Evaluation of $RECIDE_{PSY}$ - An Adviser in the Domain of Psychology

Carlos Bento Penousal Machado Ernesto Costa
Laboratório de Informática e Sistemas
Qta da Boavista, lote 1, 1
3000 Coimbra - PORTUGAL
bento@alma.uc.pt machado@alma.uc.pt ernesto@moebius.uc.pt

Abstract

This paper introduces $RECIDE_{PSY}$ a case-based adviser system in the domain of psychology. $RECIDE_{PSY}$ has a library of success and failure cases. Cases of failure are in the form of indivisible and incompatible cases and represent constraints in the generation of a new solution from successful cases in memory. We present the experimental results obtained with $RECIDE_{PSY}$. These results show that indivisible and incompatible cases reduce the number of successful cases necessary to cover the domain and decrease the number of wrong solutions given by the system.

Introduction

A Case-Based Reasoning (CBR) system depends strongly on its methods for retrieval and reuse of previous experiences. This contrasts with systems that rely on the generalization of solutions from first principles (abstract knowledge).

The combination of CBR and abstract knowledge-guided techniques led to the development of knowledge-based retrieval systems (Koton 1989). These systems use domain knowledge for constructing explanations of why a problem had a specific solution in the past. These explanations are necessary to judge the relevance of the facts describing a past problem (Bento & Costa 1993; Veloso 1992; Cain, Pazzani & Silverstein 1991; Barletta & William 1989).

In our work on CBR we are mainly concerned with two aspects. One is that the CBR approach is mostly used when a strong theory is not available and past experience is accessible. Lack of a strong theory generally means that explanations in cases are imperfect. We consider three kinds of imperfections in case explanations and use them for retrieval (Bento & Costa 1993). A second aspect relates to the role of cases of failure in CBR. Some current CBR systems make use of them to represent and explain unsuccessful experiences (Berger & Hammond 1991; Hammond 1986).

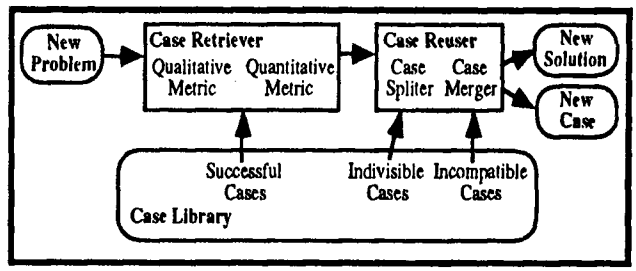

Figure 1: Functional structure of $RECIDE_{PSY}$

In our approach, cases of failure are of two types, indivisible and incompatible, and represent, respectively, *intra* and *inter*-case dependencies that were violated during case reuse. These failures are considered for the generation of new solutions.

We describe how cases of failure are used in an expert system called $RECIDE_{PSY}$ (Reasoning with Cases Imperfectly Described and Explained in the Domain of Psychology).

We present results on the system's performance with successful cases alone and with the two types of cases of failure used by $RECIDE_{PSY}$. It is shown that indivisible and incompatible cases reduce the number of successful ones necessary to cover the domain and reduce the number of wrong solutions given by the system. A comparison with other systems is provided.

Overview of $RECIDE_{PSY}$

$RECIDE_{PSY}$ functional structure comprises: a case retriever, and a case reuser (Figure 1). The case retriever accesses successful cases in the case library. For case selection it uses a qualitative and a quantitative metric. The qualitative metric (Bento & Costa 1994) clusters past cases by their potential usefulness for creation of a new solution. The quantitative metric (Bento & Costa 1993) orders cases in each cluster by similarity between them and the new problem. The case reuser receives case clusters ordered by decreasing

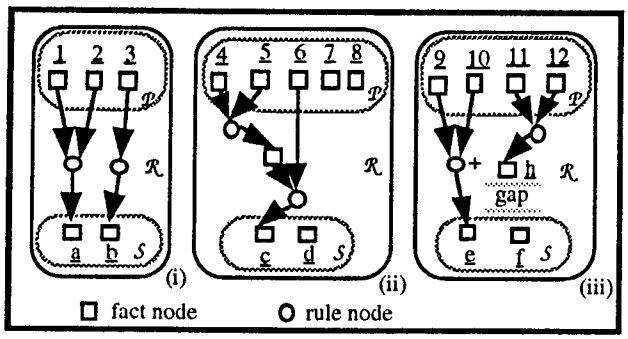

Figure 2: A case with (i) a complete set of explanations; (ii) an incomplete set of explanations; (iii) a partial and a broken explanation

similarity and generates a new case that potentially has the same solution as the new problem. New cases are generated through application of splitting and merging operators, constrained by indivisible and incompatible cases.

Case Library

The case library comprises: successful, indivisible, and incompatible cases. A successful case is represented by a triple $\langle \mathcal{P}, \mathcal{S}, \mathcal{R} \rangle$ (Figure 2) with $\mathcal{P}$ and $\mathcal{S}$, respectively, a set of facts representing past problem and solution, and $\mathcal{R}$ a set of rules representing a causal justification. The causal justifications can be viewed as a set of presumably imperfect causal explanations. An explanation is a proof tree that links facts in the problem with a fact in the solution. We consider three kinds of imperfections in explanations: (1) incomplete set of explanations; (2) partial explanations; (3) broken explanations.

In a successful case with an incomplete set of explanations some solution facts are not explained and hence are not the conclusion for any proof tree (e.g., Cases ii and iii , in Figure 2. Facts d and f in these cases' solution are not leaves of a proof tree). A partial explanation is one whose proof tree omits some branches. This means that one or more steps in the proof tree apply a rule for which the conditions are necessary but not sufficient. Rule nodes representing these rules are labeled by '+' (e.g., In Figure 2, case iii, the proof tree at the left). A broken explanation is one in which there is a gap between the proof tree and the case's solution (e.g., In Figure 2, case iii, the proof tree at the right).

As mentioned before, to generate of a new solution past cases are split and those parts considered potentially useful to solve the new problem are collected and merged into a new case that, hopefully, has a solution similar to the one of the new problem.

Indivisible cases represent case pieces that when split in the past conduced to the creation of cases comprising wrong solutions. Incompatible cases represent case pieces that led to wrong solutions and were created by merging pieces from several cases.

Indivisible and incompatible cases, are represented by a triple $\langle \mathcal{P}_f, \mathcal{S}_f, \mathcal{R}_f \rangle$ with $\mathcal{P}_f$ and $\mathcal{S}_f$ the sets of facts representing, respectively, problem and solution components, and $\mathcal{R}_f$ a set of rules. With $\langle \mathcal{P}, \mathcal{S}, \mathcal{R} \rangle$ representing the case being split for generation of a new one, the semantic for indivisible cases is:

1. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f = \emptyset$, $\mathcal{R}_f = \emptyset$, and $\mathcal{P}_f \subseteq \mathcal{P}$ then the subset $\mathcal{P}_f$ in $\mathcal{P}$ cannot be split.
2. If $\mathcal{P}_f = \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f = \emptyset$, and $\mathcal{S}_f \subseteq \mathcal{S}$ then the subset $\mathcal{S}_f$ in $\mathcal{S}$ cannot be split.
3. If $\mathcal{P}_f = \emptyset$, $\mathcal{S}_f = \emptyset$, $\mathcal{R}_f \neq \emptyset$, and $\mathcal{R}_f \subseteq \mathcal{R}$ then the subset $\mathcal{R}_f$ in $\mathcal{R}$ cannot be split.
4. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f \neq \emptyset$, and $\mathcal{P}_f \subseteq \mathcal{P} \wedge \mathcal{S}_f \subseteq \mathcal{S} \wedge \mathcal{R}_f \subseteq \mathcal{R}$ then subsets $\mathcal{P}_f$, $\mathcal{S}_f$, and $\mathcal{R}_f$ in $\mathcal{P}$, $\mathcal{S}$, and $\mathcal{R}$ have to remain in the same case piece after the splitting process.
5. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f = \emptyset$, and $\mathcal{P}_f \subseteq \mathcal{P} \wedge \mathcal{S}_f \subseteq \mathcal{S}$ then the subsets $\mathcal{P}_f$ and $\mathcal{S}_f$ in $\mathcal{P}$ and $\mathcal{S}$ have to remain in the same case piece after the splitting process.
6. If $\mathcal{P}_f = \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f \neq \emptyset$, and $\mathcal{S}_f \subseteq \mathcal{S} \wedge \mathcal{R}_f \subseteq \mathcal{R}$ then the subsets $\mathcal{S}_f$ and $\mathcal{R}_f$ in $\mathcal{S}$ and $\mathcal{R}$ have to remain in the same case piece after the splitting process.
7. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f = \emptyset$, $\mathcal{R}_f \neq \emptyset$, and $\mathcal{P}_f \subseteq \mathcal{P} \wedge \mathcal{R}_f \subseteq \mathcal{R}$ then the subsets $\mathcal{P}_f$ and $\mathcal{R}_f$ in $\mathcal{P}$ and $\mathcal{R}$ have to remain in the same case piece after the splitting process.

indivisible cases of types 1, 2, and 3 constrain the splitting of facts in $\mathcal{P}$, or $\mathcal{S}$, or $\mathcal{R}$. Types from 4 to 7 constrain splitting of pieces composed of subsets of $\mathcal{P}$, $\mathcal{S}$, and $\mathcal{R}$.

Incompatible cases represent merging constraints. With $\langle \mathcal{P}', \mathcal{S}', \mathcal{R}' \rangle$ representing the new case generated by merging past cases, the semantic for incompatible cases is:

1. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f = \emptyset$, $\mathcal{R}_f = \emptyset$ then $\mathcal{P}_f$ cannot occur in $\mathcal{P}'$ as a results of merging.
2. If $\mathcal{P}_f = \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f = \emptyset$ then $\mathcal{S}_f$ cannot occur in $\mathcal{S}'$ as a results of merging.
3. If $\mathcal{P}_f = \emptyset$, $\mathcal{S}_f = \emptyset$, $\mathcal{R}_f \neq \emptyset$ then $\mathcal{R}_f$ cannot occur in $\mathcal{R}'$ as a results of merging.
4. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f \neq \emptyset$ and $\mathcal{P}_f \subseteq \mathcal{P}' \wedge \mathcal{S}_f \subseteq \mathcal{S}' \wedge \mathcal{R}_f \subseteq \mathcal{R}'$ then $\mathcal{P}_f \wedge \mathcal{S}_f \wedge \mathcal{R}_f$ in the new case cannot be a result of merging.
5. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f = \emptyset$ and $\mathcal{P}_f \subseteq \mathcal{P}' \wedge \mathcal{S}_f \subseteq \mathcal{S}'$ then $\mathcal{P}_f \wedge \mathcal{S}_f$ in the new case cannot be a result of merging.
6. If $\mathcal{P}_f = \emptyset$, $\mathcal{S}_f \neq \emptyset$, $\mathcal{R}_f \neq \emptyset$ and $\mathcal{S}_f \subseteq \mathcal{S}' \wedge \mathcal{R}_f \subseteq \mathcal{R}'$ then $\mathcal{S}_f \wedge \mathcal{R}_f$ in the new case cannot be a result of merging.

7. If $\mathcal{P}_f \neq \emptyset$, $\mathcal{S}_f = \emptyset$, $\mathcal{R}_f \neq \emptyset$ and $\mathcal{P}_f \subseteq \mathcal{P}' \wedge \mathcal{R}_f \subseteq \mathcal{R}'$ then $\mathcal{P}_f \wedge \mathcal{R}_f$ in the new case cannot be a result of merging.

As with indivisible cases, incompatible ones of type 1, 2, and 3 relate to merging constraints at the fact level. Types from 4 to 7 relate to constraints at case's piece level.

Case Retrieval

Case retrieval is performed on a flat memory of successful cases. The retrieval process involves two steps:

1. Clustering of potentially useful cases.
2. Ranking of clustered cases.

In the first step, five clusters of past cases are created. Each cluster comprises cases for which it is believed their solution facts and the new solution facts verify a specific set relation.

Let $\mathcal{S}$ be the set of facts representing the solution for a case in memory and $\mathcal{S}'$ the set of facts representing the solution for a new problem. Each cluster is composed of the cases verifying the following conditions (in the examples it is assumed the case library is composed of cases in Figure 2):

CLUSTER No. 1 - Cases with $\mathcal{S} = \mathcal{S}'$.
e.g. If the new problem is described by the set of facts $\{\underline{1}, \underline{2}, \underline{3}\}$, cluster no. 1 will be composed of case i (Figure 2). Case i is completely explained, that is facts $\{\underline{1}, \underline{2}, \underline{3}\}$ describing case's problem are necessary and sufficient for the solution $\mathcal{S} = \{\underline{a}, \underline{b}\}$ and so the new problem's solution is $\mathcal{S}' = \mathcal{S} = \{\underline{a}, \underline{b}\}$.

CLUSTER No. 2 - Cases possibly with $\mathcal{S} = \mathcal{S}'$.
e.g. For a new problem described by $\{\underline{4}, \underline{5}, \underline{6}, \underline{7}, \underline{8}\}$, cluster no. 2 will be composed of case ii. As the new problem is similar to the one described in case ii it is possible that case and new problem's solutions are similar. The reason why we are not certain about this is that case ii is not completely explained. So, we do not know if facts $\underline{7}$ and $\underline{8}$ are causally linked with fact $\underline{d}$ in the solution. This means the problem that has the solution $\mathcal{S} = \{\underline{c}, \underline{d}\}$ may be different from the one represented in case ii provided it has facts $\underline{4}$, $\underline{5}$, and $\underline{6}$.

CLUSTER No. 3 - Cases possibly with $\mathcal{S} \supset \mathcal{S}'$.
e.g. Considering a new problem $\{\underline{1}, \underline{2}\}$, case i is the one in cluster no. 3. As $\underline{1}$ and $\underline{2}$ are the causal premises for fact $\underline{a}$ in this case's solution, it is possible that the new problem solution is $\{\underline{a}\} = \mathcal{S}' \subset \mathcal{S}$. The uncertainty about this is based on that unknown *intra*-case dependencies may be violated by splitting case i.

CLUSTER No. 4 - Cases possibly with $\mathcal{S} \subset \mathcal{S}'$.
e.g. With a new problem $\{\underline{1}, \underline{2}, \ldots , \underline{8}\}$, cases i and ii are the ones in cluster no. 4. As case i has the solution $\mathcal{S} = \{\underline{a}, \underline{b}\}$ for problem $\{\underline{1}, \underline{2}, \underline{3}\}$ and case ii's solution $\{\underline{c}, \underline{d}\}$ is supposed to be the one for problem $\{\underline{4}, \ldots , \underline{8}\}$ then it is possible that $\{\underline{a}, \underline{b}\} = \mathcal{S}_i \subset \mathcal{S}'$ and $\{\underline{c}, \underline{d}\} = \mathcal{S}_{ii} \subset \mathcal{S}'$, with $\mathcal{S}_i$ and $\mathcal{S}_{ii}$, respectively, the solutions for cases i and ii. We are not certain about this as dependencies between facts in $\mathcal{S}'$ are unknown.

CLUSTER No. 5 - Cases possibly with $\mathcal{S} \cap \mathcal{S}' \neq \emptyset$.
e.g. Assuming the new problem is $\{\underline{3}, \underline{4}, \underline{5}, \underline{6}, \underline{15}\}$, as $\underline{3}$ is necessary and sufficient for $\underline{b}$ in the context of case i this case is in this cluster. The same for case ii in which $\underline{4}$, $\underline{5}$, and $\underline{6}$ are the facts that causally explain $\underline{c}$ in the solution. In both cases it is possible that $\mathcal{S}_i \cap \mathcal{S}' = \{\underline{b}\}$ and $\mathcal{S}_{ii} \cap \mathcal{S}' = \{\underline{c}\}$. The uncertainty on this is similar to the one described for cluster no. 4.

These clusters are not necessarily disjoint. Dependencies in clusters no.s 2 and 3 can be viewed as *intra*-case dependencies. Dependencies in clusters no. 4 and 5 can be seen as *inter*-case dependencies.

Cases within each cluster are ranked by an explanation-based similarity metric (Bento & Costa 1993).

The retrieval procedure described above has two main properties: (1) case clustering organizes memory cases accordingly to their potential usefulness for the new problem's solution; and (2) it provides information on how to create a new case with a solution that is similar to the one for the new problem.

We do not have yet sufficient results to state that this retrieval method is better than one without the clustering step.

Case Reuse

Case reuse involves one of three procedures: (1) reuse, without modifications, of a past case that definitely or hopefully has the same solution as the new problem, (2) create a new case by splitting a case that seems useful for the solution of a new problem and eliminating case pieces that are assumed useless for the new solution, (3) create a new case by splitting and merging a set of past cases that, hopefully, have a solution or part of it that contributes to the new one.

The cluster of cases that is chosen for reuse determines the procedure that is applied. If cluster no. 1 is selected [1] , the case it contains is reused without any modification and the solution for it is the solution for the new problem. If cluster no. 2 is chosen its case with the highest similarity value is suggested as having the same solution as the new problem. In both situations procedure (1) is applied.

Cases in cluster no. 3 are expected to have a solution represented by a set of facts that contains the facts representing the solution for the new problem. For reuse, the system selects the case in the cluster with the highest similarity value. This case is split in a way that

[1] If cluster no. 1 is not empty then it is expected to contain only one case. If it has more than one case this means the same problem has multiple solutions.

is expected to remove the case parts that are useless for the solution of the new problem. The system removes the case's pieces that contain the subset $\mathcal{Q}$ of solution facts for which $\mathcal{S} - \mathcal{S}' = \mathcal{Q}$ with $\mathcal{S}$ the set of facts describing the case's solution and $\mathcal{S}'$ the set of facts representing the new problem's solution. Procedure (2) is used in those situations.

Reuse from clusters no. 4 or 5, containing cases that potentially contribute to the new problem's solution, involves catching the cases C_1, ..., C_n in the cluster for which it is supposed that $\mathcal{S}_1 \cup ... \cup \mathcal{S}_n = \mathcal{S}'$, with $\mathcal{S}_i$ the set of facts representing the solution for case C_i. Cases C_1, ..., C_n are split [2] and merged in order to create a new case with solution $\mathcal{S}_1 \cup ... \cup \mathcal{S}_n$. Procedure (3) is the one involved in this process.

RECIDE$_{PSY}$'s reuse unit favors the application of those clusters with lower indexes (cluster no. 1 over no. 2, ...). The reason to choose cluster no. 1 if it is not empty is obvious. It is the only cluster that has a case known to have the correct solution. The preference criterion for the other clusters is to pick the one that requires fewer splitting and merging operations for generation of a new solution. The more splitting and merging operations are performed, the more likely it is that an unknown *intra* or *inter*-case dependency is disregarded.

Indivisible and incompatible cases have a chief role in case reuse. Splitting and merging operations assume independence between the split and merged case's parts. This independence doesn't exist in general. Indivisible and incompatible cases represent *intra* and *inter*-case dependencies. Indivisible and incompatible cases are acquired interactively during problem-solving.

Putting All Together

Now we introduce how problem-solving and learning occurs in *RECIDE*$_{PSY}$. The set of steps performed since the input of a new problem until its solution (created by the system or provided by the user) is called an iteration. A working session with *RECIDE*$_{PSY}$ comprises a series of iterations.

An iteration entails the following steps: (1) retrieval; (2) reuse; (3) evaluation; and (4) knowledge acquisition. In the first step a new problem is input and the retrieval unit creates ranked clusters of cases in memory that potentially contribute to its solution. In the next step clusters are received by the reuse module that chooses one cluster applying the heuristics described in the preceding subsection. Then one of the reuse procedures is applied creating a case that potentially has the same solution as the new problem. In step 3 the system gives the user a new solution plus the case(s) in its origin. If she/he accepts the solution the iteration is complete. If the solution is not accepted, the user,

[2] Splitting is necessary as it is possible that some parts of the cases C_1, ..., C_n are useless for the new solution.

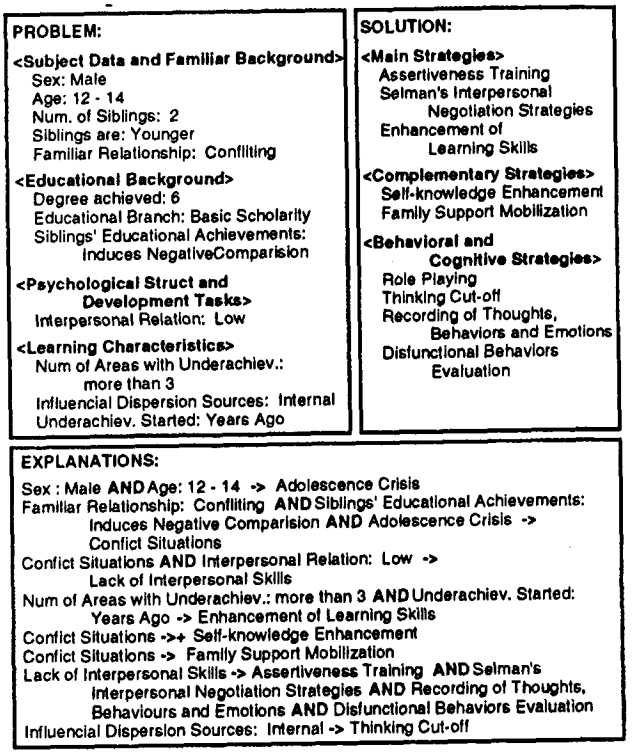

Figure 3: A case in the domain

after analyzing the cases used for its construction, describes the splitting and merging constraints that were violated and in the origin of this wrong answer. These constraints are input in terms of indivisible and incompatible cases. The correct solution and a causal justification (a new case) are also given to the system.

Test Domain

The tests we present in this paper are from *RECIDE*$_{PSY}$ an expert system in the domain of psychology. Its task is to suggest an intervention program for scholar underachievers.

A past experience comprises a context (past problem) in which a set of intervention strategies (past solution) was applied successfully. A context is described by four groups of facts (Figure 3): subject data and familiar background, educational background, psychological structure and development tasks, and learning characteristics. Intervention strategies are divided into three groups: main, complementary and behavioral and cognitive strategies. The domain vocabulary comprises 78 attributes for context description, and 101 intervention strategies.

Figure 3 represents a case in the domain as it is output by *RECIDE*$_{PSY}$. In explanations, a '$\rightarrow$' symbol, represents a complete explanation and a '$\rightarrow$+' symbol a partial explanation. This case describes a male client between 12 and 14 years old, with two siblings,

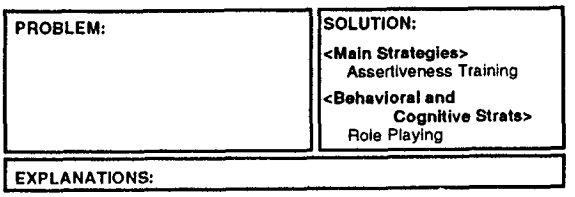

Figure 4: An indivisible case in the domain

both younger and with a conflicting relation with relatives. The level of education achieved is six (six years of basic education). He is unfavorably compared with his siblings due to their scholar achievements. Interpersonal relationship is low. His grades comprise more than three unsuccessful disciplines. He shows internal sources of dispersion and has a long history of underachievement.

The main intervention strategies that were applied were assertiveness training, Selman's interpersonal negotiation strategies, and enhancement of learning skills. The complementary strategies were self-knowledge enhancement and family support mobilization. The behavioral and cognitive set of intervention strategies were role playing, thinking cut-off, recording of thoughts, behaviors and emotions, and disfunctional behaviors evaluation.

The two former explanations provided by the experts for this intervention program were: (1) being a male client aged between twelve and fourteen are causing an adolescence crisis, and (2) a conflicting familiar relationship marked by negative comparison, associated with the adolescence crisis characterize a conflict situation.

In this task indivisible cases represent intervention strategies that cannot be split when they appear in the past case candidate for splitting (Figure 4). The indivisible case in Figure 4 states that "assertiveness Training" and "Role Playing" are two techniques that cannot be separated if they occur in a past case being reused. Incompatible cases represent the solutions given by the system that were classified as wrong. An example is shown in Figure 5. The solution component of the incompatible case in Figure 5 describes a wrong answer given by the system to the situation represented in the case's problem part.

The current domain library has 31 successful cases and 78 cases of failure (45 indivisible and 33 incompatible).

Experimental Results

Three kinds of tests (labeled TEST #1, #2, and #3) were performed. The set of successful cases used in these tests was randomly ordered and this ordering was maintained along all the experiments. In the evaluation step of each solving/learning iteration the solution given by the system was classified as correct, ambiguous, or wrong. If a solution contained more than 70%

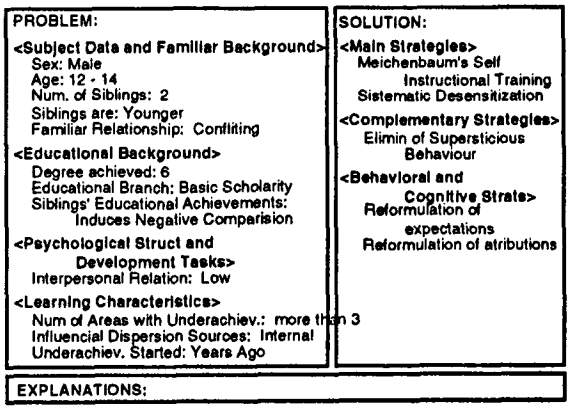

Figure 5: An incompatible case in the domain

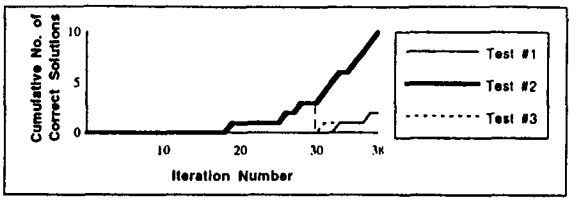

Figure 6: Cumulative number of correct solutions given by $RECIDE_{PSY}$

of the intervention strategies suggested by the expert and the number of unsuitable strategies in this solution were less than 30% of the total number of strategies suggested by the experts then it was classified as correct; a solution with a percentage of unsuitable strategies higher or equal to 30% was classified as wrong. Solutions with a percentage of suitable and unsuitable strategies out of these intervals were classified as ambiguous.

In TEST #1 only successful cases were given to the system in response to a wrong or ambiguous solution. In TEST #2, after a wrong or ambiguous solution had been suggested, the user input a set of indivisible cases and the correct solution plus her/his justification. In TEST #3, in response to a wrong or ambiguous solution, the user gave an incompatible case plus the correct case for the wrongly solved problem. Figures 6, 7, and 8 summarize the results obtained with these tests.

In TEST #1 the number of correct results (Figure 6) is quite small. Only in the 33th iteration was a correct solution suggested. TEST #3 did not provide much better results. The first correct solution occurred after the 31st iteration. In both tests only two situations were solved correctly. In TEST #2 the obtained results were different. The first correct result appeared after the 19th new problem and after the 31st iteration all the situations were solved correctly by the system, except in the 34th iteration in which the solution was wrong.

The cumulative number of wrong solutions (Fig-

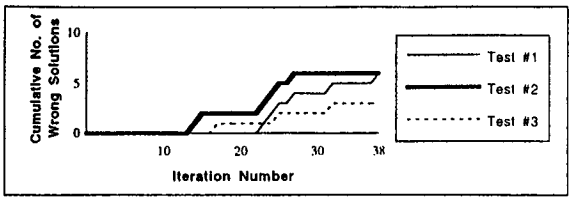

Figure 7: Cumulative number of wrong solutions given by $RECIDE_{PSY}$

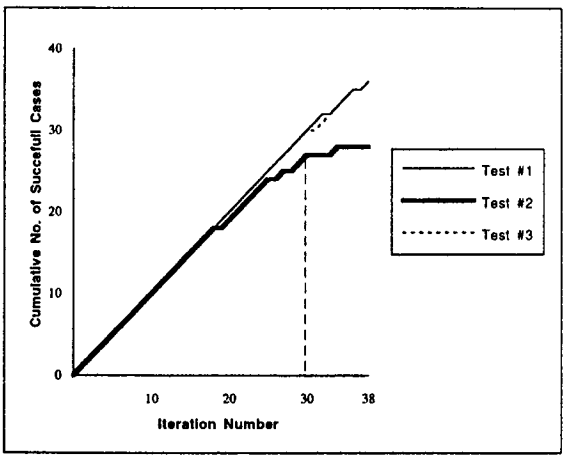

Figure 8: Cumulative number of successful cases given to $RECIDE_{PSY}$

ure 7) was peculiar in some aspects that we will discuss in the next section. With TEST #1 we did not have wrong solutions until iteration 23, after that the number increased rapidly until the 27th iteration and smoothly after that. In 38 iterations we had six wrong solutions. In TEST #2 wrong solutions were given earlier in the 14th iteration until a number of six. In TEST #3 a wrong solution occurred after the 17th iteration and the number of wrong solutions increased smoothly until a number of three in the 38th iteration.

In TEST #1 and #3, the number of cases input, in response to a wrong or ambiguous solution (Figure 8) increased approximately by the ratio 1/1 relatively to the number of iterations. In TEST #2 the evolution was similar to the other tests until the 24th iteration. Between the 30th and the 38th iteration the number of input cases only increased by one.

Analysis of Experimental Results

Considerations on the experimental results relate, at first, to the solutions (sets of intervention strategies) suggested by the system. From Figure 6, showing the cumulative number of correct solutions given by the system, it is clear that when we did not input indivisible cases describing dependencies between intervention strategies (TEST #1 and #3), the system's performance was quite bad. When indivisible cases were input (TEST #2) the performance changed rapidly after the 30th iteration. After that, the system gave always a correct solution except in the 34th iteration. This means the cases input until the 30th iteration provided a good coverage of the domain.

Another meaningful clue on system's performance is the number of wrong solutions it produced (Figure 7). The results from TEST #1 were as we expected. The system began to give some wrong results after iteration 23. The explanation for this is that before this iteration the number of cases in the system's library were not sufficient for it to provide a wrong solution and so the system kept giving ambiguous solutions. Sometimes, after this iteration, the system chose the wrong cases for splitting and merging and in consequence some wrong solutions were suggested. The results on wrong solutions in TEST #2 were odd. We did not expect to obtain wrong solutions in this test sooner than in TEST #1. The explanation we have for this is that some ambiguous solutions in TEST #2 were created from cases in memory that were erroneously retrieved. The information provided by indivisible cases caused bigger parts of these cases to be used for generation of a solution, turning it clearly into a wrong one in TEST #2. An interesting result was obtained with TEST #3. It is evident that input of incompatible cases disabled the repetition of wrong solutions suggested in the past. As we can see from Figure 7, after the 32nd iteration the number of wrong cases was 50% lower than in TEST #1 and #2.

Figure 8 shows that in TEST #1 and #3, until the 38th iteration, we had to give the system almost all the solutions and causal justification (cases). When indivisible cases are given (TEST #2) the scenario changes. The domain turns to be covered with a set of 31 cases.

Related Work

This work extends our previous view on reasoning on cases imperfectly described and explained implemented in CLASH (Bento & Costa 1993). It is also related with the work of Manuela Veloso (Veloso 1992) concerning to indexing driven by explanations and to the concept of footprint introduced by her and which we extend by considering three types of footprints.

A system which deals with cases of failure is CHEF (Hammond 1986). It stores information on why a meal plan did not work in the form of explanations of failure

ARCHIE-2 (Domeshek & Kolodner 1991) is an help system for architectural conceptual design. It keeps in memory information on failures in the form of design features that did not work in the creation of a conceptual design.

These two systems have in common the creation of new solutions from single cases and the use of information on failures.

Two other systems, CLAVIER (Barletta & Henessy 1989) and COOKIE (McCartney 1989), create a new case from multiple pieces of past cases.

CLAVIER is a shop floor assistant for autoclave loading. Its cases represent layouts of objects to be cured. It uses pieces of past cases for generation of new solutions. Adaptation comprises substitution and composition operations. This system uses information on unacceptable composition operations.

COOKIE is a planning system that generates new plans by splitting and merging old ones. The author of this system describes two types of difficulties in the adaptation process: (1) synchronization; (2) partial plan interaction. Synchronization relates to temporal constraints in merging multiple plans. It is assumed that, by default, partial plans do not interact.

Comparing RECIDE with CHEF, cases of failure within our framework are more 'case-based like' in the sense that we do not provide explanations for failures. Our cases of failure represent cases's components that are indivisible or incompatible.

The information used by ARCHIE-2 on design features that did not work can be represented within our approach by incompatible cases.

RECIDE, CLAVIER and COOKIE, all use case pieces for the generation of a new case. In general we defend that in systems that generate a new case by splitting a previous one or by splitting and merging a set of previous cases it is important to handle knowledge on *intra*-case dependencies (indivisible cases) and *inter*-case dependencies (incompatible cases). We believe this knowledge is frequently easier to acquire than failure explanations with the extra advantage that indivisible and incompatible cases are represented at the operational level.

Conclusions

We have provided evidence that indivisible and incompatible cases improve the performance of our case-based system. Indivisible cases lower the number of successful experiences needed for domain coverage. Incompatible cases lower the number of wrong solutions given by the system which is also a worthwhile result.

Also some final remarks must be made on the tests described in the paper. One is that it would be interesting to know the system's performance for a bigger set of new situations and, hopefully, to confirm that the domain is well covered with 31 cases when indivisible ones are provided.

A second question relates to coexistence of indivisible and incompatible cases. In our tests it has been shown that indivisible cases decrease the number of successful experiences necessary for domain coverage. Incompatible cases decrease the number of wrong solutions given by the system. It would be interesting to verify if coexistence of the two types of cases of failure sums up the benefits provided by each one alone.

A third aspect worthy of studying would be the effect of changing the input ordering of the new problems and cases.

Acknowledgments

We would like to thank Paula Vieira and Eduarda Góis who provided the case library, José Luis Ferreira and the anonymous reviewers for many useful comments, and the Luso-American Foundation and Fundacão Calouste-Gulbenkian for financial support.

References

Barletta, Ralph, and Mark, William 1989. Explanation-Based Indexing of Cases. In Proceedings of the Second Case-Based Reasoning Workshop. Morgan Kaufmann.

Barletta, Ralph, and Hennessy, Daniel 1989. Case Adaptation in Autoclave Layout Design. In Proceedings of the Second Case-Based Reasoning Workshop. Morgan Kaufmann.

Bento, Carlos, and Costa, Ernesto 1993. A Similarity Metric for Retrieval of Cases Imperfectly Described and Explained. In Preprints of the First European Workshop on Case-Based Reasoning (EWCBR-93). Univ. of Kaiserslautern, Germany.

Bento, Carlos, and Costa, Ernesto 1994. A Qualitative Approach for Retrieval of Cases Imperfectly Described and Explained, Technical Report, DEE-UC-001-94, Dept. de Eng. Electrotecnica, Univ. de Coimbra.

Berger, Jeffrey, and Hammond, Kristian 1991. Roengten: A Memory-based Approach to Radiation Therapy Treatment Design. In Proceedings of the Third Case-based Reasoning Workshop. Morgan Kaufmann.

Cain, Timothy, Pazzani, M. J. and Silverstein, Glenn 1991. Using Domain Knowledge to Influence Similarity Judgments. In Proceedings of the Third Case-Based Reasoning Workshop. Morgan-Kaufmann.

Domeshek, Eric, and Kolodner, Janet 1991. Toward a Case-based Aid for Architecture. *International Journal of Expert Systems* 4(2): 201-220.

Hammond, Kristian 1986. CHEF: A Model of Case-Based Planning. In Proceedings of AAAI-86. Cambridge, MA: AAAI Press / MIT Press.

Koton, Phyllis 1989. Using Experience in Learning and Problem Solving, Ph. D. diss, Laboratory of Computer Science, Massachusets Institute of Technology, MIT/LCS/TR-441.

McCartney, Robert 1993. Planning from Partial and Multiple Episodes in a Case-based Planner. In Proceedings of the Workshop on Case-based Reasoning of the National Conference on Artificial Intelligence. Technical Report WS-93-01. AAAI Press. Menlo Park, CA.

Veloso, Manuela 1992. Learning by Analogical Reasoning in General Problem Solving. Ph.D. diss., School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.