sqsubseteq \psi_2$ when ψ_1 is more general (or equal) than ψ_2[1]. Another interpretation of subsumption is that of an "information content" order: $\psi_1 \sqsubseteq \psi_2$ means that all the information in ψ_1 (all that is true for ψ_1) is also contained in ψ_2 (is also true for ψ_2).

The subsumption relation induces a partial order in the terms in a language $\mathcal{L}$, thus, the pair $\langle \mathcal{L}, \sqsubseteq\rangle$ is a *poset* for a given set of terms $\mathcal{L}$; additionally, we

[1] In machine learning terms, $A \sqsubseteq B$ means that A is more general than B, while in description logics it has the opposite meaning, since it is seen as "set inclusion" of their interpretations.

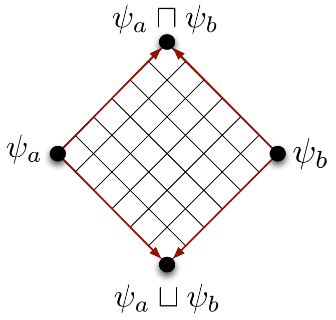


Fig. 2. A schema of the relationships between two terms ψ_a and ψ_b and their unification (below) and antiunification (above).

assume that $\mathcal{L}$ contains the infimum element $\perp$ (or "any"), and the supremum element $\top$ (or "none") with respect to the subsumption order. In the rest of this paper we will call the pair $\langle \mathcal{L}, \sqsubseteq \rangle$ *generalization space*.

Given the subsumption relation, for any two terms ψ_1 and ψ_2 we can define the *anti-unification* of two terms $(\psi_1 \sqcap \psi_2)$ as their *least general generalization*, representing to the most specific term that subsumes both. If two terms have nothing in common, then $\psi_1 \sqcap \psi_2 = \perp$. Thus, anti-unification encapsulates in a single description *all* that is shared by two given terms. Moreover, depending on the language $\mathcal{L}$, the anti-unification might be unique or not. Anti-unification is defined as follows:

$$\psi_1 \sqcap \psi_2 = \psi : (\psi \sqsubseteq \psi_1 \;\; \wedge \;\; \psi \sqsubseteq \psi_2) \wedge (\nexists \psi' \sqsupset \psi : \;\; \psi' \sqsubseteq \psi_1 \;\; \wedge \;\; \psi' \sqsubseteq \psi_2)$$

The dual operation to the anti-unification is that of *unification* $(\psi_1 \sqcup \psi_2)$, which is the *most general specialization* of a two given terms.:

$$\psi_1 \sqcup \psi_2 = \psi : (\psi_1 \sqsubseteq \psi \;\; \wedge \;\; \psi_2 \sqsubseteq \psi) \wedge (\nexists \psi' \sqsubset \psi : \;\; \psi_1 \sqsubseteq \psi' \;\; \wedge \;\; \psi_2 \sqsubseteq \psi')$$

That is to say, the unifier has the addition of the content of the two original terms (all that is true in ψ_1 and ψ_2 is also true in $\psi_1 \sqcup \psi_2$). However, not every pair of terms may be unified: if two terms have contradictory information then they have no unifier —which is equivalent to say that their unifier is "none": $\psi_1 \sqcup \psi_2 = \top$. Moreover, depending on the language $\mathcal{L}$, the unification of two terms, when exists, may be unique or not.

Figure 2 shows both anti-unification (the most specific common generalization), and unification (the most general common specialization). Moreover, unification and anti-unification might be unique or not depending on the structure induced by subsumption over the terms in a language $\mathcal{L}$.

A *refinement operator* $\rho : \mathcal{L} \rightarrow \wp(\mathcal{L})$ is a function that, given a term in $\mathcal{L}$, yields the set of direct (or minimal) generalizations or specializations of that term. In short, given a tern ψ, a refinement yields a generalization (resp. specialization) φ such that there is no generalization (resp. specialization) between ψ and φ in a language $\mathcal{L}$. Formally, the refinement relation is defined as:

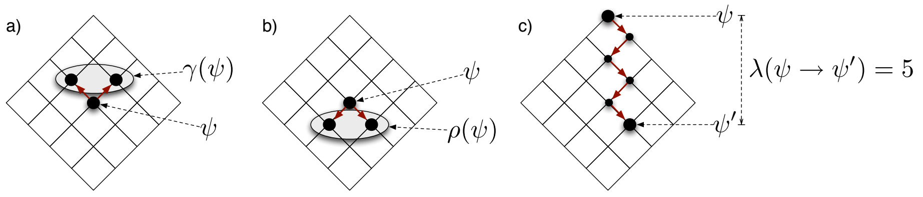


Fig. 3. Three schemas showing a generalization refinement operator γ, a specialization refinement operator ρ, and the distance between two terms.

Definition 1. *Two feature terms hold a refinement relation* $\psi_1 \prec \psi_2$ *iff*

$$\psi_1 \sqsubseteq \psi_2 \ \wedge \ \nexists \psi' : \psi_1 \sqsubset \psi' \sqsubset \psi_2$$

That is to say, the relation $\psi_1 \prec \psi_2$ holds when specializing from ψ_1 to ψ_2 is a minimal specialization step: there is no intermediate term ψ' in $\mathcal{L}$ between ψ_1 and ψ_2. Specifically, a specialization refinement operator is defined as follows:

$$\rho(\psi) = \{\psi' \in \mathcal{L} | \psi \prec \psi'\}$$

Whereas a generalization refinement operator is defined as follows:

$$\gamma(\psi) = \{\psi' \in \mathcal{L} | \psi' \prec \psi\}$$

Figure 3 illustrates the idea of both generalization and specialization refinement operators. Refinement operators can be used to navigate the space of terms using search strategies, and are widely used in Inductive Logic Programming [8]. In previous work, we made use of refinement operators to define similarity measures for CBR systems [11]. For instance, in the vehicle example used above, if we have a term representing "a german minivan", a generalization refinement operator would return generalizations like "an european minivan", or "a german vehicle". If we apply the generalization operator again to "an european minivan", we can get terms like "a minivan", or "an european vehicle". A specialization refinement operator would perform the opposite task, and given a term like "a german minivan", would return terms like "a Mercedes minivan", or "a red german minivan".

The length of the path between two terms ψ and ψ', noted as $\lambda(\psi \rightarrow \psi')$, is the number of times a refinement operator has to be used to reach ψ' from ψ. Moreover, the minimal path between two terms estimates their distance in the generalization space, as illustrated in Figure 3.c. Finally, the distance $\lambda(\perp \rightarrow \psi)$ measures the amount of information contained in one term [11].

4 Amalgams on a Generalization Space

This section formally defines the notion of amalgam over a generalization space. Let us introduce some auxiliary definitions first. First, when two terms are not unifiable, some of their generalizations, however, may be unifiable:

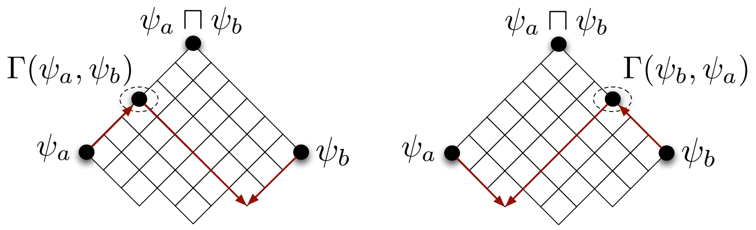


Fig. 4. A schema of the *most specific unifiable generalizations* $\Gamma(\psi_a, \psi_b)$ of term ψ_a with respect to ψ_b and $\Gamma(\psi_b, \psi_a)$ of term ψ_b with respect to ψ_a.

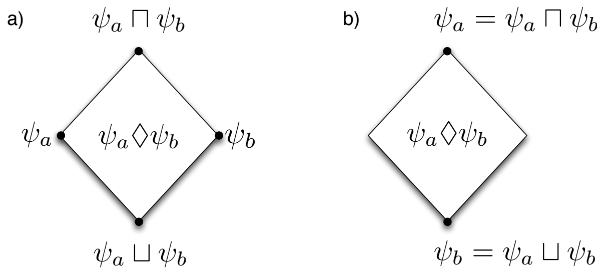


Fig. 5. Two schemas of the *interstice* space $\psi_a \lozenge \psi_b$ between two terms ψ_a and ψ_b: a) the general schema and b) the special case where $\psi_a \sqsubseteq \psi_b$.

Definition 2. *The set G of* unifiable generalizations *of a term ψ_a with respect to another term ψ_b is: $G(\psi_a, \psi_b) = \{\psi \in \mathcal{L} | \psi \sqsubseteq \psi_a \wedge \psi \sqcup \psi_b \neq \top\}$.*

Moreover, we are interested in the most specific unifiable generalizations in G. That is to say the terms in G that are greatest lower bounds of G.

Definition 3. **(MUG)** *The set Γ of* most specific unifiable generalizations *(mug) of term ψ_a with respect to ψ_b is defined as:*

$$\Gamma(\psi_a, \psi_b) = \{\psi \in G(\psi_a, \psi_b) | \nexists \psi' \in G(\psi_a, \psi_b) : \psi' \sqsubset \psi\}$$

In other words, the most specific unifiable generalizations of a term ψ_a with respect to another term ψ_b is the set of most specific generalizations of ψ_a which unify with ψ_b. Notice that if they are unifiable ($\psi_a \sqcup \psi_b \neq \top$) then $\Gamma(\psi_a, \psi_b) = \{\psi_a\}$. Figure 4 illustrates this idea.

Definition 4. **(Interstice)** *The* interstice *between two unifiable terms ψ_a and ψ_b, is the set of terms such that:*

$$\psi_a \lozenge \psi_b = \{\psi \in \mathcal{L} | (\psi_a \sqcap \psi_b) \sqsubseteq \psi \ \wedge \ \psi \sqsubseteq (\psi_a \sqcup \psi_b)\}$$

that is to say, the set of terms that are in the $\lozenge$-shaped space (shown in Fig. 5.a) defined by ψ_a and ψ_b (metaphorically "right and left"), and their antiunification

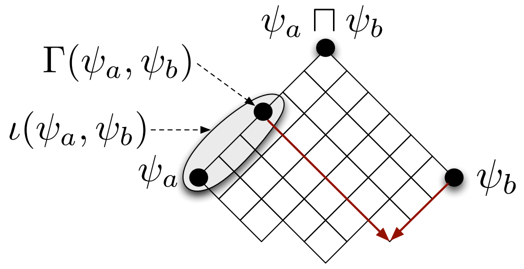


Fig. 6. Schema showing the sets of amalgamable generalizations $\iota(\psi_a, \psi_b)$ of a term ψ_a with respect to a term ψ_b.

$\psi_a \sqcap \psi_b$ unification $\psi_a \sqcup \psi_b$ (metaphorically "up and down"). Notice that when ψ_a and ψ_b are not unifiable then $\psi_a \sqcup \psi_b = \top$ and there is not strict limit below ($\top$ denotes failure to unify and is the term subsumed by all terms in $\mathcal{L}$).

A particular case of interest is the interstice between two terms ψ_a and ψ_b when $\psi_a \sqsubseteq \psi_b$ (see Figure 5.b): this interstice contains all the terms ψ such that $\psi_a \sqsubseteq \psi \sqsubseteq \psi_b$ — i.e. all terms between ψ_a and ψ_b in the generalization space.

Out of all the terms in the interstice, we are interested in those that contain the maximum amount of information from ψ_a and ψ_b, which we will call the amalgam space.

Definition 5. *The set of generalizations of a term ψ_a* amalgamable *with another term ψ_b is the set*

$$\iota(\psi_a, \psi_b) = \bigcup_{\varphi_a \in \Gamma(\psi_a, \psi_b)} \varphi_a \Diamond \psi_a$$

that is to say, for each *mug* of ψ_a with respect to ψ_b there is an interstice between ψ_a and this mug; the terms in these interstices are amalgamables, and their union is the set of all amalgamable generalizations of a term ψ_1 with another term ψ_2. Figure 6 illustrates this idea where the mug of ψ_a with respect to ψ_b is a set of a single term, and where we can see that the terms in $\iota(\psi_a, \psi_b)$ corresponds to the terms in the interstices between ψ_a and the mug, i.e. in the paths going from ψ_a to the mug.

Definition 6. (Amalgam) *The* amalgams *of two terms ψ_a and ψ_b is the set of terms such that:*

$$\psi_a \Upsilon \psi_b = \{\phi \in \mathcal{L} | \exists \varphi_a \in \iota(\psi_a, \psi_b) \wedge \exists \varphi_b \in \iota(\psi_b, \psi_a) : \phi = \varphi_a \sqcup \varphi_b\}$$

We call *amalgamation operation* the function $\Upsilon : \mathcal{L} \times \mathcal{L} \rightarrow \wp(\mathcal{L})$ that deterines the set of amalgam terms for any pair of terms, and we call *amalgam space* of two terms the part of the generalization space that contains their amalgams. Thus, a term ϕ is an *amalgam* of ψ_a and ψ_b if ϕ is the unification of two terms φ_a and φ_b that are amalgamable with ψ_b and ψ_a respectively. Figure 7 illustrates this idea,

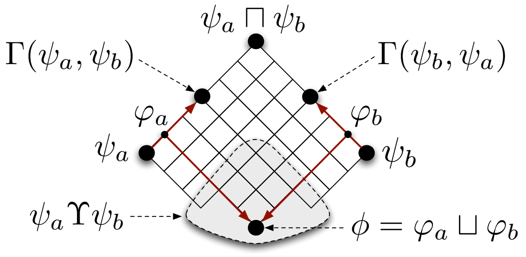


Fig. 7. Schema showing (in grey) the space of the *amalgams* $\psi_a \Upsilon \psi_b$ between two terms ψ_a and ψ_b.

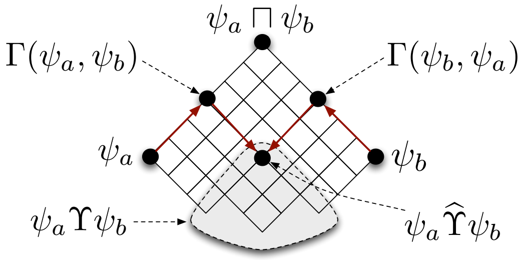


Fig. 8. Schema showing the set of upper bounds $\psi_a \widehat{\Upsilon} \psi_b$ of the space of *amalgams* $\psi_a \Upsilon \psi_b$ (in grey) between two terms ψ_a and ψ_b.

showing as a grey area the space of amalgams and one point in that space as the amalgam corresponding to $\phi = \varphi_a \sqcup \varphi_b$. In the special case where two terms are unifiable ($\psi_a \sqcup \psi_b \neq \top$), then unifiers and amalgams coincide: $\psi_a \Upsilon \psi_b = \psi_a \sqcup \psi_b$.

Next we will define the upper bounds of a space of amalgams:

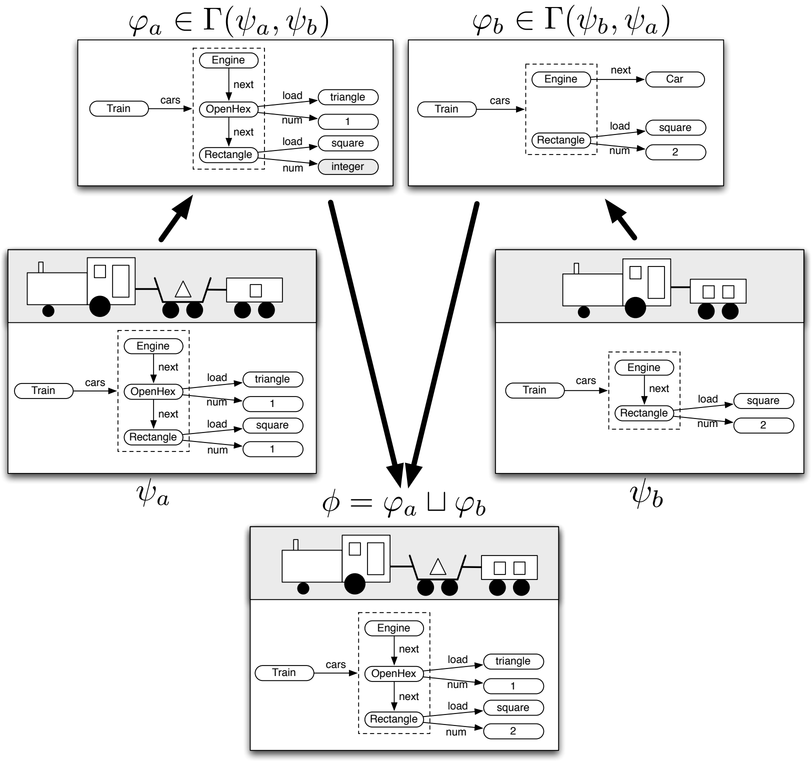
Definition 7. *The set of* upper bounds $\psi_a \widehat{\Upsilon} \psi_b$ *of an amalgam space* $\psi_a \Upsilon \psi_b$ *is the minimum set such that* $\forall \phi \in \psi_a \Upsilon \psi_b,\ \exists \phi' \in \psi_a \widehat{\Upsilon} \psi_b : \phi' \sqsubseteq \phi$.

The set of upper bounds can be determined as follows:

$$\psi_a \widehat{\Upsilon} \psi_b = \{\phi \in \psi_a \Upsilon \psi_b | \exists \varphi_a \in \Gamma(\psi_a, \psi_b) \wedge \exists \varphi_b \in \Gamma(\psi_b, \psi_a) : \phi = \varphi_a \sqcup \varphi_b\}$$

That is to say, given two terms, ψ_a and ψ_b, the set of pair-wise unifications of terms in their mugs $\Gamma(\psi_a, \psi_b)$ and $\Gamma(\psi_a, \psi_b)$ produces the set of upper bounds of the space of amalgams. Figure 8 shows the upper bounds of two terms in the special case where their mugs are unique.

Figure 9 shows an example where two trains, ψ_a and ψ_b are represented using feature terms [2]. Each train is represented as a set of cars, where the first one is an engine. Each car can have a load, which is represented as a load type (triangles or squares), and the number of items as an integer. The two trains do not unify, since the first car after the engine in one of them is an open hexagon, and it is a rectangle in the other one. Additionally, the number of items



$\varphi_a \in \Gamma(\psi_a, \psi_b)$
$\varphi_b \in \Gamma(\psi_b, \psi_a)$
ψ_a
ψ_b
$\phi = \varphi_a \sqcup \varphi_b$
Train
cars
Engine
next
OpenHex
Rectangle
load
num
triangle
square
1
2
integer
Car

Fig. 9. Amalgam between two trains, represented using a feature terms.

do not match. However, the description of the first train can be generalized by removing the restriction that the number of items loaded in the second car is 1 (resulting in term φ_a), and the second train can be generalized by removing the restriction that the rectangle car is the next car after the engine (resulting in term φ_b). After that, the two generalizations unify in a term ϕ, which is an amalgam of both original trains. Notice that the resulting amalgam has features of both trains: ϕ has two cars after the engine, like the first train, but the load of the rectangle car is two squares, like in the second train.

The next section discusses how amalgams can be used for Reuse in CBR.

5 CBR Reuse through Amalgams

In this section we will discuss how the amalgam operation can be used for reuse. We'd like to emphasize that amalgam is not proposed as a complete reuse technique, but as a piece that can be used inside of CBR reuse techniques.

Search in the solution space is an approach to case reuse which is commonly used for tasks where the solution is a comple structure, such as in configuration [16] or planning tasks [1]. CBR systems which use search for reuse typically

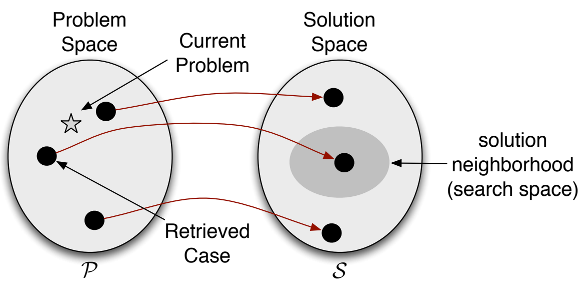


Fig. 10. Search-based case reuse: a single case is retrieved, and its solution neighborhood is explored using search.

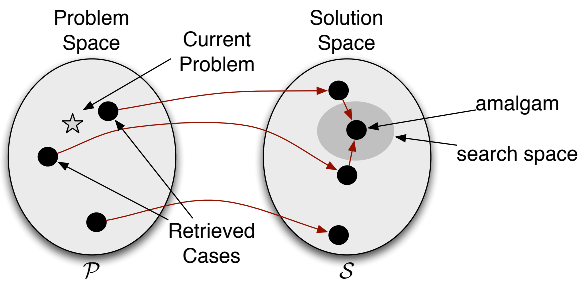
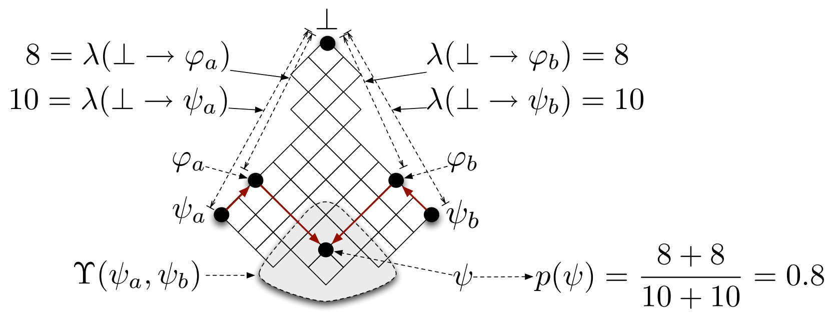


Fig. 11. Reusing the solutions of two cases by amalgamation plus search.

retrieve a single case, and use the solution in that case as the starting point of a search process in the solution space. The main idea is that it is expected that the solution to the problem at hand will be close to the solution in the retrieved case. Figure 10 illustrates this idea, where given a target problem, the most similar case (according to a similarity in the problem space) is retrieved, and then the neighborhood of the solution of the retrieved case is explored using search.

When using search as a reuse strategy, we need both a strategy to (systematically) explore the solution space in a neighborhood of the retrieved solution and a criteria to stop the search. In our framework we will model this two elements: a) refinement operators (defined in Section 3) define a systematic way to traverse the solution space, b) we will assume the existence of a predicate, $v(P, S)$, which given a problem and a candidate solution, assesses whether the solution is valid for the problem at hand: $v : \mathcal{P} \times \mathcal{S} \rightarrow \{true, false\}$, where $\mathcal{P}$ is the problem space, $\mathcal{S}$ is the solution space, and $v(P, S) = true$ when S is a valid solution for P. For example, in a configuration task, such predicate would return true when a particular solution is a valid configuration and false when it is an invalid or incomplete configuration; in a planning domain, it would return true when the solution is a complete plan consistent with the problem specification.

Search-based approaches to reuse are, often, limited to reusing a solution from a single case. This is because search techniques require a starting point



$\perp$
$8 = \lambda(\perp \to \varphi_a)$
$10 = \lambda(\perp \to \psi_a)$
$\lambda(\perp \to \varphi_b) = 8$
$\lambda(\perp \to \psi_b) = 10$
φ_a
ψ_a
φ_b
ψ_b
$\Upsilon(\psi_a, \psi_b)$
ψ
$p(\psi) = \frac{8+8}{10+10} = 0.8$

Fig. 12. An schema of the preservation degree $p(\psi)$ of an amalgam ψ.

for the search process, which corresponds to the solution of the retrieved case. However, if we wanted to reuse solutions from more than one case, there is no clear starting point. Our proposal is to use an amalgam as the starting point to search for solutions to the problem at hand, as illustrated in Figure 11.

An underlying assumption of search-based reuse approaches is that it is preferable to preserve as much from the retrieved solution as possible. [2]. If we use amalgams to generate a starting point of the search, we need a way to measure how much information of the original solutions an amalgam still preserves. For that purpose, we define the *preservation degree* of an amalgam.

Definition 8. (Preservation Degree) *Given an amalgam $\psi \in \psi_a \Upsilon \psi_b$ which is the unification $\phi = \varphi_a \sqcup \varphi_b$ of two terms such that $\varphi_a \in \iota(\psi_a, \psi_b)$ and $\varphi_b \in \iota(\psi_b, \psi_a)$, the* preservation degree *$p(\psi)$ is defined as:*

$$p(\psi) = \frac{\lambda(\perp \to \varphi_a) + \lambda(\perp \to \varphi_a)}{\lambda(\perp \to \psi_a) + \lambda(\perp \to \psi_a)}$$

where $\lambda(\psi \to \psi')$ is the number of times a refinement operator has to be used to reach ψ' from ψ —i.e. the distance between ψ and ψ' in the generalization space.

The preservation degree is the ratio of information preserved in the amalgam ψ with respect to the information present in the original terms ψ_a and ψ_b. The information preserved is measured by the addition of the information contained in the two amalgamable terms φ_a and φ_b which result in the amalgam.

As shown in Figure 12, the preservation degree is high when ψ_a and ψ_b had to be generalized very little in order to obtain the amalgam ψ. In other words, if the λ-distances between ψ_a and φ_a and between ψ_b and φ_b are low, preservation is high. If ψ_a and ψ_b had to be generalized a lot before finding amalgamable generalizations, then the preservation degree of the resulting amalgam will be low. In other words, the higher the λ-distances between ψ_a and φ_a and between ψ_b and φ_b, the lower the preservation degree.

[2] Although this assumption is only true for conservative adaptation techniques.

$$\begin{tikzcd}
 & & \psi_a \arrow[dl, "\psi_b"'] \arrow[dr, "\psi_c"] & & \\
 & \Upsilon(\psi_a, \psi_b) \ni \psi_a^1 \arrow[dl] \arrow[d] & & \psi_a^2 \in \Upsilon(\psi_a, \psi_c) \arrow[d] \arrow[dr] & \\
\ldots & \ldots & & \ldots & \ldots
\end{tikzcd}$$

Fig. 13. The search space defined by the process of amalgamating a tentative solution ψ_a with other solutions from cases in the case base.

Using the previous notions, we can define a basic way of reusing the solutions ψ_a and ψ_b from two cases to solve a particular problem P in the following way:

1. Compute the amalgam $\psi \in \psi_a \Upsilon \psi_b$ with the highest preservation degree.
2. Search in the neighborhood of ψ (the neighborhood can be searched using refinement operators) until we find a solution ψ^* such that $v(P, \psi^*) = true$.

This section has presented amalgam as a solution to reusing solutions from multiple cases in the context of search-based reuse. We have focused our explanation on reusing solutions from two cases, but the techniques generalize easily to reusing three or more solutions. The amalgam operation can be extended to amalgamate more than two terms in an easy way.

A different scenario could be where a CBR system initially finds a tentative solution ψ_a and then, transforms this solution by adding elements from another solution ψ_b. For instance, let us consider the first cubicle of Fig. 1 as the initial solution, and let us imagine there are certain aspects of the second cubicle that we would like to add to the initial solution: the result is the "amalgam cubicle" in Fig. 1. If we iterate this process we have a search proces, as shown in Figure 13. Starting from ψ_a, the initial solution, we can amalgamate ψ_a with ψ_b (or with ψ_c) obtaining a new solution ψ_a^1 (resp. ψ_a^2). Next, we can iterate amalgamation over ψ_a^1 and ψ_a^b, obtaining a search tree as shown in Fig. 13.

Thus, the amalgam operation provides a theoretical framework for combining case solutions, which can be used inside of case reuse strategies in different ways, only a few of which we have discussed here. Finally, recall that the only assumption required for working with amalgams is that the solution space of a CBR system can be expressed as a generalization space. Tasks such as configuration or planning naturally suit this framework. And even classification tasks could trivially fit our framework.

6 Related Work

The notion of amalgam is closely related to that of *merge* in the formalism of fluid construction grammar (FCG) [14] in that both relax or generalize the notion of unification. The merge operation is defined in FGC, a formalism in the family of unification-based grammars [6, 5] used in computational linguistics. Specifically, FGC has been proposed as a framework for language emergence and evolution, a process in which new structures are created or invented. This goal requires more flexibility than the operations used in unification-based grammars, which leaded to FGC defining operations like merge and extended form of unification.

The *merge* operation is informally defined as follows: "[...] merging a source expression s and a pattern expression p means changing the source expression such that it unifies with the pattern expression" [14]. That is to say, $merge(p, s)$ is a an operation that finds a changed source expression s' and yields as result their unification ($p \sqcup s'$). Notice that merge is asymmetric, since the two terms being merged are distinguished: one is changed (because it is the source) while the other remains unchanged (because it is the pattern). The amalgam of two terms ψ_1 and ψ_2, however, is symmetric: both terms are in equal standing, both are "changed" by generalization, eliminating some parts, in order to obtain an amalgam by the unification of those generalizations.

Merging operators have also been studied in *belief merging* [7], where the goal is to merge two knowledge bases (beliefs plus integrity constraints) while maintaining consistency. This approach was applied to CBR [3] by viewing case combination in reuse as a belief merging problem. Specifically, each case is viewed as a knowledge base, and the merging is generating a new knowledge base that preserves the relevant integrity constraints. Integrity constraints play a similar role to our validity predicate. However, this approach focuses on metric spaces, whereas our amalgam operation focuses on generalization spaces, and conceiving amalgam as a relaxation of the unification operation. Moreover, CBR techniques for adapting case solutions are also relevant to this paper, since the main research goal for introducing amalgams is to provide a formalization, as a search process, of reusing past cases' solutions to build a new solution adapted to the current problem. We will discuss several reuse techniques that can be understood as search processes over the solution space: local search, Abstract-Refine, compositional adaptation, and plan adaptation.

Compositional adaptation are reuse techniques that find new solutions from multiple cases, which were analyzed for configuration tasks in [16], where the approach *adaptation-as-configuration* is also presented. This approach has two specific operators (compose and decompose) that achieve compositional adaptation in "conceptual hierarchy oriented configuration." *Compose* merges multiple concept instances that have already been configured, while *Decompose* gives the subconcept instances of a given concept instance. These operations work upon *is-a* and *part-of* relations in an object-oriented hierarchy. The notions of amalgam and interstice space are intended to formalize the process of combining parts of different case solutions in a more abstract, domain-independent way. We do not assume that the task is configuration or that an object-oriented hierarchy is present, only that solutions are in a generalization space. Moreover, amalgams allows us to characterize the solution space of CBR as a search space for the purposes of combining multiple case solutions into a new solution for the current problem.

The Abstract-Refine approach to case-based planning [1] performs reuse in two steps: first a description is abstracted (not generalized), and second the abstract description is refined in search of an adapted description. The main difference with our approach is that Abstract-Refine starts from one description in the solution space and then explores the neighborhood of that description, while the starting point using amalgams is a term which is a combination of the solution in more than one case.

Finally, local search uses a taxonomy of classes to perform a Generalize-Refine process in search for an adapted solution and, as before, the main difference from our approach is that it explores the neighborhood of one description. SPA (Systematic Plan Adapter) [4] is an approach for adapting plans as search in a refinement graph of plans. SPA is systematic and complete and, as local search, is based on adapting a single plan. MPA (Multi-Plan Adapter) [12] is an extension which allows for reusing multiple plans. MPA breaks the different plans into smaller pieces, which are then recombined together. The complexity breaking these plans into smaller pieces, however, is very high and MPA uses only a heuristic. Compared to MPA, our approach avoids the need of this previous process of breaking down while providing a systematic way to combine multiple solutions into a single one.

7 Conclusions

This paper has presented a new operation between terms called *amalgam*. This operation can be seen as a relaxation of unification. An amalgam of two terms is a new term which contains as much information from the two original terms as possible. This new term is constructed by generalizing the two original terms as little as possible, and then unifying them. Moreover, we have presented how can the amalgam between two solutions be used for multiple solution reuse in CBR systems. The framework presented in this paper is domain-independent, and does not assume anything about the task the CBR system has to solve other than being able to express the solution as a term in a generalization space.

CBR systems typically only exploit similarity in the problem space, but domains where solutions are complex structures could also benefit from similarity in the solution space. The amalgam operation presented in this paper explores such idea, since the amalgam space defines a subset of terms in the solution space which are similar not to one but to two given solutions. We are already working on a technique implementing the amalgam operator for feature terms, but it's out of the scope of this paper.

As future work, we'd like to continue exploring the idea of using similarity relations in the solution space. More specifically, we plan to explore the applications of the amalgam operation to develop search-based techniques for CBR Reuse for specific tasks like configuration and hierarchical planning. Finally, amalgamating more than two solutions is also part of future work, since amalgamating more than two terms increases the number of constraints and, thus, enlarges the search space for the the techniques implementing amalgamation.

Acknowledgements. This research was partially supported by project Next-CBR (TIN2009-13692-C03-01).

References

[1] Ralph Bergmann and Wolfgang Wilke. Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research (JAIR)*, 3:53–118, 1995.

[2] Bob Carpenter. Typed feature structures: an extension of first-order terms. In V. Saraswat and K. Ueda, editors, *Proceedings of the International Symposium on Logic Programming*, pages 187–201, San Diego, 1991.

[3] Julien Cojan and Jean Lieber. Belief merging-based case combination. In *Case-Based Reasoning Research and Development (ICCBR'09)*, volume 5650 of *Lecture Notes in Artificial Intelligence*, pages 105–119, 2009.

[4] Steve Hanks and Daniel S. Weld. A domain-independent algorithm for plan adaptation. *J. Artificial Intelligence Research*, 2(1):319–360, 1994.

[5] Ray Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution.* Oxford University Press, 2002.

[6] Martin Kay. Functional unification grammar: a formalism for machine translation. In *Proc. 10th Int. Conf. on Computational Linguistics*, pages 75–78, Morristown, NJ, USA, 1984. Association for Computational Linguistics.

[7] S. Konieczny, J. Lang, and P. Marquis. Da2 merging operators. *Artif. Intell.*, 157(1-2):49–79, 2004.

[8] Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming. Techniques and Applications.* Ellis Horwood, 1994.

[9] R. Lopez De Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M. L. Maher, M. T. Cox, K. Forbus, M. Keane, A. Aamodt, and I. Watson. Retrieval, reuse, revision and retention in case-based reasoning. *Knowl. Eng. Rev.*, 20(3):215–240, 2005.

[10] Héctor Muñoz-Avila and Michael Cox. Case-based plan adaptation: An analysis and review. *IEEE Intelligent Systems*, 23:75–81, 2007.

[11] Santiago Ontañón and Enric Plaza. On similarity measures based on a refinement lattice. In *Case-Based Reasoning Research and Development (ICCBR'09)*, volume 5650 of *Lecture Notes in Artificial Intelligence*, pages 240–255, 2009.

[12] Ashwin Ram and Anthony Francis. Multi-plan retrieval and adaptation in an experience-based agent. In David B. Leake, editor, *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* AAAI Press, 1996.

[13] Barry Smyth and Mark T. Keane. Adaptation-guided retrieval: questioning the similarity assumption in reasoning. *Artificial intelligence*, 102:249–293, 1998.

[14] Luc Steels and Joachim De Beule. Unify and merge in fluid construction grammar. In *Symbol Grounding and Beyond*, volume 4211 of *Lecture Notes in Computer Science*, pages 197–223. Springer-Verlag, 2006.

[15] Wolfgang Wilke and Ralph Bergmann. Techniques and knowledge used for adaptation during case-based problem solving. In *In 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA-98*, pages 497–506. Springer, 1998.

[16] Wolfgang Wilke, Barry Smyth, and Padraig Cunningham. Using configuration techniques for adaptation. In *Case-Based Reasoning Technology*, volume 1400 of *Lecture Notes in Computer Science*, pages 139–168. Springer-Verlag, 1998.