Learning Plan Abstractions

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. Abstraction has been identified as a powerful means to reduce the complexity of planning problems. In this paper, a formal model and a method are described for learning abstract plans from concrete plans. In this model, the problem of plan abstraction is decomposed into finding a *state abstraction mapping* and a *sequence abstraction mapping*. The definition of an *abstract planning world* and a *generic state abstraction theory* allows a totally different terminology to be introduced for the description of the abstract plans. Thereby, not only generalizations but real abstractions can be established which require a shift in representation. With the described explanation-based learning procedure PABS, deductively justified abstractions are constructed which are tailored to the application domain described by the theory. The proposed abstraction methodology is applied to learn from machine-oriented programs. An abstract program can be acquired on a higher programming level which is characterized through the use of abstract data types instead of machine-oriented data representations.

1 Introduction and Motivation

Michalski and 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. In recent machine learning literature a few methods have been proposed to generate abstractions of concepts by use of domain theories. Giordana, Saita and Roverso [1991] have introduced semantic abstraction to deductively change the representation of an example while reducing its level-of-detail. Mozetic and Holzbaur's [1991] abstraction operators allow the elimination of details in a generalized concept description obtained by explanation-based learning [Mitchell et al., 1986]. Both methods are restricted to the description of static concepts and cannot handle planning problems.

Abstraction has been identified as a powerful method to overcome the computational intractability for planning. In hierarchical planning [Sacerdoti, 1974; Wilkins, 1984] abstract planning spaces are introduced in addition to the concrete planning level, to allow a solution to a planning problem to be constructed at a less detailed level before refining it towards a concrete solution. If an optimal hierarchy of abstraction layers can be constructed, hierarchical planning can reduce the planning complexity from exponential to linear time [Korf, 1988; Knoblock, 1989]. Unfortunately, this analysis is based on several assumptions according the structure of the problem domain which are very unlikely to be fulfilled for real-world domains (as already recognized by Knoblock). One of the most restrictive assumptions is the

downward solution property, which states that an abstract solution can really be refined at the next detailed layer. For real-world planning problems, such an optimal hierarchy of abstraction levels can usually not be found or even be constructed automatically (as proposed by Knoblock [1989]) to achieve a dramatic complexity reduction by hierarchical planning. Consequently, hierarchical planning cannot be called the ultimate approach for solving practical planning problems.

In human problem solving, many complicated planning problems (e.g. in mechanical engineering or in programming) are not solved from scratch. More often, previous successful solutions are reused in a more or less abstract manner [Spur, 1979; Jeffries, 1988; Schmalhofer et al., 1992]. Besides case-based reasoning [Kolodner, 1987], skeletal plan refinement [Friedland et al., 1985] is an approach which closely relates to this observation. For planning, generalized plans are retrieved from a data-base and refined according to a new problem specification. A library of abstract plans which provide a heuristic decomposition of a complex planning problem into several smaller subproblems can maintain some of the complexity results obtained from the analysis of hierarchical planning even if the Knoblock's limiting assumptions do not hold in a problem domain. An appropriate set of abstract plans avoids the construction of a problem solution at an abstract level from scratch. This is especially important, if a large number of abstract operations have to be supported in a domain, so that planning at the abstract level becomes a complicated task too. Even if an abstract plan can be found by search, it cannot be guaranteed that such a plan describes a useful problem decomposition so that it can really be refined to a concrete solution.

However, abstract plans which have been proven useful in previous problem solving experience, promise to be successfully applicable for the new problems as well. But therefore abstractions are needed which use a terminology that is already established in the problem domain. Abstraction by simply dropping conditions of the concrete level as usually performed by hierarchical planning, is not sufficient for that purpose.

For learning such abstract plans, from concrete planning cases, no method has been established until now. This article proposes the extension of methods of knowledge intensive learning to produce domain tailored abstractions, but not only generalizations. In machine learning literature, many explanation-based learning approaches have been developed to learn from plans [Fikes et al., 1972; Minton et al., 1989; Schmalhofer et al., 1991; Bergmann, 1992a] or programs [Bergmann, 1992b] to allow them to be reused in novel situations. All these approaches are able to construct generalizations of plans but they are unable to result in abstractions because no knowledge about abstract operations is contained in the domain theory. To overcome this problem, methods must be found to efficiently utilize domain knowledge about abstract operations and planning goals in order to produce deductively justified abstractions.

In the next section a formal model of plan abstraction is introduced. A five-phase procedure for constructing abstract plans is proposed in section three. An example application of this method for abstracting a machine level program is demonstrated and finally the limitations of the approach and further research perspectives are discussed.

2 Formal Model of Plan Abstraction

The core idea of abstraction is to achieve a reduction of the level-of-detail of a description and thereby to change the representation language into a simpler language than the original [Plaisted, 1981; Tenenberg, 1987; Giordana et al., 1991; Mozetic et al., 1991]. A plan or a program is composed of operations which are executed in a specific order and thereby change the state of a world. Therefore, plan 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.

2.1 The Problem of Abstraction

Since the goal is to construct domain tailored plan abstractions, a concrete and an abstract planning space is assumed to be defined. Each of the two planning spaces is represented as a STRIPS world. Note, that a totally different terminology will usually be employed to represent the concrete and the abstract world. 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 $<P\alpha, D\alpha, A\alpha>_{\alpha} \in Op$, 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 $\mathbb{S} = 2^R$ be the set of all states of the world. A *plan p* in a world W is a sequence $(o_1,..., o_n)$ of operators from Op. In a world W an initial state $s_0 \in \mathbb{S}$ and a plan $p=(o_1,..., o_n)$ *induce* a sequence of states $s_1 \in \mathbb{S},...,s_n \in \mathbb{S}$ where $s_{i-1} \cup T \vdash P_{o_i}$ and $s_i = (s_{i-1} \setminus D_{o_i}) \cup A_{o_i}$. Two plans $p=(o_1,...,o_n)$ and $p'=(o_1',...,o_n')$ are called *equivalent* iff in every state $s_0=s_0' \in \mathbb{S}$ follows that $s_i = s_i'$, $i \in \{1,...,n\}$ for the states induced by the plans.

The abstraction of a plan requires a reduction of the level-of-detail for the description of the world states and a reduction of the number of states that describe the problem solution. Both changes in the level of detail imply a change of the representation for states as well as for operators. Therefore, 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 assumed to be given. The problem of plan abstraction can now be described as transforming a plan p_C from the concrete world W_C into a plan p_a in the abstract world W_a, with several conditions being satisfied. This transformation can formally be decomposed into two mappings, a *state abstraction mapping* $\mathfrak{a}$, and a *sequence abstraction mapping* $\mathfrak{b}$ as follows:

Definition 2: A *state abstraction mapping* $\alpha: \mathfrak{S}_C \rightarrow \mathfrak{S}_a$ is a mapping from $\mathfrak{S}_C$, the set of all states in the concrete world, to $\mathfrak{S}_a$, the set of all states in the abstract world, that satisfies the following conditions:
a) If $sc \cup T_C$ is consistent then $\alpha(sc) \cup T_a$ is consistent for all $sc \in \mathfrak{S}_C$.
b) If $sc \cup sc' \cup T_C$ is consistent then $\alpha(sc \cup sc') \supseteq \alpha(sc) \cup \alpha(sc')$ for all $sc, sc' \in \mathfrak{S}_C$.

Definition 3: A *sequence abstraction mapping* $\beta: N \rightarrow N$ relates an abstract state sequence $(sa_0,...,sa_n)$ to a concrete state sequence $(sc_0,...,sc_m)$ by mapping the indices i of the abstract states sa_i onto the indices j of the concrete states sc_j, so that the following properties hold:
a) $\beta(0) = 0$ and $\beta(n) = m$: 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) $\beta(u) < \beta(v)$ 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 p_a is an *abstraction* of a plan p_C if there exists a state abstraction mapping $\alpha: \mathfrak{S}_C \rightarrow \mathfrak{S}_a$ and a sequence abstraction mapping $\beta: N \rightarrow N$, so that:
If p_C and an initial state sc_0 induce the state sequence $(sc_0,...,sc_m)$ and $sa_0 = \alpha(sc_0)$ and $(sa_0,...,sa_n)$ is the state sequence which is induced by sa_0 and the abstract plan p_a, then
$\alpha(sc_{\beta(i)}) = sa_i$ holds for all $i \in \{1,...,n\}$.

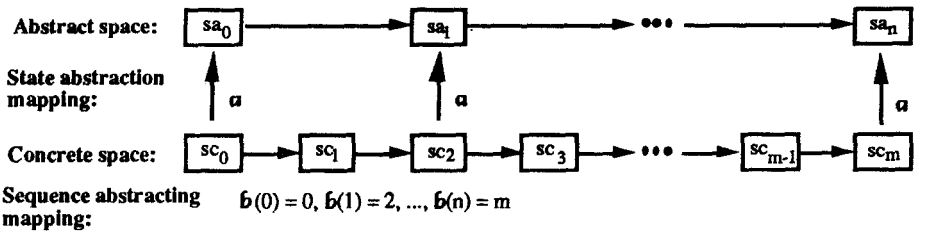

Figure 1: Demonstration of the plan abstraction methodology

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 α into states of the abstract space. The sequence abstraction mapping β maps the indices of the abstract states to the indices of the corresponding states in the concrete world. It becomes clear, that plan abstraction is defined by a pair of abstraction mappings (α,β). 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 State Abstraction Mappings with Deductive Justifications

For the construction of domain tailored abstraction mappings, a justification of the state abstraction mappings by knowledge of the domain is necessary. Such domain knowledge should state which kinds of useful abstractions usually occur in a domain, and how they can be defined in concrete terms. Thereby, 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 [1991] relate atomic formulae of an abstract Language L' to terms of a corresponding concrete language L. Similarly, in this article, a generic state abstraction theory T_g is defined as a set of axioms of the form $\Psi \leftrightarrow D_1 \vee D_2 \vee ... \vee D_n$, where Ψ is an essential sentence of the abstract world and $D_1,...,D_n$ are conjunctions of sentences of the concrete world. For the deductively justified state abstraction mapping α it is required that: if $\Psi \in \alpha(sc)$ then $sc \cup T_C \cup T_g \vdash \Psi$. The reverse implication, namely that every essential sentence for which $sc \cup T_C \cup T_g \vdash \Psi$ holds is an element of $\alpha(sc)$, is not demanded, since a minimal consistent state abstraction mapping (according to the $\supset$ - relation) should be reached.

3 PABS: A Procedure for Constructing Plan Abstractions

As described in section 2, the task of constructing an abstraction of a plan can be seen as the problem of finding a deductively justified state abstraction mapping and a sequence abstraction mapping so that a related plan in the abstract world exists. This section describes PABS (**P**lan **Ab**straction), a five-phase procedure which supports the automatic construction of plan abstractions. In the first phase of PABS, the execution of the concrete plan p_C is simulated and the sequence of the induced states in the concrete planning space is computed. In the second phase, for each of these states, an abstract description is derived by utilizing the generic abstraction theory. In third phase, for each pair of the abstract planning states, it is checked, whether there exists an abstract operation which is applicable and which the transforms abstract states into each other. A directed graph of candidate abstract operations is finally constructed in this phase. In phase four, a complete and consistent path from the initial abstract state to the final abstract state is searched. With this path a state abstraction mapping and a sequence abstraction mapping can be determined. Finally, explanation-based generalization is applied in phase five of PABS, to generalize the sequence of abstract operations into an abstract plan with corresponding application conditions.

3.1 Phase-I: Simulation of the Concrete Plan

By simulating the execution of the concrete plan p_C, the state sequence $(sc_1,...,sc_m)$ is computed (see Figure 1), which is induced by the plan p_C and a given initial state sc_0. 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 an explanation for the effects caused by the operations. Such a step is also performed in procedures for plan generalizations [Fikes, 1972; Schmalhofer et al., 1991; Bergmann, 1992a] and for constructing programming schemes from symbolic traces [Bergmann, 1992b].

3.2 Phase-II: Construction of State Abstractions

The second step 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 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 for $\Psi \in R_a$: $sc_i \cup T_c \cup T_g \vdash \Psi$ holds, then Ψ is included into the state abstraction sa'_i. If both, the concrete and abstract world descriptions consist of a finite set of essential sentences, and if the type of the theories T_c and T_g allow to decide the truth of a sentence, then the abstract state descriptions can be generated automatically. By allowing some user support in this phase more concise abstract state descriptions may be obtained, thereby reducing the overall search in subsequent phases.

3.3 Phase-III: Constructing State Transitions with Abstract Operations

The goal of the third phase is to identify candidate abstract operations for the abstract plan. For each pair (sa'_u, sa'_v) with $u < v$, it is checked, if there exists an abstract operation O_α described by $<P_\alpha, D_\alpha, A_\alpha>$ which is applicable in sa'_u and which transforms sa'_u into sa'_v. If $sa'_u \cup T_a \vdash P_\alpha$ 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.

3.4 Phase-IV: Establishing a Consistent Path of Abstract Operations

From the constructed graph, a complete and consistent path $p_a = (o_1,...,o_n)$ from the initial abstract state sa'_0 to the final state sa'_m is searched. The consistency requirement for a path 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 p_a is applicable, which means that the application conditions 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.

Through the selected path $p_a = (o_1,...,o_n)$, a sequence abstraction mapping β is implicitly defined through: $\beta(i) := j$ iff the i-th abstract operation of the path connects state sa'_k with sa'_j. Additionally, a corresponding sequence of minimal abstract state descriptions $sa_0,...,sa_n$ with $sa'_{\beta(i)} \supseteq sa_i$ is determined. These minimal abstract states contain exactly those sentences which are necessary to guarantee the applicability of the operations in the path. Thereby $sa_0,...,sa_n$ represents the state sequence which is induced by p_a. Note that a state abstraction mapping α is also implicitly defined

through: $\alpha(sc) = \{\Psi \mid sc \cup T_C \cup T_g \vdash \Psi$ and $\exists$ i$\in \{1,...,n\}$ with $\Psi \in sa_i\}$. So, the abstract plan is completely defined according to definition 4.

For realistic planning situations, the abstract world description will likely consist of a large number of essential sentences and abstract operations at many different levels of abstraction. Therefore it is expected that more than one plan abstraction can be found according to the above definition. In this situation a heuristic criterion (e.g. a good ratio n/m of the number of abstract to concrete states) can be chosen or the user has to decide which of the paths from the initial to the final state represent the intended abstraction. Additionally, the number of possible abstract plans can be reduced to those which are valid for a larger set of concrete plans. Such *shared abstract plans* [Bergmann, 1992c] are supposed to be more useful for solving the future planning problems.

3.5 Phase-V: Constructing the Final Abstract Plan with Application Conditions

From the path p_a and the dependency network which justifies its consistency, a generalization 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 et al., 1988] for the learned concept is determined by the essential sentences and the operators available in the abstract world. Within this subexplanation, the remaining derivations are generalized by standard goal regression as used by 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.

4 An Example: Abstracting a Machine Level Program

To demonstrate the introduced terminology and the proposed method, the abstraction of an example program is followed through the five phases of PABS. The goal is to find an abstraction for a computer program which is formulated in terms of a machine language and contains only simple assignments of logical operations on boolean variables. For that purpose, the concrete and the abstract world descriptions as well as the generic state abstraction theory must be defined first. Figure 2[1] shows a fraction of this domain knowledge. The concrete world describes the semantics of the operations of the machine language (and,or,...) in which the program, to be abstracted, is written. For example "and(x,y,z)" specifies the assignment "z := x and y". In the formalization, the precondition states that the input-arguments of the operations must have already been bound to a boolean value (t1, t2). The sentence in the add-list

[1] x,y,z,t1,..,t4,s1,...,s3 stand for variables.

specifies the result (t3) of the operations ,while the delete-list removes all earlier assignments to the output variable that differ from the new value[2].

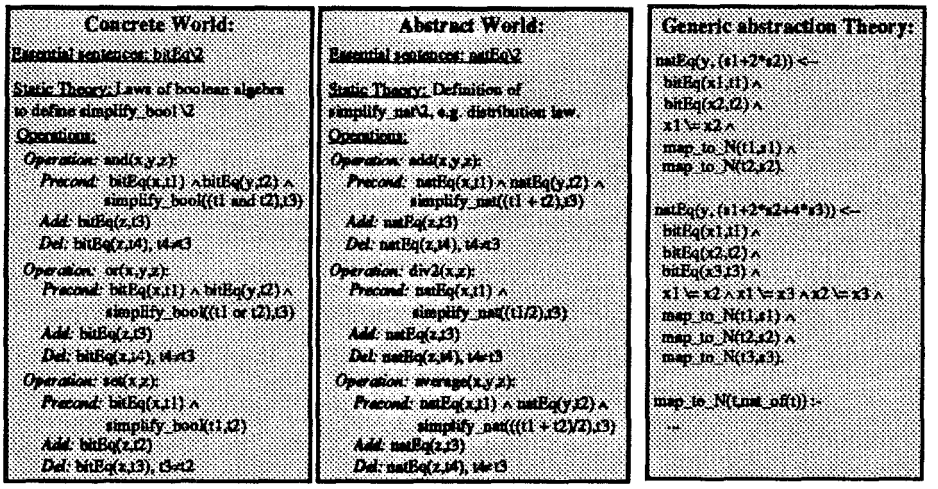

Figure 2: Domain knowledge for plan abstraction

The abstract world determines the high-level operations of the domain (add, div2,...) from which the abstraction is composed. These are some numeric operations on natural numbers. The generic abstraction theory represents abstract data types (natural numbers) and their implementation (two or three-bit values).

$P_{average}$:			
	1. xor(x1,x3,x5)	5. and(x2,x4,x7)	9. or(x7,x9,x7)
	2. and(x1,x3,x8)	6. and(x2,x8,x9)	10. set(x6,x5)
	3. xor(x2,x4,x6)	7. or(x7,x9,x7)	11. set(x7,x6)
	4. xor(x6,x8,x6)	8. and(x4,x8,x9)	12. set(0,x7)

Figure 3: $P_{average}$: An example program for abstraction

The concrete plan to be abstracted is a computer program which consists of 12 sequential steps as shown in Figure 3. This program computes the average of two natural numbers which are represented as two-bit values. The variables x1,...,x4 represent the input values, x5,...,x7 represent the output value, and x8, x9 are additional auxiliary variables.

Figure 4 shows the five phases of PABS which are applied for the abstraction of the example program: In phase-I, the simulation of $P_{average}$ through symbolic execution [Bergmann, 1992b] results in the concrete state sequence $sc_0,...,sc_{12}$. Each of the states is described by the actual value of the relevant boolean variables (bitEq) which depend from the symbolic program input $t_1,...,t_4$. In the goal state sc_{12}, the program output specification is derived. In phase-II, each of the concrete states is abstracted by deriving essential sentences of the abstract state (the "natEq" predicates) by utilizing the generic abstraction theory. The generic abstraction theory describes

---

2 This operator formalization is a slight extension of the traditional STRIPS-representation.

the possible kinds of data abstraction by representing natural numbers as bit-vectors. The states sa'0,...,sa'12 computed in phase-II contain the derived descriptions in terms of the abstract world. For example, the sentence "natEq(y1, nat_of(t1)+2*nat_of(t2))" in state sa'0 expresses, that there exists a natural number y1 whose value is computed from the value of the bits x1 and x2. The first clause of the generic abstraction theory from Figure 2 was applied to derive this data abstraction.

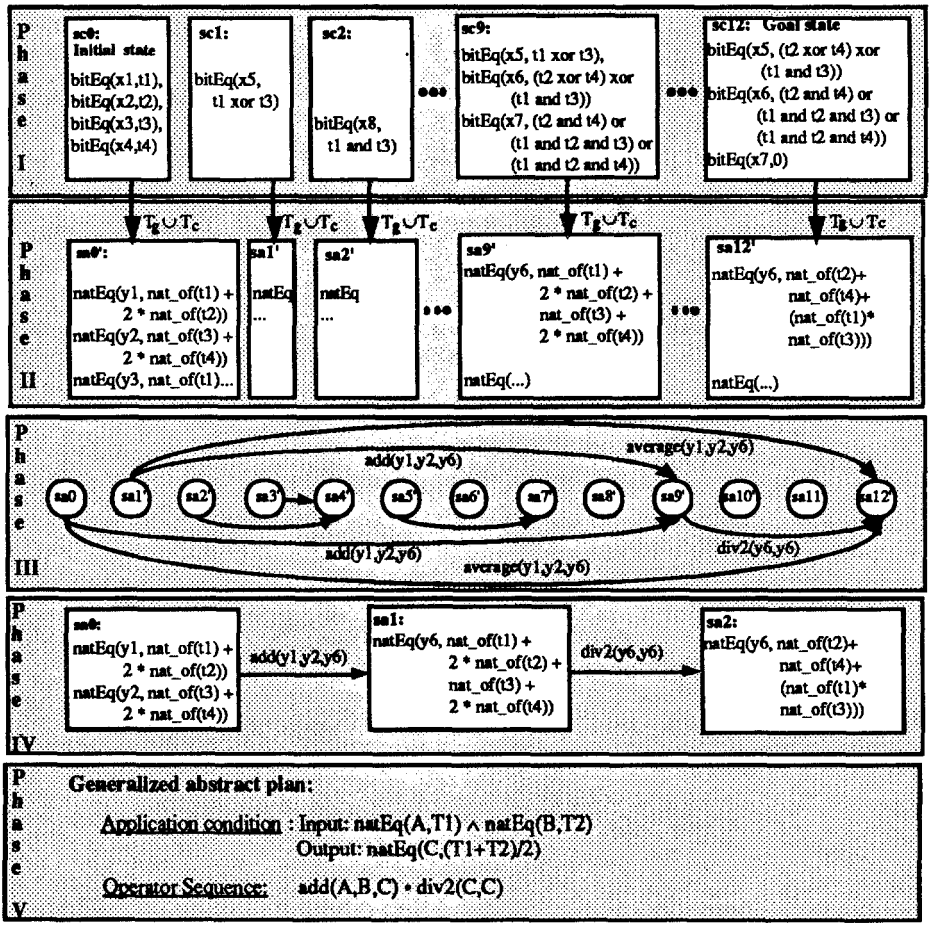

Figure 4: The five phases of PABS for the abstraction of P average

In phase-III, the graph of the candidate abstract operations is expanded. For example, the link from sa'0 to sa'9 labeled with "add(y1,y2,y6)" specifies that the add-operation is applicable in the abstract state sa'0 and creates some of the effects (here: natEq(y6,...)) which are contained in the state sa'9. Using this graph, a consistent path from sa'0 to sa'12 is searched in phase-IV. In the example the path sa'0-->sa'9-->sa'12 is selected to describe the intended abstraction. Note that the path sa'0-->sa'12 is also a consistent path which is assumed to be rejected. Through the selection of the path, a

sequence abstraction mapping $\mathfrak{b}$: N → N is defined by: $\mathfrak{b}(0) = 0$, $\mathfrak{b}(1) = 9$ and $\mathfrak{b}(2) = 12$. The corresponding state abstraction mapping $\mathfrak{a}$ is defined as follows[3]:

$\mathfrak{a}$({ bitEq(x1,T1), bitEq(x2,T2) }) = {natEq(y1, nat_of(T1)+2*nat_of(T2)}

$\mathfrak{a}$({ bitEq(x3,T1), bitEq(x4,T2) }) = {natEq(y2, nat_of(T1)+2*nat_of(T2)}

$\mathfrak{a}$({ bitEq(x5,T1), bitEq(x6,T2), bitEq(x7,T3) }) =

{natEq(y6,nat_of(T1)+2*nat_of(T2)+4*nat_of(T3))}

Note that $\mathfrak{a}$ exactly creates those preconditions and effects that are required for the abstract operations in the path. In phase-V, an explanation-based generalization computes the final abstract plan. This generalization is responsible, for example for the replacement of the constant "nat_of(t1)+2*nat_of(t2)" by the variable A. The completely abstracted plan expresses that whenever the average of the natural numbers T1 and T2 has to be computed, the sequence of adding T1 and T2 and dividing the result by two is an appropriate abstract plan. This is exactly the plan which a programmer had in mind when he designs the program $P_{average}$.

5 Discussion

Abstraction has been recognized as an important mechanism in planning as well as in machine learning. In this paper a new formal model for the description of plan abstraction has been presented. The PABS procedure for learning deductively justified abstract plans from concrete plans has been delineated and demonstrated for a realistic example. One major advantage of the proposed method is, that it allows a totally different terminology to be introduced for abstraction. Thereby it permits the construction of real domain oriented abstractions which require a shift in the representation language.

This is in contrast to Knoblock's method for learning abstract planning spaces [Knoblock, 1989], where abstraction always occurs by dropping sentences of the concrete world. This kind of abstraction is only a special case of the type of abstractions, allowed by the proposed model. The restrictions can be characterized by limiting state abstraction mappings to those, where $\mathfrak{a}(s) \subset s$ is satisfied.

Unlike other well known techniques for explanation-based learning of search control rules for planning (PRODIGY) [Minton et al., 1989], PABS can acquire domain-oriented problem decompositions rather than 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.

Plan abstractions how they are derived by PLANEREUS [Anderson et al., 1988] or by Tenenberg's approach [Tenenberg, 1986] can also be shown to be a special type of the abstractions created by 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 occurs by generalizing each operator according to the hierarchy. Therefore, the number of operations contained in the resulting abstract plan is not reduced. In PABS, this type of abstraction can be described by a special sequence abstraction mapping defined by b(i) := i.

Within the Soar framework, Unruh & Rosenbloom [1989] have proposed an abstraction technique which they characterize as a general weak method, in that it uses no domain-specific knowledge about how to perform abstractions. This is in contrast

---

[3] T1,..,T3 stand for boolean variables.

to the PABS approach, since PABS draws power from the knowledge about useful domain specific abstractions which have been proven to be successful in human problem solving. On the other hand, the requirement of this abstract domain knowledge may be difficult to fulfill for domains which are less well understood as programming. In this case, interactive knowledge acquisition methods [Schmalhofer et al., 1991] can be applied for eliciting problem class descriptions [Bergmann et al., 1991] and hierarchies of abstract operations [Schmidt, 1992] from a domain expert to establish the concrete and the abstract world descriptions. This becomes possible, since Korf's requirements [Korf, 1988; Knoblock, 1989] on abstract world models need not to be fulfilled for learning abstract plans.

Acknowledgment

I would like to thank Franz Schmalhofer, Stefan Boschert and Andreas Dannenmann for helpful discussions and comments on earlier drafts of this paper as well as Wolfgang Wilke for implementing the prototypical learner. This research was supported by grant Schm 648/1 from "Deutsche Forschungsgemeinschaft".

References

Anderson, J. S. and Farley, A. M. (1988). Plan abstraction based on operator generalization. In *Proceedings of the 7th National Conference on Artificial Intelligence* (pp. 100-104), Morgan Kaufmann, San Mateo.

Bergmann, R. and Schmalhofer, F. (1991). CECoS: A case experience combination system for knowledge acquisition for expert systems. *Behavior Research Methods, Instruments and Computers*, 23, 142-148.

Bergmann, R. (1992a). Knowledge acquisition by generating skeletal plans. In Schmalhofer, F., Strube, G., and Wetter, T. (Eds.). *Contemporary Knowledge Engineering and Cognition* (pp. 125-134). Heidelberg: Springer.

Bergmann, R. (1992b). Explanation-based learning for the automated reuse of programs. In *Proceedings of the IEEE-Conference on Computer Systems and Software Engineering, COMPEURO92, pp. 109-110.*

Bergmann, R. (1992c). Learning abstract plans to speed up hierarchical planning. In *Proceedings of the ML'92-Workshop "Knowledge Compilation and Speedup Learning"*, Aberdeen, Scotland.

Braverman, M. and Russell, S. (1988). Boundaries of operationality. *Proceedings of the 5th International Conference on Machine Learning* (pp. 221-234). Ann Arbor, MI: Morgan Kaufmann.

Fikes, R.E., Hart, P.E., and Nilsson, N.J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence*, 3, 251-288.

Friedland, P.E. and Iwasaki, Y. (1985). The concept and implementation of skeletal plans. *Journal of Automated Reasoning.*

Giordana, A., Roverso, D., and Saitta, L. (1991). Abstracting background knowledge for concept learning. In Kodratoff, Y. (Ed.), *Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91* (pp. 1-13). Berlin: Springer.

Jeffries, R. Turner, A.A., and Polson, P.G. (1988). The processes involved in designing software. In J.R. Anderson (Ed.) *Cognitive Skills and their Acquisition* (pp. 255-283). Hillsdale, NJ: Lawrence Erlbaum.

Knoblock, C.A. (1989). A theory of abstraction for hierarchical planning. In *Proceedings of the Workshop on Change of Representation and Inductive Bias* (pp. 81-104). Boston, MA: Kluwer.

Kolodner, J. L. (1987). Extending problem solver capabilities through case-based inference. Proceedings of the 4th International Workshop on Machine Learning (pp. 167-178). Irvine, CA: Morgan Kaufmann.

Korf, R.E. (1988). Optimal path-finding algorithms. In Kumar (Ed.), *Search in Artificial Intelligence* (pp. 223-267). New York: Springer.

Lifschitz, V. (1987). On the semantics of STRIPS. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop* (pp. 1-9). Timberline, Oregon.

Michalski, R.S. and Kodratoff, Y. (1990). Research in machine learning: Recent progress, classification of methods, and future directions. In Kodratoff, Y. and Michalski, R.S. (Eds.), *Machine learning: An Artificial Intelligence Approach* (Vol. 3, pp. 3-30). San Mateo, CA: Morgan Kaufmann.

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

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

Mozetic, I. and Holzbaur, C. (1991). Extending explanation-based generalization by abstraction operators. In Kodratoff, Y. (Ed.), *Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91* (pp. 282-297). Berlin: Springer.

Plaisted, D. (1981). Theorem proving with abstraction. *Artificial Intelligence*, **16**, 47-108.

Sacerdoti, E.D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, **5**, 115-135.

Schmalhofer, F., Bergmann, R., Kühn, O., and Schmidt, G. (1991). Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations. In Christaller, T. (Ed.), *GWAI-91: 15. Fachtagung für Künstliche Intelligenz*, Informatik Fachberichte (pp. 62-71). Springer-Verlag.

Schmalhofer, F., Globig, C., and Thoben, J. (1992). The refitting of plans by a human expert. In Schmalhofer, F., Strube, G., and Wetter, T. (Eds.). *Contemporary Knowledge Engineering and Cognition* (pp. 122-131). Heidelberg: Springer.

Schmidt, G. (1992). Knowledge acquisition from text in a complex domain. In *Proceedings of the Fifth International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, Paderborn, Germany (in press).

Spur, G. (1979). Produktionstechnik im Wandel. München: Carl Hanser Verlag.

Tenenberg, J. (1986). Planning with abstraction. In *Proceedings of the 6th National Conference on Artificial Intelligence* (pp. 76-80), Philadelphia, PA.

Tenenberg, J. (1987). Preserving consistency across abstraction mappings. In McDermott, J. (Ed.), *Proceedings of the 10th International Conference on Artificial Intelligence* (pp. 1011-1014). Los Altos, CA: Morgan Kaufmann.

Unruh, A. and Rosenbloom, P. S. (1989). Abstraction in problem solving and learning. In *Proceedings of the International Joint Conference on Artificial Intelligence-89* (pp. 590-595). Detroit, MI: Morgan Kaufmann.

Wilkins, D. (1984). Domain-independent Planning: Representation and plan generation. *Artificial Intelligence*, **22**, 269-301.