PARIS: Flexible Plan Adaptation by Abstraction and Refinement

Ralph Bergmann and Wolfgang Wilke
Centre for Learning Systems and Applications (LSA)
University of Kaiserslautern
Dept. of Computer Science
P.O.-Box 3049, D-67653 Kaiserslautern, Germany
e-mail: bergmann@informatik.uni-kl.de

1 Introducion

Paris (Plan Abstraction and Refinement in an Integrated System) [4, 2] is a domain independent case-based planning system which allows the flexible reuse of planning cases by abstraction and refinement. This approach is mainly inspired by the observation that reuse of plans must not be restricted to a single description level. In domains with a high variation in the problems, the reuse of past solutions must be achieved at various levels of abstraction.

In Paris, planning cases given at the concrete level are abstracted to several levels of abstraction which leads to a set of *abstract cases* that are stored in the case-base. This case abstraction is done automatically in the retain phase of the CBR-process modell [1]. When a new problem must be solved, an abstract case is retrieved whose problem description matches the current problem exactly at the abstract level. In the subsequent reuse phase, the abstract solution is refined, i.e., the details that are not contained in the abstract case are added to achieve a full solution of the problem. This refinement is done by a generative planner that performs a forward directed state space search. We now explain how abstract cases are constructed and reused in this case-based reasoning process. Figure 1 shows an overview of the whole system and its components. Besides case abstraction and refinement, PARIS also includes an explanation-based approach for *generalizing cases* during learning and for *specializing* them during problem solving. A complete description of the system can be found in [2].

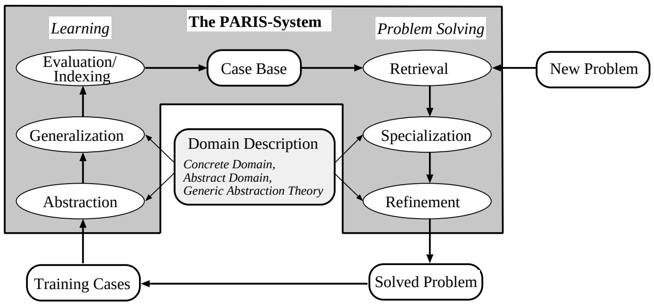

Figure 1. The Components of the PARIS-System

2 Case Abstraction

Case abstraction means reducing the level of detail contained in the problem description and in the solution of a case, i.e., an abstract case contains less operators and less states than the concrete case. Furthermore, abstract operators and states are described using more abstract terms which typically requires also a reduced number of predicates. In general, a change of the complete representation language between abstraction levels may be required to achieve meaningful und useful abstractions in a domain. Therefore, Paris assumes that in addition to the concrete planning domain, an *abstract planning domain* is contained as part of the general knowledge. This abstract domain may introduce a new representation language. It contains a set of *abstract operators* together with a set of rules that describe different ways of abstracting concrete states. For example in the domain of planning rotary symmetric workpieces the concrete domain contains opertors and predicates to describe the detailed contour of workpieces and individual cut operations that must be performed. The abstract domain abstracts from the detailed contour and represents larger units, called complex processing areas, together with the status of their processing (e.g. not processed, roughly processed, or completed).

Figure 2 presents an example of the relationship between a concrete case and an abstract case. The left side shows a section of a concrete case, depicting how a step-like contour with two grooves is manufactured by a sub-plan consisting of 6 steps. The abstract case, shown in the middle of this figure, abstracts from the detailed contour and just represents a complex processing area named A that includes raw and fine elements. The corresponding abstract plan contains 2 abstract steps: processing in a raw manner and processing in a fine manner. The arrows between the concrete and the abstract case show how concrete and abstract states correspond. Each abstract state is derived from one of the existing concrete states (state abstraction). However, not all concrete

ECAI 96 Workshop: Adaptation in CBR.
Edited by A. Voss et al.
Published in 1996 by John Wiley & Sons, Ltd.

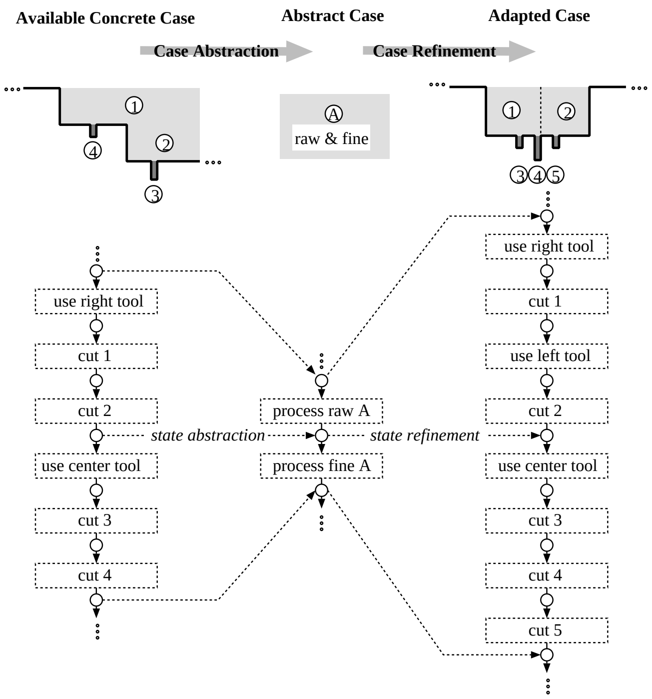

Figure 2. Example of generating and refining abstract cases.

states are abstracted. Some concrete states are skipped because they are considered a detail that should not be maintained in the abstract case. As a byproduct of this state abstraction, a sequence of concrete operators is abstracted to a single abstract operator. Note that the above explained kind of case abstraction is performed by an automatic procedure in PARIS as part of the retain-phase of the CBR-cycle. Abstract cases are then stored in the case base, indexed by an abstraction hierarchy (for details see [4]).

2.1 Retrieval and Reuse of Abstract Cases

When a new problem must be solved, an abstract case is retrieved which exactly matches the current problem at the abstract level. If several matching abstract cases are contained in the case-base, the retrieval procedure (using an abstraction hierarchy for indexing) selects the case that is located at the lowest level of abstraction. The motivation for this is that for more concrete cases, less work has to be done during refinement to achieve a complete solution. This corresponds to a similarity measure that ranks two cases more similar the lower the level of abstraction is, on which the problems are identical (see also [3]).

During the reuse phase, the abstract solution contained in the retrieved case must be refined to become a fully detailed complete solution. The right side of Figure 2 shows an example of such a refinement. While the contour of the two workpieces differs drastically at the concrete level, the abstract case matches exactly because the 5 atomic contour elements in the new problem can be abstracted to a complex processing area with raw and fine elements. The abstract opertors of the abstract case are then used to guide the state-based planner to find a refined solution to the problem. Therefore, each abstract state is used as a kind of sub-goal. The planner starts with the concrete initial state from the new problem description and searches for a sequences of concrete operators leading to a concrete state that can be abstracted to the first abstract state in the abstract case. The resulting operator sequence is a refinement of the first abstract operator. All remaining abstract operators are then sequentially refined in the same way. In the portion of the case shown in Figure 2, the abstract operator *process raw A* is refined to a sequence of four concrete steps which manufacture area 1 and 2. The next abstract operator is refined to a four-step sequence which manufactures the grooves 3, 4, and 5.

Note that PARIS allows the *reuse of problem decompositions* at different levels of abstraction. Abstract plans decompose the original problem into a set of much smaller subproblems. If these problems are small enough and mostly independent from each other, the underlying planner is able to solve them without stepping into the intractability problem.

3 Retrieving and Refining Abstract Cases

Now we assume a case-base which contains a set of abstract cases that can be reused to solve a new problem. During problem solving, an abstract case must be selected from a case-base, and the abstract plan contained in this case must be refined to become a solution to the new problem. During case retrieval we must search for an abstract case which is *applicable*, i.e. it contains a problem description that is an abstraction of the current problem. In order to decide whether an abstract problem description is an abstraction of the current problem in hand, we apply the generic abstraction theory already used during case abstraction. Thereby an abstract problem description of the new problem is inferred from the concrete problem description. The abstracted problem description can be matched against the abstract problem description contained in the abstract case.

If a matching case is found, it must be refined to come to a solution to the current problem. This refinement starts with the concrete initial state from the problem statement. A search is performed to find a sequence of concrete operations which leads to a concrete state that can be abstracted with the generic abstraction theory to match the second abstract state contained in the abstract case. If the first abstract operator can be refined a new concrete state is found. This state can then be taken as a starting state to refine the next abstract operator in the same manner. If this refinement fails we can backtrack to the refinement of the previous operator and try to find an alternative refinement. If the whole refinement process reaches the final abstract operator it must directly search for an operator sequence which leads to the concrete goal state of the new problem. If this concrete goal state has been reached the concatenation of concrete partial solutions leads to a complete solution to original problem.

This refinement demands for a search procedure which allows an abstract goal specification. All kinds of forward-directed search such as depth-first iterative-deepening or best-first search procedures can be used for this purpose. In PARIS we use depth-first iterative-deepening search.

Please note that PARIS allows the *reuse of problem de-*

compositions at different levels of abstraction. Abstract plans decompose the original problem into a set of much smaller subproblems. These subproblems are solved by a search-based problem solver. The problem decomposition leads to a significant reduction of the overall search that must be performed to solve the problem [7]. With pure search the worst-case time complexity for finding the required solution by search is $O(b^n)$, where n is the length of the solution and b is the average branching factor. If the problem is decomposed by an abstract solution into k subproblems, each of which require a solution of length $n_1, \ldots, n_k$, respectively, with $n_1 + n_2 + \cdots + n_k = n$, the worst-case time complexity for finding the complete solution is $O(b^{n_1} + b^{n_2} + \cdots + b^{n_k})$ which is $O(b^{max(n_1,n_2,\ldots,n_k)})$.

In [4] we also report on a series of different experiments designed for evaluating PARIS. Table 1 summarizes some of the experimental results that were obtained by using PARIS to solve 100 different process planning problems. In this experiment, PARIS was trained with 5 different cases (10 cases in the second run) which are available for reuse during problem solving. The table summarizes the percentage of problems that could be solved and the average computation time required to solve each problem. The results are compared to those obtained in a separate experiment in which each problem is solved by pure search without reuse.

	Solved Problems	Solution Time
reusing 5 cases	83 %	59 sec.
reusing 10 cases	86 %	56 sec.
no reuse	29 %	157 sec.

Table 1. Percentage of solved problems and average solution time.

4 Related Work

4.1 RESYN/CBR

RESYN/CBR (this volume) is a case-based planning system which is specialized for the synthesis of plans in organic chemistry. The chemical starting material can be considered the initial planning state and the target molecule the goal state. Chemical transforms are the planning operators. Compared to PARIS, RESYN/CBR does not make use of abstraction. There are no abstract operator or abstract descriptions of states (molecules). However, it uses a subsumption hierarchy to organize the case-base which seems to be similar to the approach used in PARIS where an abstraction hierarchy is used.

4.2 Deja Vu

DEJA VU (this volume) is a hierarchical CBR system which solves tasks similar to planning. However, DEJA VU has no explicit representation of states and operators as used in PARIS. Consequently, it cannot solve planning problems from scratch without CBR and it cannot guarantee the correctness of produced solutions. Like PARIS, DEJA VU uses cases at different levels of abstraction. Unlike PARIS, DEJA VU allows to combine different cases - conrete and more abstract ones - to build a new solution. Since DEJA VU does not represent operators explicitly, it uses different kinds of adaptation transformations (i.e. called specialists, strategies). So, from a knowledge acquisition point of view, both systems are quite different. PARIS requires the acquisition of operators at different levels of abstraction, each of which must be described by their precondition and effects. DEJA VU requires the acquisition of general adaptation transformations. It seems that both systems have advantages in different domains.

4.3 EADOCS

EADOCS (this volume) is a case-based design system that uses an object-oriented representation of designs. Designs are represented as aggregations and specializations of objects. When reusing a design, only parts of this object structure may be useful. Reusing only the top-level objects of such a case may be comparable to reusing an abstract case as in PARIS.

4.4 Other approaches to case-based planning

Most similar to the PARIS approach are the case-based planning systems Prodigy/Analogy [10], PRIAR [6] and CAPLAN/CBC [9, 8]. However, these systems reuse planning cases directly and without doing any abstraction. Our approach is also related to the idea of skeletal plans [5]. In the skeletal plan approach no model of the operators (neither concrete nor abstract) is used to describe the preconditions and effects of operators as is done in PARIS. There is no explicit notion of states and abstraction or refinement of states. Instead, the plan refinement is achieved by stepping down a hierarchy of operators, guided by heuristic rules for operator selection.

5 Appendix

This section addresses particularly the questions to be discussed in the workshop.

5.1 Abstraction

In PARIS different levels of abstraction (concrete and abstract planning domain) are explicit, i.e., they are defined as part of the general knowledge used by the system. Although there are only two distinct levels, an arbitrary number of abstraction levels can be merged into the abstract planning domain (see also [4]). Different vocabularies are used for each level of abstraction. This is required to represent the "naturally" occuring abstraction levels of the domain. One of the main purposes for using abstraction is to increase the flexibility of reuse. If a case cannot be used at a concrete level, it may however be reused at some abstract level.

5.2 Adaptation Knowledge

PARIS requires a complete hierarchical model of the planning domain, i.e., set of operator descriptions (STRIPS) on several levels of abstraction. Additionally, it requires a generic abstraction theory that allows to abstract planning states. This knowledge is sufficient for from-scratch problem solving and for determining the correctness of a solution. It is used in the retrieval, reuse, and retain phase. The approach is limited to planning task but it is domain independent. The knowledge is represented by STRIPS operators and horn clauses.

5.3 Retrieval and Indexing

The abstraction approach used in PARIS strongly affects the similarity assessment and the memory organisation. The retrieval only selects cases that match the problem exactly at a certain level of abstraction. If several matching cases are contained in the case-base, the retrieval selects the case that is located at the lowest level of abstraction. Thereby, similarity is defined by the level of abstraction on which problems match exactly. This approach, however, does not guarantee that the selected cases can be adapted. In order to increase the *probability* that only adaptable cases are retrieved, statistic information about cases is collected during adaptation. This information influences the retrieval and allows to "forget" cases that are not adaptable for many situations. This statistic information also takes the retrieval time into acoount.

5.4 Adaptation

In planning, adaptation is always required because of the large (typically infinite) solution space. It is extremely unlikely that a plan can be reused for a new problem without modification. This holds particularly for the domain of mechanical engineering. So, the available cases are unsufficient. The adatation approach is described in more detail in sections 2 and 3. The cases do not encode domain knowledge; the encode control knowledge. Cases are not combined during adaptation. A generic problem solver (a state-space planner) is used for refining abstract cases. Case-based problem solving is used to increase the performance (reduction of the problem solving time) and the competence (increase the number of solved problems within a certain time limit). This has been validated in several experiments (see [4, 2]). The adaptation is not performed in co-operation with the user.

5.5 Case structure

Like other approaches in case-based planning, PARIS uses a case structure which clearly destinguishes between a problem and a solution part. The problem part is defined by the initial state and the final state, and the solution part consists of a totally ordered sequence of operators. This view is definitly influenced by the classical planning literature. All approaches that rely on this view can only be applied for action planning tasks. The advantage is that the kind of problems to be solved is clearly defined and criteria exist which define when a solution is correct. The disadvantages is the limited flexibility. Only such kinds of problems can be solved by the system that can be represented by combining the predefined goals. This distinction also influences the indexing in that only the problem part needs to be used for indexing.

5.6 Requirements and Limitations

The most important prerequisite of this method is the availability of the required background knowledge, namely the concrete world description, the abstract world description, and the generic abstraction theory. For the construction of a planning system, the concrete world description must be acquired anyway, since it specifies the language of the problem description (essential sentences) and the problem solution (operators). The abstract world and the generic abstraction theory must be acquired additionally. We feel that this is indeed the price we have to pay to make reuse more flexible and thereby planning more tractable in certain practical situations. Nevertheless, the formulation of an adequate abstract domain theory is crucial to the success of the approach. If those abstract operators are missing which are required to express a useful abstract plan, reuse cannot be achieved. What we need are mostly independently refinable abstract operators. If such operators exist, they can be simply represented in the abstract domain using the whole representational power. Additionally, the requirement that the abstract domain is given by the user has also the advantage that the learned abstract cases are expressed in terms the user is familiar with. Thereby, the user can understand an abstract case very easily. The main goal of adaptation in case-based planning is to reduce the problem solving time. So, the goal of the retrieval is to find a case that can be adapted with minimal effort. This is reflected in the retrieval procedure of the PARIS system through the statistic information that is collected during case adaptation.

REFERENCES

[1] A. Aamodt and E. Plaza, 'Case-based reasoning: Foundational issues, methodological variations, and system approaches', *AI Communications*, **7**(1), 39–59, (1994).

[2] R. Bergmann, *Effizientes Problemlösen durch flexible Wiederverwendung von Fällen auf verschiedenen Abstraktionsebenen*, Ph.D. dissertation, University of Kaiserslautern, 1996.

[3] R. Bergmann, G. Pews, and W. Wilke, 'Explanation-based similarity: A unifying approach for integrating domain knowledge into case-based reasoning', in *Topics in Case-Based Reasoning*, eds., S. Wess, K.-D. Althoff, and M.M. Richter, volume 837 of *Lecture Notes on Artificial Intelligence*, 182–196, Springer, (1994).

[4] R. Bergmann and W. Wilke, 'Building and refining abstract planning cases by change of representation language', *Journal of Artificial Intelligence Research*, **3**, 53–118, (1995).

[5] P. E. Friedland and Y. Iwasaki, 'The concept and implementation of skeletal plans', *Journal of Automated Reasoning*, **1**(2), 161–208, (1985).

[6] S. Kambhampati and J. Hendler, 'A validation-structure-based theory of plan modifications', *Artificial Intelligence*, (1992).

[7] R. E. Korf, 'Planning as search: A quantitative approach', *Artifical Intelligence*, **33**, 65–88, (1987).

[8] H. Muñoz, J. Paulokat, and S. Wess, 'Controlling a nonlinear hierachical planner using case-based reasoning', in *Proceedings 2nd EWCBR*, (1994).

[9] J. Paulokat and S. Wess, 'Planning for machining workpieces with a partial-order, nonlinear planner', in *AAAI-Fall Symposium on Planning and Learning: On to Real Applications*, (1994).

[10] M. M. Veloso and J. G. Carbonell, 'Towards scaling up machine learning: A case study with derivational analogy in PRODIGY', in *Machine Learning Methods for Planning*, ed., Steven Minton, chapter 8, 233–272, Morgan Kaufmann, (1993).