Learning and Clustering Plan Abstractions to Improve Hierarchical Planning

Ralph Bergmann
German Research Center for Artificial Intelligence
Erwin-Schrödinger-Str. Bau 57
D-6750 Kaiserslautern, Germany
E-Mail: bergmann@informatik.uni-kl.de

Abstract

Hierachical planning can be improved by explanation-based learning (EBL) of abstract plans from detailed, successfully solved planning problems. Abstract plans, expressed in well-established terms of the domain, serve as useful problem decompositions which can drastically reduce the planning complexity. The learned plan abstraction must be valid for a class of planning cases rather than for a single case, to ensure their successful application in a larger spectrum of new situations. A hierarchical organization of the newly learned knowledge must be archieved to overcome the utility problem in EBL. This paper presents a new formal model of shared plan abstraction and the closely related explanation-based procedure S-PABS. Unlike other apporaches to plan abstraction, our model allows a total different terminology to be introduced at the abstract level. Finally, an unsupervised incremental procedure for constructing a hierachy of shared abstract plans is proposed, as a kind of concept formation over explanations.

1 INTRODUCTION

Recently, a lot of approaches have been proposed for improving planning by the use of machine learning techniques. Many approaches favor the analysis of success cases of problem solutions to create new amounts of knowledge which can be additionally used by a planning system. Planning improvement is usually achieved through the shortening of the problem solving process by the reuse of earlier problem solving experience from an expert or the planning system itself. While in case-based reasoning [Kolodner, 1987] a large collection of very detailed cases is organized for efficient retrieval and modification, traditional learning approaches prefer the extraction of more general knowledge from problem solutions. With explanation-based learning (EBL) [Mitchell *et al.*, 1986] knowledge about the problem domain can be incorporated to extract control rules [Minton *et al.*, 1989] macro-operators [Fikes *et al.*, 1972; Korf, 1985; Tadepalli, 1991] or skeletal plans [Schmalhofer *et al.*, 1991a; Bergmann, 1992b] as *generalizations* of successful (or failed) planning cases. Unfortunately, it shows that simply storing generalizations does not guarantee a speed up effect in many situations. This is because the search for applicable short-

cut rules and their matching can sometimes exceed the savings which their utilization will cause [Minton, 1990]. To overcome this so-called *utility problem*, Yoo and Fisher [Yoo and Fisher, 1991] have recently proposed the construction of a hierarchical classification tree to store generalizations from multiple examples and to allow efficient retrieval of generalizations.

In an other line of research on planning, *abstraction* has been identified as a powerful method to make planning more tractable [Sacerdoti, 1974; Friedland and Iwasaki, 1985]. Korf [Korf, 1988] has shown that, under certain conditions [Knoblock, 1989], abstraction can reduce a planning problem which requires a search in an exponential space to a linear problem. Although Korf's requirements are very difficult to fulfill, especially for real world applications, the introduction of abstract planning spaces is generally assumed to be very useful for improving planning performance. The main computational advantage of having a good abstract solution to a planning problem is, that one large search space of the complexity b^n (b is the branching factor and n the length of a solution plan) can be decomposed into k smaller search spaces in which the problem solution requires a reduced complexity of $b^{n_1}+b^{n_2}+\ldots+b^{n_k}$ $(n_1+n_2+\ldots+n_k = n)$. As also identified by [Knoblock, 1989] it is important to avoid backtracking across abstraction levels as well as backtracking across subproblems within an abstraction level. Fairly independently solvable subproblems should be learned by an abstraction process. But this seems difficult to guarantee for real word domains by employing general week methods [Unruh and Rosenbloom, 1989] without domain specific abstraction knowledge.

From the results concerning the utility problem in EBL and from the promissing complexity reductions that can be theoretically achieved by abstraction the following requirements on a learning procedure to reduce planning complexity are derived:

1. A learning procedure should learn from success cases and schould construct abstract plans which serves as useful decompositions of a planning problem into several smaller problems.

2. The created abstractions should be tailored to the application domain and should use an established terminology which has shown to lead to independently solvable subproblems in human problem solving.

3. Problem decompositions should be learned that are shared by larger class of planning cases rather than only by a single case. Thereby, abstractions can be selected which can promise to be successfully applicable in a larger spectrum of new situations.

4. The body of the learned knowledge must be arranged in a memory organization structure which allows an efficient retrieval of abstract plans.

This paper presents a new formal model of plan abstraction and demonstrates how explanation-based learning can be applied with respect to the above mentioned requirements. In the next section, a model of shared plan abstractions from mutiple plans is introduced. In section three, the five phase S-PABS (Shared Plan Abstraction) procedure is presented as a method which learns form a set of plans and comes out with a shared abstract plan. Section four shows how methods of incremental concept formation can be utilized to construct a classification tree of the learned abstract plans.

The described approach is demonstrated for the familiar 'Towers of Hanoi' domain. Finally, section five discusses perspectives and related work in connection with the S-PABS approach.

2 FORMAL MODEL OF PLAN ABSTRACTION

Michalski and Kodratoff [Michalski and Y.Kodratoff, 1990] have recently pointed out that abstraction has to be distinguished from generalization. While generalization transforms a description along a set-superset dimension, abstraction transforms a description along a level-of-detail dimension which usually involves a change in the representation space from the original into a simpler language [Plaisted, 1981; Tenenberg, 1987; Giordana *et al.*, 1991; Mozetic and Holzbaur, 1991]. A plan is composed of operations which are executed in a specific order and thereby successively change the state of a world. Within this planning model, abstraction has two independent dimensions: On the first dimension a change in the level of detail for the representation of single states is described. On the second dimension a change in the level of detail is declared by reducing the number of states contained in a plan. As a consequence, a change of the representation of the state description and a change of the operations which describe the state transitions is required. Both dimensions of abstraction are essential to achieve a reduction of the complexity for planning.

2.1 ABSTRACTION OF A SINGLE PLAN

In the following, the formal model of plan abstarction from a single plan [Bergmann, 1992c] is introduced first. The commonly used notation for the description of worlds, states and plans (e.g. [Fikes *et al.*, 1972; Lifschitz, 1987; Knoblock, 1989]) is further assumed:

Definition 1: A *STRIPS world W* is a triple (R,T,Op) over a first-order language L, where R is a set of essential sentences [Lifschitz, 1987] which describe the dynamic aspects of a state of the world. T is a static theory which allows to deduce additional properties of a state in the world. Op is a set of operators represented by descriptions $\langle P\alpha, D\alpha, A\alpha\rangle_{\alpha\in Op}$, where $P\alpha$ is the precondition formula, $D\alpha$ is the delete list and $A\alpha$ is the add list [Fikes *et al.*, 1972]. As usual, a *state* s of a world W is described by a subset of the essential sentences from R. The theory T is implicitly assumed to be valid in all states of the world. Let $\mathcal{S} = 2^R$ be the set of all states of the world.

A *plan* p in a world W is a sequence $(o_1, \ldots, o_n)$ of operators from Op. In a world W an initial state $s_0 \in \mathcal{S}$ and a plan $p = (o_1, \ldots, o_n)$ *induce* a sequence of states $s_1 \in \mathcal{S}$,..., $s_n \in \mathcal{S}$ where $s_{i-1} \cup T \vdash P_{o_i}$ and $s_i = (s_{i-1} \backslash D_{o_i}) \cup A_{o_i}$. Two plans $p = (o_1, \ldots, o_n)$ and $p' = (o'_1, \ldots, o'_n)$ are called *equivalent* iff in every state $s_0 = s'_0 \in \mathcal{S}$ follows that $s_i = s'_i, i \in \{1, \ldots, n\}$ for the states induced by the plans.

In the following, we assume that the two world descriptions $W_c = (R_c, T_c, Op_c)$ (the concrete world) and $W_a = (R_a, T_a, Op_a)$ (the abstract world) are given, as appropirate descriptions of the planning domain (see requirement 2). The problem of plan abstraction can now be described as transforming a plan pc from the concrete world W_c into a plan pa in the abstract world W_a, with several conditions being satisfied. In the presented model, this transformation is formally decomposed into two mappings, a *state abstraction mapping a*, and *a se-*

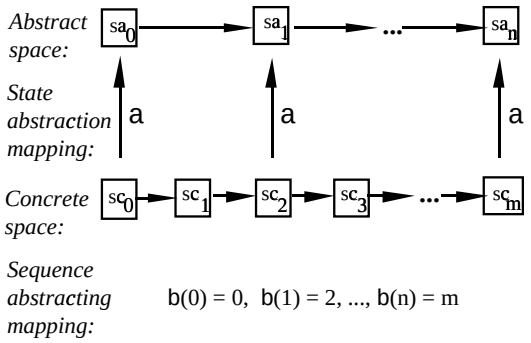

Figure 1: Demonstration of the Plan Abstraction Methodology

quence abstraction mapping b as follows:
Definition 2: A *state abstraction mapping a*: $S_c \rightarrow S_a$ is a mapping from S_c, the set of all states in the concrete world, to S_a, the set of all states in the abstract world, that satisfies the following conditions:

a) If $sc \cup T_c$ is consistent then $a(\mathrm{sc}) \cup T_a$ is consistent for all sc $\in \mathcal{S}_c$ (maintain consistency).

b) If sc $\cup sc' \cup$ Tc is consistent then $a(sc \cup sc') \subseteq a(sc) \cup a(sc')$ for all sc, $sc' \in S_c$ (monotony property).

The state abstraction mapping transforms a concrete state description into an abstract state description and thereby changes the representation of a state from concrete to abstract.
Definition 3: A *sequence abstraction mapping b*: $N \rightarrow N$ relates an abstract state sequence $(sa_0, \ldots, sa_n)$ to a concrete state sequence $(sc_0, \ldots, sc_m)$ by mapping the indices $i \in \{1, \ldots, n\}$ of the abstract states sa_i into the indices $j \in \{1, \ldots, m\}$ of the concrete states sc_j, so that the following properties hold:

a) $b(0) = 0$ and $b(\mathrm{n}) = \mathrm{m}$: The initial state and the goal state of the abstract sequence must correspond to the initial and goal state of the respective concrete state sequence.

b) $b(\mathrm{u}) < b(\mathrm{v})$ iff u<v: The order of the states defined through the concrete state sequence must be maintained for the abstract state sequence.

Definition 4: A plan pa is an *abstraction* of a plan pc if there exists a state abstraction mapping a: $\mathcal{S}$c $\rightarrow \mathcal{S}$a and a sequence abstraction mapping b: $N \rightarrow N$, so that: If pc and an initial state sc_0 induce the state sequence $(sc_0, \ldots, sc_m)$ and $sa_0 = a(sc_0)$ and $(sa_0, \ldots, sa_n)$ is the state sequence which is induced by sa_0 and the abstract plan pa, then $a(sc_{b(i)}) = sa_i$ holds for all $i \in \{1, \ldots, n\}$.
This definition of abstraction is demonstrated by Figure 1. The concrete space shows the sequence of m operations together with the induced state sequence. Selected states induced by the concrete plan (i.e. sc_0, sc_2, and sc_m) are mapped by the state abstraction mapping a into states of the abstract space. The sequence abstraction mapping b maps the indices of the abstract states to the corresponding states in the concrete world. It becomes clear, that plan abstraction is defined by a pair of abstraction mappings (a,b). This is because these mappings uniquely define an equivalence class of abstract plans according to the defined plan equivalence (if their exists an abstract plan at all).

2.2 DOMAIN JUSTIFICATION FOR STATE ABSTRACTION MAPPINGS

For the construction of domain tailored abstraction mappings (see requirement 2), a justification of the state abstraction mappings by knowledge of the domain is necessary. Otherwise, arbitrary mappings of concrete objects to abstract objects in the description of the states can-

not be avoided. Therefore, additional domain knowledge should state which kinds of useful abstractions usually occur in a domain, and how those abstractions can be defined in concrete terms. So, the construction of abstraction mappings is a priori restricted to possibly useful ones.
Generic abstraction theories for semantic abstraction, as introduced by Giordana, Roverso and Saitta [Giordana *et al.*, 1991] relate atomic formulae of an abstract language to terms of a corresponding concrete language. In an adaption of this idea, a generic state abstraction theory T_g in our model is defined as a set of axioms of the form $\Psi \leftrightarrow D_1 \vee D_2 \vee \ldots \vee D_n$, where Ψ is an essential sentence of the abstract world and $D_1, \ldots, D_n$ are conjunctions of sentences of the concrete world. For a justified state abstraction mapping a we can now require that: if $\Psi \in a(\mathrm{sc})$ then $sc \cup Tc \cup Tg \vdash \Psi$. Since a minimal consistent state abstraction mapping (according to the $\supset$ - relation) should be reached, the reverse implication, namely that every essential sentences Ψ for which $sc \cup Tc \cup Tg \vdash \Psi$ holds, is an element of $a(\mathrm{sc})$, is not demanded.

2.3 SHARED ABSTRACTIONS FROM SEVERAL PLANS

Since the definition of plan abstraction according to definition 4 only describes the abstraction of a single plan, the presented model must be extended to deal with he formulation of one shared abstraction form a set of plans (see requirement 3). Intuitively, an abstract plan which holds for a set of plans must be an abstraction for each of the plans. So, for each of the plans to be abstracted, a state abstraction mapping and a sequence abstraction mapping must be found, so that the same sequences of abstracted state descriptions are created. This idea is captured in the next definition

Definition 5: Let $P = \{< pc_1, sc_{0,1} > , \ldots, < pc_k, sc_{0,k} >\}$ be a set of k plans pc_ν and belonging initial states [1] $sc_{0,\nu}$ and let $(sc_{0,\nu}, \ldots, sc_{m_\nu,\nu})_{\nu \in \{1,\ldots,k\}}$ denote the sequences of states which are induced by P. A plan pa is a *shared abstraction of a set of plans* P if there exist k state abstraction mappings $a^{(1)}, \ldots, a^{(k)}$ and k sequence abstraction mappings $b^{(1)}, \ldots, b^{(k)}$, so that: If $sa_0 := a^{(1)}(sc_{0,1})$ and $(sa_0, \ldots, sa_n)$ is the state sequence which is induced by sa_0 and the abstract plan pa, then $a^{(\nu)}(sc_{[b^{(\nu)}(i)],\nu}) = sa_i$ holds for all $i \in \{0, \ldots, n\}$ and for all $\nu \in \{1, \ldots, k\}$.

2.4 AN EXAMPLE

To illustrate the formal model of shared plan abstraction a small example is introduced now. As a planning domain, the familiar Tower-of-Hanoi (ToH) problem - a problem analyzed by several researchers before (e.g. [Korf, 1985]) - is used. The ToH-problem involves three vertical pegs and a number of shaped disks, all of different sizes. In the initial state, all the disks are stacked on one peg in decreasing order of size. The goal is to stack the pegs in the same order on one of the other pegs. The only legal action is to move a single top disk on a peg to another peg subject to the constraint, that a larger disk may never be placed on a smaller disk.
Figure 2 shows two example problem solutions to the 2-disk and the 3-disk ToH problem. In the top of this figure, the plan pc_1 for the 2-disk ToH-problem is shown. In the initial state $sc_{0,1}$ the two disks are stacked on the left most peg

[1] Note, that we use only one index to refer to different states of a plan when only one plan is involved. A second index (e. g. ν) is added, if it is necessary to differentiate between the state sequences of different plans.

a. The other pegs b and c are empty. The stacked disks of the three pegs are represented as three columns of natural numbers, where the value of a number reflects the sizes of the respective disks (small numbers stand for small disks). An empty peg is represented by the underscore (_) symbol. The first operation of the plan pc_1 is the move of the top disk (1) on peg a to peg b. This operation is represented by the term *m(a,b)* which denotes the application of the operator *move* (abrivated by m) with the two parameters source peg a and destination peg b. The second operation moves the top disk from peg a to peg c and the third operation achieves the goal state in which all pegs are correctly store on the right most peg c. The bottom of Figure 2 shows the plan pc_2 as a solution to the 3-disk ToH-problem. The three disks (numbered *1,2,3*) are moved with 7 legal moves from peg a to peg c. The shared abstraction which is derived according to the model of plan abstraction is indicated between the two concrete plans.

Here, the abstract world consists of a different terminology for the descriptions of the states and operations than the concrete world. In the state descriptions, the symbol t is intoduced as an abstraction of a complete tower of all disks contained in a problem. The symbol l stands for the largest disk and s abbreviates a small tower of disks, which is a tower that does not contain the largest disk. The generic abstraction theory which is required to allow justified abstraction mappings exactly defines these new abstract objects *(t, l, s)* in terms of the concrete disks. Additionally, new abstract operations which act on the changed state representation are introduced. The operation *split* splits a complete tower t into the largest disk and the small remaining tower s. The operation m' moves a single disk or any tower to a new location. The *join* operation stacks a small tower on top of a large disk. Note that all these abstract operations are not legal elementary operations of the concrete ToH-domain. The two state abstraction mappings and the two sequence abstraction mappings which define the shared abstraction of the 2-disk and the 3-disk problem are listed in Table 1[2]. $a^{(1)}$ maps

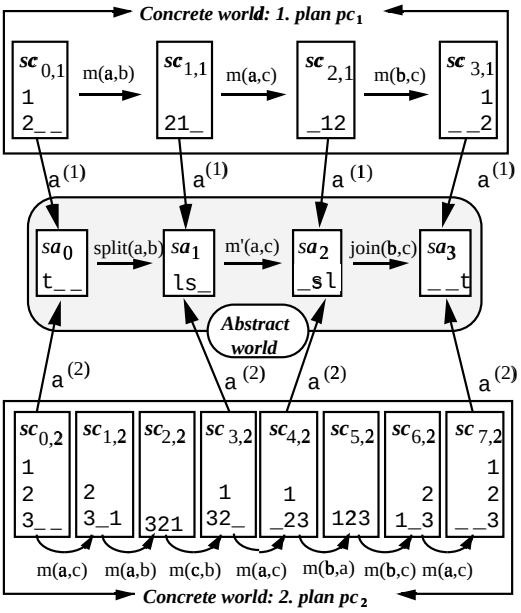

Figure 2: An Example for a Shared Abstraction for two plans

Table 1: Example for Abstraction Mappings

	Plan pc_1 (2 disks)	Plan pc_2 (3 disks)
State abstraction mappings:	$a^{(1)}([1]) = s$ $a^{(1)}([2]) = l$ $a^{(1)}([\frac{1}{2}]) = t$	$a^{(2)}([\frac{1}{2}]) = s$ $a^{(2)}([3]) = l$ $a^{(2)}([\frac{1}{\frac{2}{3}}]) = t$
Sequence abstraction mappings:	$b^{(1)}(0) = 0$ $b^{(1)}(1) = 1$ $b^{(1)}(2) = 2$ $b^{(1)}(3) = 3$	$b^{(2)}(0) = 0$ $b^{(2)}(1) = 3$ $b^{(2)}(2) = 4$ $b^{(2)}(3) = 7$

[2] To simplify the presentation, the shown state abstraction mappings only note the mapping of disk configurations, although they formally map complete states, which are sets of essential

the concrete from the plan pc_1 onto the abstract states and $a^{(2)}$ maps the concrete states which result from the plan pc_2 onto the same sequence of abstract states. The sequence abstraction mapping $b^{(1)}$ mirrors the fact, that each state in the solution of the 2-disk problem is mapped onto a respective abstract state, while for the 3-disk problem only 4 of the 8 states have an abstract image according to $b^{(2)}$.

3 EXPLANATION-BASED LEARNING OF ABSTRACTIONS

As described in section 2, the task of constructing a shared plan abstraction can be seen as the problem of finding deductively justified state abstraction mappings and sequence abstraction mappings so that a related abstract plan - composed of the operators of the abstract world - exists. This section introduces S-PABS [3] (shared plan abstraction), an EBL-procedure which consists of five distinct phases.

3.1 EXPLANATION STRUCTURES

Traditonal EBL has the property, that the knowledge which is derived from an example as operational generalization is already a consequence of the incorporated domain theory. No knowledge can be acquired which is more general than the theory used for explanation, or more precisely, no rules can be derived which are more general than the rules which are used to *explain* the examples. To allow EBL to come out with described type of plan abstractions, the explanations which are constructed must explain the example plans in terms of the abstract world description. Therefore our domain theory must not only contain the concrete world description, but also the abstract world description and the generic abstraction theory. Such additional knowledge is not necessary for other approaches for learning generalizations of plans [Minton *et al.*, 1989; Bergmann, 1992b].
Moreover, the construction of shared abstractions on possibly different levels of abstraction necessitates the derivation of several alternative explanations (in the abstract world), for each plan involved. The most specifc explanation in the abstract world which is valid for all plans may then be selected as the common explanation. This explanation is complete in the sense, that all operations of the plan are explained, but in a less detailed way. This is in constrast to the kind of shared explanations which are constructed by *Induction over Explanations* IOE [Flann and Dietterich, 1989] or the EXOR system [Yoo and Fisher, 1991]. In these two approaches, the common subexplanation is not anymore complete for all examples.

3.2 THE S-PABS PROCEDURE

The S-PABS learning procedure is decomposed into five distinct phases in which the different types of knowledge is used to infer an appropriate explanation structure, from which an abstract plan can be derived. The first three phases are executed separately for each of plans $\{pc_1, \ldots, pc_k\}$. Phase-IV superimposes the candidate abstract explanations yielding a set of shared plans, which are further variabilized in phase-V. In the following the five phases are explained in

sentences.

[3] S-PABS is an extension of the PABS-procedure, proposed in [Bergmann, 1992c].

detail.

3.2.1 Phase-I: Plan Simulation (Application of Concrete World Knowledge)

By simulating the execution of the concrete plan pc, the state sequence $(sc_1, \ldots, sc_m)$ which is induced by the plan pc and a given initial state sc_0 is computed (see Figure 1). During this simulation, the definition of the operators Op_c and the static theory T_c are applied to derive all those essential sentences which holds in the respective states. The proofs that exist for the applicability of each operator can now be seen as a concrete-level explanation for the effects caused by the operations. Such a kind of explanation is also constructed in EBL-procedures for plan generalisations [Fikes *et al.*, 1972; Chien, 1989; Schmalhofer *et al.*, 1991a; Bergmann, 1992b] and for constructing programming schemes from verified programs [Bergmann, 1992a]. For the 3-disk ToH-problem the computed state sequence $sc_{0,2}, \ldots, sc_{7,2}$ is shown in the lower section of Figure 2.

3.2.2 Phase-II: Constructing State Abstractions (Application of the Generic Abstraction Theory)

The second phase performs a prerequisite for the composition of the deductively justified state abstraction mapping. With the generic state abstraction theory T_g, an abstract

state description sa'_i is derived for each state sc_i which was computed in the first step. Essential sentences $\Psi \in R_a$ of the abstract world description W_a are checked, whether they can be inferred from $sc_i \cup T_c \cup T_g$. If $sc_i \cup T_c \cup T_g \vdash \Psi$ holds, then Ψ is included into the state abstraction sa'_i. Although, each of the concrete states is transformed with the guidance of the generic abstraction theory into abstract descriptions, not every state has a meaningful abstract interpretation. For which states the abstraction turns out to be useful, can ultimately be answered in phase-IV. For the 3-disk ToH-problem, some of the abstract essential sentences which are derived from the states $sc_{0,2}, sc_{3,2}, sc_{4,2}, sc_{7,2}$ are shown in the centre of Figure 2 and also in Figure 3.

3.2.3 Phase III: Constructing Abstract State Transitions (Application of Abstract World Knowledge)

The goal of the third phase is to identify candidate abstract operations for an abstract plan. For each pair of abstracted states (sa'_u, sa'_v) with u < v, it is checked, if there exists an abstract operation $O_\alpha \in Op_a$ described by $< P_\alpha, D_\alpha, A_\alpha >$ which is applicable in sa'_u and which transforms sa'_u into sa'_v. If $sa'_u \cup T_a \vdash P_a$ and if every sentence of A_α is contained in sa'_v and none of the sentences of D_α is contained in sa'_v then the operation O_α is noted to be a candidate for the abstract plan. A directed graph is constructed, where the nodes of the graph are built by the abstract states sa'_i and where links between the states are introduced for those operations that are candidates for achieving the respective state transitions. The proofs that exist for the validation of P_α in sa'_u are stored together with the corresponding operation. A fraction of the complete graph of the candidate abstract operations is shown Figure 3. For some of the abstract states, the description derived in phase II is shown. In addition to the disk abstraction symbols *t,l,s* which have already been introduced in section 2.4, the symbol n represents the second

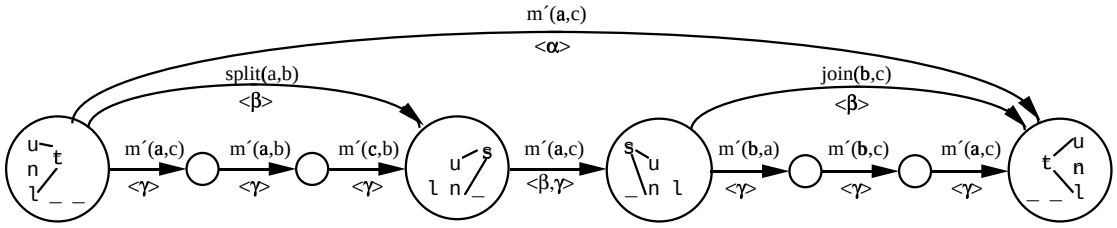

Figure 3: Graph of Candidate Abstract Operations from 3 and 4-disk ToH-problem and Indicated Consistent Paths <α,β,γ>

largest disk and the symbol u represents any tower (complete or partial) which does not contain the largest or the second largest disk. The presented graph can be derived from the 3-disk ToH-problem as well as for then 4-disk ToH-problem.

3.2.4 Phase IV: Establishing Shared Consistent Paths

From the constructed graph, complete and consistent paths from the initial abstract state sa'_0 to the final state sa'_m are searched, where each path determines a complete abstract explanation. The consistency requirement for such a path pa $= (o_1, \ldots, o_n)(o_i \in Op_a)$ expresses, that every essential sentence which guarantees the applicability of the operator $o_{i+1}(sa'_i \cup T_a \vdash P_\alpha)$ is created by a preceding operation (through the add-list) and is protected until o_{i+1} is applied, or the essential sentence is already true in the initial state and is protected until o_{i+1} is applied. This condition assures that the plan represented by the path pa is indeed applicable, which means that the preconditions for all operations are satisfied in the states in which they are executed. This consistency requirement can be verified by analyzing the dependencies of the involved operations. The graph shown in Figure 3 consists of five paths from the initial to the final abstract state, but only three paths, marked $< \alpha >, < \beta >$, and $< \gamma >$ fulfill the introduced consistency requirement. Although two states (and one operation) are shared by the paths $< \beta >$ and $< \gamma >$, the crossing of them does not represent a consistent path. This is because the operations in path $< \beta >$ work on the disk abstractions noted by the symbols *t,l,s*, whereas the operations in path $< \gamma >$ rely on a more detailed view on the disk-configuration represented by the symbols *l,n,u*.

The modules of the S-PABS procedure reported so far, does not take the construction of shared explanations into account. So they are be executed separately for each of the available plans. The determined set of consistent paths can then easily be superimposed to select only those paths which are shared by all of the example plans. For judging if some paths are shared, it is important to take the intermediate states induced by the plans into consideration too (refer to definition 5). In the example graph from Figure 3, we can identify that the path $< \alpha >$ is shared by the plans for the 1,2,3 and 4-disk ToH-problem. Path $< \beta >$ represents an explanation for the 2,3 and 4-disk problem, while path $< \gamma >$ only holds for the 3 and the 4-disk problem. From this example we can also see, that possibly more then one path can survive the process of intersection. So all three paths $< \alpha >, < \beta >$, and $< \gamma >$ are shared consistent paths for the 3 and the 4-disk ToH-problem. In this case S-

PABS will come out with several plan abstractions from which some may be selected for further usage. Although this may seem to be a problem, it can also be used as a guidance for a clustering process, as it will be pointed out in section 4.

3.2.5 PhaseV: Constructing the Final Abstract Plan Representation

From a shared abstract path and the dependency network which justifies its consistency, a variabilization of the abstracted plan can be established. With the dependency network, which functions as an explanation structure, explanation-based generalization can be applied to compute the least subexplanation which justifies all operations of the abstract path. The proofs that correspond to the justification of the abstract states by the generic abstraction theory are pruned Thereby, the boundary of operationality [Braverman and Russel, 1988] for the learned concept is determined by the essential sentences and the operators available in the abstract world and by the set of examples from which the shared abstractions are generated. Within this subexplanation, the remaining derivations are generalized by standard goal regression as used by Mitchell, Keller and Kedar-Cabelli [Mitchell *et al.*, 1986]. Thereby, constants are turned into variables. The final generalized explanation thus only contains relations which describe the generalized operations together with a generalized specification of the application conditions for the operator sequence. As an example, the variabilized abstract plan which results from path $< \beta >$ is shown in Figure 4. Note that the capital letters X, Z, Y now indicate variables which stand for the pegs of ToH.

3.3 DETERMIMATION OF STATE AND SEQUENCE ABSTRACTION MAPPINGS

As demonstrated, the described five-phase-procedure computes shared plan abstractions. Now we want to briefly elaborate the relation between the S-PABS procedure and the formal model described in section 2.
As pointed out in definition 5, a shared abstract plan which is vaild for a set of k concrete plans is described by k state abstraction mappings and k sequence abstraction mappings. These mappings are implicitly defined during the first four phases of the procedure. For each source plan pc_ν the sequence abstraction mappings $b^{(\nu)}$ is defined through the selection of a consistent path:

Application Condition:	
- initial state:	on_peg(X,t), on_peg(Y,_), on_peg(Z,_)
- goal state:	on_peg(X,_), on_peg(Y,t), on_peg(Z,_)
- constraints:	X≠Y, X ≠Z, Y≠Z
Operator Sequence:	split(X,Z), m'(X,Y), join(Z,Y)

Figure 4: Variabilized Abstract Plan as Result of path <β>

$b^{(\nu)}(i) := j$ *iff* the i-th operation of the shared consistent path pa connects state $sa'_{1,\nu}$ with $sa'_{j,\nu}$, where these states are derived from the concrete states $sc'_{1,\nu}$ and $sc'_{j,\nu}$ induced by the plan pc_ν.
Additionally, the shared path defines a corresponding sequence of minimal abstract state descriptions $sa_0, \ldots, sa_n$ with $sa'_{[b^{(\nu)}(i)],\nu} \supseteq sa_i$. These minimal abstract states contain exactly those sentences which are necessary to guarantee the applicability of the operations in the path. For each source plan pc_ν a state abstraction mapping $a^{(\nu)}$ can now be defined through:
$a^{(\nu)}(sc) := \{\Psi | sc \cup T_c \cup T_g \vdash \Psi$ and $\exists i \in \{1, \ldots, n\}$ such that $\Psi \in sa_i$ and

$sc \subseteq sc_{[b^{(\nu)}(i)],\nu}\}$. So S-PABS produces indeed shared plan abstractions according to the proposed model. For the abstract plan from path $< \beta >$ (Figure 4) the abstraction mappings have already been presented in Table 1.

4 CLASSIFICATION OF PLAN ABSTRACTIONS

Since now, this paper has presented an approach for speeding up planning by learning shared abstractions from successful planning cases. Thereby, the introduced speedup requirements 1,2 and 3 are satisified. This section deals with approaches to an efficient organization and utilization of the learned abstract plans to achieve the fourth requirement.
As already proposed by Yoo and Fisher [Yoo and Fisher, 1991], concept formation over explanations is a method that combines the explanation-based paradigm with the paradigm of concept formation [Gennari *et al.*, 1989] to result in a method which can automatically create a hierarchical classification tree of shared explanations.

4.1 FUNDAMENTALS FOR THE HIERARCHY CONSTRUCTION

The basic idea is to construct a classification hierarchy, in which each node in the hierarchy reflects the shared abstraction of a set of planning cases. The abstract plan which can be stored at a node, is then valid for all of the node's descendents. To construct such a classification hierarchy, one important property of S-PABS can be utilized.

If we look at S-PABS as a learning function $\mathcal{L}$ which maps a *set of concrete plans* onto a *set of abstract plans*, we

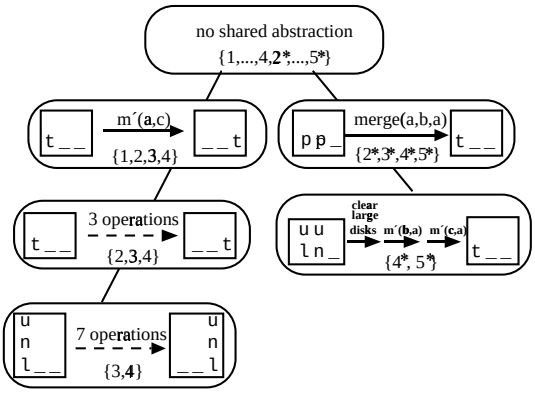

Figure 5: Classification Hierachy of Plan Abstractions

can easiliy see that the following inclusion holds:

$$\mathcal{L}(P1) \subseteq \mathcal{L}(P2) \text{ if } P2 \subseteq P1$$

This statement expresses the obvious property, that abstractions learned from a set of concrete plans are also valid abstractions for any subset. On the other hand, it is clear that extending the set of plans from which abstractions are to be constructed may reduce the set of resulting shared abstractions. A classification of abstract plans can be build on the basis of a classification of the concrete example plans. If C_i is a node in the classification hierarchy, then let EC_i denote the set of the concrete example plans which belong to that class. If C_i is a subclass of C_j in the hierarchy, then $EC_i \subseteq EC_j$ holds and therefore $\mathcal{L}(EC_j) \subseteq \mathcal{L}(EC_i)$ is also true. An abstract plan pa_{C_i} can be associated with each class C_i, where $pa_{C_i} \in \mathcal{L}(EC_i)$.
A typical representative abstract plan for a class C_i should be chosen in a way that it differs from the abstract plans which have been selected for the superclasses of C_i. To achieve this condition, pa_{C_i} can be designated as follows: $pa_{C_i} \in (\mathcal{L}(EC_i) \setminus \mathcal{L}(EC_j))$ if C_j is superclass of C_i. The application domain as well as the classification hierarchy has

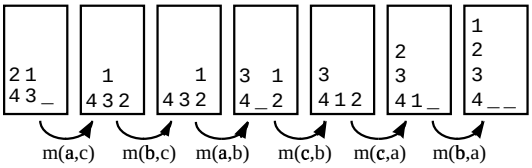

Figure 6: The 4-Disk Merge Problem 4*

an important impact on the size of the space of candidates ($\mathcal{L}(EC_i) \setminus \mathcal{L}(EC_j)$) from which to select an abstract plan for a class. A perfect hierarchy should be constructed in a way, that for each class the space of candidates contains only one item. In this case the set of abstract plans which is constructed for a set of example plans EC_i is distributed over all the nodes along the path from the root node of the hierarchy to the class C_i.

An example of such a classification hierarchy for the ToH domain is demonstrated in Figure 5, where 8 different problems are classified. As the first four problems (noted as 1,2,3,4) the traditional 1,2,3 and 4-disk ToH-problem (see section 2.4) are concerned. The other four problems $2^*, 3^*, 4^*, 5^*$ represent a variation of the typical ToH-problems. These problems deal with the merging of two fragmented towers on the pegs *a* and *b* into one complete tower located on peg *a*. The 4-disk merge problem (4*) and its concrete solution is briefely illustrated in Figure 6. In the hierarchy shown in Figure 5, the abstract plans for the different classes and the set of problems from which they are derived by S-PABS are shown. This hierarchy is constructed so, that each class can be characterized by a sole plan since $|\mathcal{L}(EC_i) \setminus \mathcal{L}(EC_j)| = 1$. Only the root node of the hierarchy does not contain an abstract plan because no shared abstractions between the traditional ToH-problems and the merge problems can be found.

4.2 INCREMENTAL CONSTRUCTION OF THE HIERARCHY

For an incremental construction of such a classification hierarchy, a newly observed solution to a planning problem E must be incorporated into an existing hierarchy. For that purpose, S-PABS computes $\mathcal{L}(EC_i), \mathcal{L}(\{E\})$ and $\mathcal{L}(EC_i \cup \{E\})$ for a class C_i of the hierarchy (initially the root). When the three sets of resulting abstractions are compared, three different situations are distinguished:

a) If $\mathcal{L}(EC_i \cup \{E\}) = \mathcal{L}(EC_i) = \mathcal{L}(\{E\})$ then the new example is simply discarded and not incorporated, because all of the examples abstractions are already contained in a class of the hierarchy.

b) If it shows that $\mathcal{L}(EC_i \cup \{E\}) = \mathcal{L}(EC_i)$ but $\mathcal{L}(EC_i) \neq \mathcal{L}(\{E\})$ then E becomes a member of the class C_i and the classification proceeds to the descendants of C_i. One subclass C_j is chosen in which E fits best. This selection is guided by a criterion that can be rated by $|\mathcal{L}(EC_j)| - |\mathcal{L}(EC_j \cup \{E\})|$, the number of abstractions of the class C_j which are not shared by E. (If no subclass can be chosen according to this criterion, then the subclass which contains the largest number of abstract plans is selected). If C_i is a leaf node of the hierarchy, then a new subclass C_j is created, which exactly contains the example plan E.

c) If $\mathcal{L}(EC_i \cup \{E\}) \neq \mathcal{L}(EC_i)$ then the new example does not completely fall in the scope class C_i (which is the best selection as guaranteed in case b). Therefore, a new class node called C_k is created and inserted between C_i and its father. This new

class is initialized with the example plans of C_i supplemented by the example E ($EC_k := EC_i \cup \{E\}$). Additionally, if $\mathcal{L}(EC_k) \neq \mathcal{L}(\{E\})$ a new child of C_k is created, which contains only the example plan E.

With this procedure the classification hierarchy shown in Figure 5 can be incrementally constructed from the eight ToH-problems.

5 DISCUSSION AND RELATED WORK

In this paper, four key requirements for speeding up hierarchical planning by learning abstract plans were derived from recent machine learning and planning literature. A new formal model of shared plan abstraction and the explanation-based S-PABS procedure were introduced to fulfill the first three requirements. According to the last requirement, an adaption of the idea of concept formation over explanations was further outlined as a method to incrementally construct a hierarchical classification of the abstract plans.

Unlike other well known techniques for learning search control rules for planning (PRODIGY) by explanation-based learning [Minton *et al.*, 1989] or inductive learning [Leckie and Zuckermann, 1991], S-PABS can acquire domain oriented problem decompositions rather then more or less restricted operator selection rules. Search control rules can guide the search in a single problem space but cannot reduce planning complexity by switching to an abstract problem description. On the other hand PRODIGY is able to learn from failed solution tracks which actually cannot be preformed by S-PABS.

Learning macro-operators has shown to speedup planning if the problem domain is serially decomposable [Korf, 1985] or satisfies the sparse solution space bias [Tadepalli, 1991]. For domains, that does not satisfy these requirements, learning and storing macro-operators can lead to the utility problem [Minton, 1990] due to the specificity and the limited applicability of learned knowledge. Abstract plans tend to overcome this problem since the knowledge representation is simplified through abstraction and the scope of utilization is increased.

Recently, a few approaches to plan abstraction have been proposed. In Knoblock's method for learning abstract planning spaces [Knoblock, 1989], abstraction always occurs by dropping sentences of the concrete world. This kind of abstraction is only a special case of the type of abstractions we allow. This restrictions can be characterized by limiting state abstraction mappings those, where $a(s) \subset s$ is satisfied. The major advantage of our model is, that it allows a totally different terminology to be introduced. Thereby it permits the construction of real domain oriented abstractions which require a shift of the representation language.

Plan abstractions how they are derived by PLANEREUS [Anderson and Farly, 1988] or by Tenenberg's approach [Tenenberg, 1986] can also be shown to be a special type of the abstraction created by S-PABS. The former systems use a taxonomic hierarchy of operations (which is learned in the case of PLANEREUS) to construct an abstract plan from a concrete plan. Plan abstraction occures by generalizing each operator according to the hierarchy. Therefore, the number of operations contained in the resulting abstract plan is not reduced. In our model, this type of abstraction can be described by the special sequence abstraction mapping $b(i) := i$.

Within the Soar framework, Unruh and Rosenbloom [Unruh and Rosenbloom, 1989] have proposed an abstraction technique which can be characterized as general weak method, in that it uses no domain-specific knowledge about how to perform abstractions. This is in constrast to our approach, since we want to draw power from the knowledge about useful domain specific abstractions which have been proven successful in human problem solving. The knowledge needs of our method demand for the development of special knowledge acquisition methods to transform expert experience into a formal descriptions of an abstract world and a generic abstraction theory. Case-oriented knowledge acquisition tools for planning domains [Schmalhofer *et al.*, 1991b; Bergmann and Schmalhofer, 1991; Schmidt and Schmalhofer, 1990] can help to fulfil these requirements of S-PABS.

Acknowledgements
I would like to thank Franz Schmalhofer and Stefan Boschert for helpful discussions and comments on the topic of plan abstraction. Andreas Dannenmann and Wolfgang Wilke provided very helpful comments on early versions of this paper. This research was supported by grant Schm 648/1 from "Deutsche Forschungsgemeinschaft".

References

[Anderson and Farly, 1988] J. S. Anderson and A. M. Farly. Plan abstraction based on operator generalization. In *Proceedings of the 7th International Conference on Artifical Intelligence*, pages 100–104, San Mateo, 1988. Morgan Kaufmann.

[Bergmann and Schmalhofer, 1991] R. Bergmann and F. Schmalhofer. Cecos: A case experience combination system for knowledge acquisition for expert systems. *Behavior Research Methods, Instruments and Computers*, 23:142–148, 1991.

[Bergmann, 1992a] R. Bergmann. Explanation-based learning for the automated reuse of programs. In *Proceedings of the IEEE-Conference on Computer Systems and Software Engineering, COMPEURO92. To appear*, 1992. (in press).

[Bergmann, 1992b] R. Bergmann. Knowledge acquisition by generating skeletal plans. In F. Schmalhofer, G. Strube, and Th. Wetter, editors, *Contemporary Knowledge Engineering and Cognition*, Heidelberg, 1992. Springer. (in press).

[Bergmann, 1992c] R. Bergmann. Learning plan abstractions. In *GWAI-92 16th German Workshop on Artificial Intelligence*, page in press. Springer-Verlag, 1992.

[Braverman and Russel, 1988] M. S. Braverman and S. J. Russel. Boundaries of operationality. In J. Laird, editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 221–234, San Mateo, CA, 1988. Morgan Kaufmann.

[Chien, 1989] S. A. Chien. Using and refining simplifications: Explanation-based learning of plans in intractable domains. In *Proceedings of the International Joint Conference on Artificial Intelligence-89*, volume 1, pages 590–595, Detroit, MI, 1989. Morgan Kaufmann.

[Fikes *et al.*, 1972] R. E. Fikes, P. E. Hart, and N. J. Nilsson. Learning and

executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Flann and Dietterich, 1989] N.S. Flann and T.G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4(2):187–226, 1989.

[Friedland and Iwasaki, 1985] P. E. Friedland and Y. Iwasaki. The concept and implementation of skeletal plans. *Journal of Automated Reasoning*, pages 161–208, 1985.

[Gennari *et al.*, 1989] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.

[Giordana *et al.*, 1991] A. Giordana, D. Roverso, and L. Saitta. Abstracting background knowledge for concept learning. In Y. Kodratoff, editor, *Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91*, pages 1–13, Berlin, 1991. Springer.

[Knoblock, 1989] C. A. Knoblock. A theory of abstraction for hierachical planning. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, pages 81–104, Boston, MA, 1989. Kluwer.

[Kolodner, 1987] J. L. Kolodner. Extending problem solver capabilities through case-based inference. In Morgan Kaufmann, editor, *Proceedings of the 4th International Workshop on Machine Learning*, pages 167–178, Irvine, Ca, 1987.

[Korf, 1985] R. E. Korf. Macro-operators: A weak method fr learning. *Artifical Intelligence*, 26:35–77, 1985.

[Korf, 1988] R.E. Korf. Optimal path-finding algorithms. In V. Kumar L. Kanal, editor, *Search in Artificial Intelligence*, pages 223–267. Springer, New York, 1988.

[Leckie and Zuckermann, 1991] C. Leckie and I. Zuckermann. Learning search control ruels for planning: An inductive apporach. In L. Birnbaum and G. Collins, editors, *Machine Learning, Proceedings of the 8th International Workshop (ML91)*, pages 422–426, San Mateo, CA, 1991. Morgan Kaufmann.

[Lifschitz, 1987] V. Lifschitz. On the semantics of strips. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, pages 1–9, Timberline, Oregon, 1987.

[Michalski and Y.Kodratoff, 1990] R. S. Michalski and Y.Kodratoff. Research in machine learning: Recent progress, classification of methods, and future directions. In Y. Kodratoff and R. S. Michalski, editors, *Machine learning: An artificial intelligence approach*, volume 3, chapter 1, pages 3–30. Morgan Kaufmann, San Mateo, CA, 1990.

[Minton *et al.*, 1989] S. Minton, J. G. Carbonell, C.A. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning:a problem solving perspective. *Artificial Intelligence*, 40:63–118, 1989.

[Minton, 1990] S. Minton. Quantitativ results concerning the utility of explanation-based learning. *Artifical Intelligence*, 42:363–391, 1990.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[Mozetic and Holzbaur, 1991] I. Mozetic and

C. Holzbaur. Extending explanation-based generalization by abstraction operators. In Y. Kodratoff, editor, *Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91*, pages 282–297, Berlin, 1991. Springer.

[Plaisted, 1981] D. Plaisted. Theorem proving with abstraction. *Artifical Intelligence*, 16:47–108, 1981.

[Sacerdoti, 1974] E.D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Schmalhofer *et al.*, 1991a] F. Schmalhofer, R. Bergmann, O. Kühn, and G. Schmidt. Using integrated knowledge acquisition to prepare sophisticated expert plans for their reuse in novel situations. In Thomas Christaller, editor, *GWAI-91 15th German Workshop on Artificial Intelligence*, pages 62–71. Springer-Verlag, 1991.

[Schmalhofer *et al.*, 1991b] F. Schmalhofer, O. Kühn, and G. Schmidt. Integrated knowledge aquisition from text, previously solved cases and expert memories. *Applied Artificial Intelligence*, 5:311–337, 1991.

[Schmidt and Schmalhofer, 1990] G. Schmidt and F. Schmalhofer. Case-oriented knowledge acquisition from texts. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, and M. van Someren, editors, *Current Trends in Knowledge Acquisition*, pages 302–312, Amsterdam, May 1990. IOS Press.

[Tadepalli, 1991] P. Tadepalli. A formalization of explanation-based macro-operator learning. In Morgan Kaufmann, editor, *Proceedings of the International Join Conference on Artifical Intelligence-91*, pages 616–622, 1991.

[Tenenberg, 1986] J. Tenenberg. Preserving consistency across abstraction mappings. In MCdermott, editor, *Proceedings of the 6th International Conference on Artifical Intelligence*, pages 76–80, Philadelphia, PA, 1986.

[Tenenberg, 1987] J. Tenenberg. Preserving consistency across abstraction mappings. In J. McDermott, editor, *Proceedings of the 10th International Conference on Artifical Intelligence*, pages 1011–1014, Los Altos, CA, 1987. Morgan Kaufmann.

[Unruh and Rosenbloom, 1989] A. Unruh and P.S Rosenbloom. Abstraction in problem solving and learning. In *Proceedings of the International Join Conference on Artifical Intelligence-89*, pages 590–595, Detroit, MI, 1989. Morgan Kaufmann.

[Yoo and Fisher, 1991] J. Yoo and D. Fisher. Concept formation over explanations and problem-solving experience. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence*, volume 2, pages 630–637, San Mateo, CA, 1991. Morgan Kaufmann.