Current Trends in AI Planning
C. Bäckström and E. Sandewall (Eds.)
IOS Press, 1994

Knowledge level planning

J. Stuart Aitken
DFKI
Erwin-Schrodinger-Strasse,
Postfach 2080,
D-6750 Kaiserslautern,
Germany
email: aitken@dfki.uni-kl.de
phone: +49 631 205-343

Nigel Shadbolt
A.I. Group, Dept of Psychology,
University of Nottingham,
University Park,
Nottingham,
England NG7 2RD
email: nrs@psyc.nott.ac.uk
phone: +44 602 515317

Abstract.
This paper describes how a proven planning technique can be employed within a *structured approach* to knowledge based system development (KADS), via the definition of a model of expertise.
This paper proposes a KADS model of expertise for hierarchical skeletal plan refinement. This model is based on a conceptual analysis by Kühn and Schmalhofer (1992) of skeletal planning as described in Friedland and Iwasaki (1985). The separation of domain and control knowledge is well defined in the KADS problem solving model and we explore the implications of this on the choice of domain representation formalism. We show how sound formalisms such as a temporal logic can be used for knowledge representation at the domain level.

1. Introduction.

Friedland and Iwasaki (1985) observed that planning problems were often solved by the modification of a previously successful solution. In particular they identified a two step procedure often used by molecular biologists to plan experiments: firstly a past solution (a skeletal plan) is associated with the current problem and secondly, the skeletal plan is refined to derive an appropriately modified plan. Past solutions may be described at various degrees of abstraction and this entails the process of hierarchical refinement in order to derive a concrete plan which solves the current problem.

There is a strong conceptual method within the analysis of Friedland and Iwasaki, however this method is not expressed in a sufficiently explicit form for it to be immediately used as a model of expertise with the KADS approach to KBS development (Wielinga, B.J. Schreiber, A.T. Breuker, J.A. (1992)). It is the aim of this paper to specify such a model of expertise based on Friedland and Iwasaki's method. The model of expertise can be viewed as a bridge from a proven planning technique to the structured approach to knowledge based system development of KADS.In the KADS methodology, planning is viewed as a synthetic task and to-date there are few validated models of expertise for synthetic tasks, therefore we believe the model proposed in this paper to be of value in this regard. There are other well known approaches to planning which also appear to have a strong underlying conceptual method and hence may yield useful models of expertise, means-end analysis, Newell and Simon (1963) for example.

The proposed inference structure for plan refinement represents the reasoning processes and knowledge structures that Friedland and Iwasaki identify as being relevant to Hierarchical Skeletal Plan Refinement (HSPR). The inference structure does not describe the control strategy of Iwasaki's SPEX planner, rather, the inference structure represents the elemental reasoning processes with no reference to the control issue. The KADS methodology maintains a clear distinction between task knowledge (or control structures), the inference structure and the domain knowledge. There is also a distinction of domain and two types of control knowledge in SPEX, however a comparison of the two architectures is outside the scope of this paper.

We describe an application in which temporal logic is used to define the effects of an action. The planning process therefore creates a *logical model* of the application at each point of planned time[1]. This is a significant aspect of our approach as it allows the use of a formal calculus with sound semantics at the domain layer. The use of formal calculi in planning has remained a topic of interest in AI, despite the discouraging analysis of the computational properties of reasoning in formal systems (eg. Chapman (1987)). This paper aims to show the advantages of a clear distinction between domain and control knowledge.

The remainder of this paper has the following structure, in Section 2 the inference structure for hierarchical skeletal plan refinement is specified, in Section 3 an example of the use of this structure in the domain of production planning is presented. Section 3 includes a typical example of the trace of the solution to a planning problem. Some conclusions of this work are drawn in Section 4.

2. An inference structure for hierarchical skeletal plan refinement

HSPR begins with the selection of a skeletal plan and then proceeds to construct a concrete plan through the refinement of the abstract operators of the skeletal plan. Our proposal for an inference structure for HSPR remains true to the central ideas of hierarchical plan refinement while emphasising the role of a state based description of the world. Of particular importance in state-based planning is the construction of a temporal model which describes the state of the world at each point in time within the plan. The temporal model is one component of the inference structure we propose, illustrated in Figure 1.

The inference structure is composed of meta-classes, knowledge theories and static domain knowledge. Meta-classes define the role in the problem solving process of the concepts or structures of the domain. The relationship between meta-classes is defined by knowledge sources which specify a primitive inference step which has specified input and output roles. Knowledge sources may also take static domain knowledge as an input. In Figure 1 meta-classes and static domain knowledge are indicated by rectangles and knowledge sources are indicated by ovals. The inference structure shows the flow of data (by the direction of the arrows) but does not indicate control knowledge.

This inference structure describes the refinement of a skeletal plan, the heuristic

[1]The choice of domain ontology is actually independent of the definition of the inference structure (the model of expertise)

process of the selection of a skeletal plan (which clearly must preceeed refinement) is described in Kühn and Schmalhofer (1992).

The `select` knowledge source has the task of selecting the next operator of the *skeletal plan* for refinement. Selection is based on past selections and on ordering criteria. For example it is preferable to plan the re-tooling operations prior to cutting operations - as the former are more constrained. The outcome of `select` is the instantiation of the *selected operator* meta-class.

The process of refining the *selected operator* is done with reference to the *abstract operator - subplan hierarchy*. Each abstract operator has a daughter node on the hierarchy or is a terminal node. Each terminal node has a corresponding concrete subplan. Therefore refining an abstract operator requires descending the hierarchy to a terminal node and then finding the corresponding concrete subplan. The higher up the hierarchy an abstract operator is, the greater the space of concrete subplans to be considered will be. The outcome of this procedure, defined by the `match` knowledge source, is the identification of a *selected subplan* (consisting of one or more concrete operators) which can then be considered for inclusion in the *concrete plan*.

The *selected subplan* may only be added to the *concrete plan* if the preconditions or postconditions of the operator definition rules match the state of the world as defined by the temporal model. If the preconditions of a rule match the state of the world at T1, for example, then the temporal model can be updated by defining the post conditions to be true at T2. In this case the *selected subplan* can be appended to the *concrete plan*. If preconditions of the rules defining the operators of *selected subplan* do not match the temporal model then the operators cannot be applied. In this case the procedure of refining the abstract operator must be repeated. The `apply` knowledge source tests the *selected subplan* and updates the *concrete plan* and the *temporal model* where appropriate.

The *temporal model* can be tested for correctness by the *evaluate* knowledge source. If the temporal model defines a route from the initial state to the goal state then the plan will be assigned to the *verified model* meta-class. This meta-class is one of the inputs to the `instantiate` knowledge source which completes the *concrete plan* by instantiating any remaining variables. This completes the refinement of the skeletal plan.

The inference structure provides guidance in the knowledge acquisition process by specifying a definite model of problem solving for the knowledge engineer to work with. The knowledge engineer must first choose a problem solving model which is appropriate for the task under consideration. Having made this choice the model then guides the engineer in the types of knowledge that must be specified. For example, a set of proven skeletal plans, a hierarchy of operators and a set of rules defining the effects of concrete operators are the minimum requirements for the application of the inference structure described above. The knowledge engineer must acquire this knowledge from expert sources. It may be that during this process the problem solving model is modified or it may be discovered that the initial choice was incorrect. It both of these cases, it is argued, that the guidance in the initial stages of acquisition is beneficial.

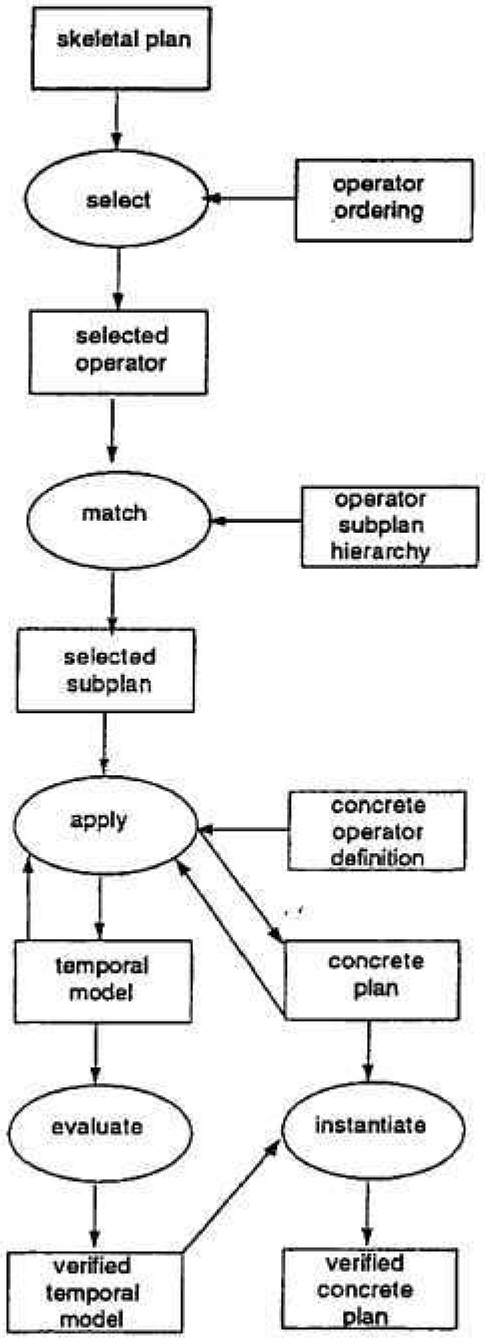

Figure 1: Inference structure for hierarchical skeletal plan refinement

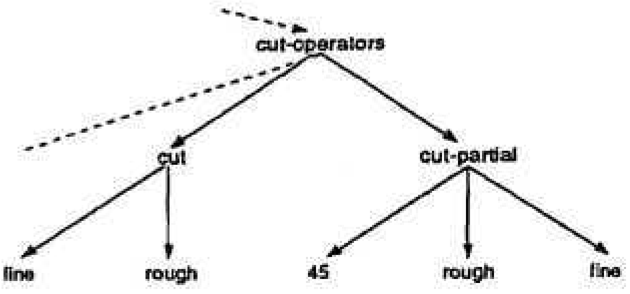

Figure 2: The hierarchy of abstract operators

3. The application of HSPR to production planning

HSPR was adopted by Kühn, Schmalhofer, Schmidt et al (1991) for the ARC-TEC project. This project adopted the KADS methodology and applied it to the domain of planning the manufacture of mechanical parts. A typical problem in this domain is machining the profile of an axle using a lathe. This may require a series of cuts to be made in order to produce a range of surface contours and finishes. The appropriate cutting tools and clamping tools must be selected and applied in the correct order. Production planning requires knowledge based reasoning to be combined with reasoning about action.

The following section presents the production planning problem in detail and specifies a logical formalisation of its key features. In this work we have taken the key features to be workpiece geometry and machine environment. KADS terminology is introduced and explained as it is used. Subsequent sections deal with the formalisation of the domain and inference knowledge and present a trace of the output of the running system.

3.1. The domain layer

In KADS the domain layer represents concepts, relations, attributive relations and structures of the domain in question. Examples of these are: facts, empirical associations and structural associations (Wielinga, Schreiber and Breuker (1992)).

The domain layer of the HSPR model contains 3 theories: a theory for ordering abstract operators, a hierarchy of abstract operators and their corresponding concrete sub-plans and a set of rules defining concrete operators.

The operator ordering theory specifies a theory of preference in the selection of abstract operators for refinement, for example, the relation
$(\forall$ `i) prefer(de-chuck,cut-full(i))` defines that the de-chucking operation should be selected before a cutting operation.

A skeletal plan must be specified in terms of operators named in the hierarchy of abstract operators. A section of this hierarchy is illustrated in Figure 2. Each leaf node of the hierarchy corresponds to a sequence of concrete operators (a concrete subplan). This characterisation allows skeletal plans to be defined at a range of levels of abstraction, from a leaf node which corresponds to a single concrete subplan, to an internal node which names a general class of operations. An abstract operator may also describe the execution several concrete operators. This data structure is referred to as the *abstract operator- subplan hierarchy.*

To allow plans to be re-used the definitions of the operators of a successful plan must be generalised (Schmalhofer et al (1991)). This is achieved through knowledge acquisition. For example, in a past solution a particular cut operation may remove 5mm of material from the workpiece. An expert might generalise this feature by allowing the depth of cut to lie in the range 1mm-6mm. As a result of acquiring this knowledge the definition of the operator would be modified by defining depth-of-cut as a variable, with a preferred value of 5mm and a requirement that this value must be within the range 1mm-6mm. The domain level representation language must be sufficiently expressive to be able to represent actions which have variable parameters.

The objects in the application domain that we must represent in the logical language are cylinders whose radius and surface finish may vary across their length (z axis). These three dimensional workpieces may be fully described by their two dimensional cross-section and this simplification allows us to define each surface of the cross-section by its cartesian coordinates. This scheme of domain representation is not an abstracted description, as a symbolic representation of surfaces and edges would be, however it is unlikely that this approach could be extended to the three dimensional case.

An example of a domain level rule in temporal logic is given in Figure 3. This rule defines the generalised *rough-cutting* operation. The symbols in capital letters represent predicates, the symbols `all`, `implies`, `and`, `i-future` are logical operators, the symbols `vector`, `cut` are functions and all other symbols are variables or numbers. The predicate `SURFACE(i,v)` states that workpiece surface `i` is defined by the set of parameters `v`, `v` is tuple of 5 values:

`v = vector(z,r,dz,dr,rgh)` where
- `z` = the z co-ordinate of the left-hand edge of the surface
- `r` = the radius of the left-hand edge of the surface
- `dz` = the change in z co-ordinate over the length of the surface
- `dr` = the change in radius over the length of the surface
- `rgh` = surface roughness

Surfaces are named by an integer. By convention the left-most surface is 1, the adjacent surface is 2 and so on. If the rule defined in Figure 3 is applied with j=2 then the rule states that surface 1 has its `dr` value reduced by `dep`, surfaces 2 and 3 have their `r` values reduced by `dep` and surface 3 has its `dr` value increased by `dep` (the `dr` value of surface 3 is negative). The variable `dep` is the depth of cut. Figure 4 shows the effect of this rule in a graphical form. The conditions of this rule ensure that it can only be applied to horizontal surfaces that have adjacent vertical surfaces, as illustrated.

Temporal rules which define the effects of cutting a horizontal surface which has non-vertical adjacent surfaces have also been defined. Another class of rules we have defined are those which specify a cut partially across a surface. In this case the number of surfaces increases by two and, as previously, we must define the changes in lengths of all the modified surfaces. The persistence of the `ENVIRONMENT` and `SURFACE` predicates must be explicitly defined. If there are five surfaces and surface 4 is cut, then by the rule of Figure 3 we can deduce the effects on surfaces 3, 4 and 5. However we must also define a rule which states the surfaces 1 and 2 are unchanged. In the case of a partial-length cutting operation the names of surfaces which are not affected by the cutting operation may change (due to the production of new surfaces) and this must be explicitly defined.

Our domain level formalisation of planning can be summarised as follows: The con-

```
all(i,all(j,all(k,all(zi,all(ri,all(dri,all(rghi,all(new-dri,
all(zj,all(rj,all(dzj,all(rghj,all(zk,all(drk,all(rghk,
all(new-rj,all(new-drk,all(dep,all(cm,all(ct,
           implies(and(ENVIRONMENT(high-force(cm),rough(ct)),
                   and(SURFACE(i,vector(zi,ri,0,dri,rghi)),
                   and(PLUS(i,1,j),
                   and(SURFACE(j,vector(zj,rj,dzj,0,rghj)),
                   and(PLUS(j,1,k),
                   and(SURFACE(k,vector(zk,rj,0,drk,rghk)),
                   and(MINUS(rj,dep,new-rj),
                   and(PLUS(drk,dep,new-drk),
                       MINUS(dri,dep,new-dri)))))))))),
           i-future(cut(rough,surface(j),dep),
                   and(SURFACE(i,vector(zi,ri,0,new-dri,rghi)),
                   and(SURFACE(j,vector(zj,new-rj,dzj,0,rough)),
                   and(SURFACE(k,vector(zk,new-rj,0,new-drk,rghk)),
                       ENVIRONMENT(high-force(cm),rough(ct))))))
                                                  ))))))))))))))))))))))
```

Figure 3: A concrete operator definition rule in temporal logic.

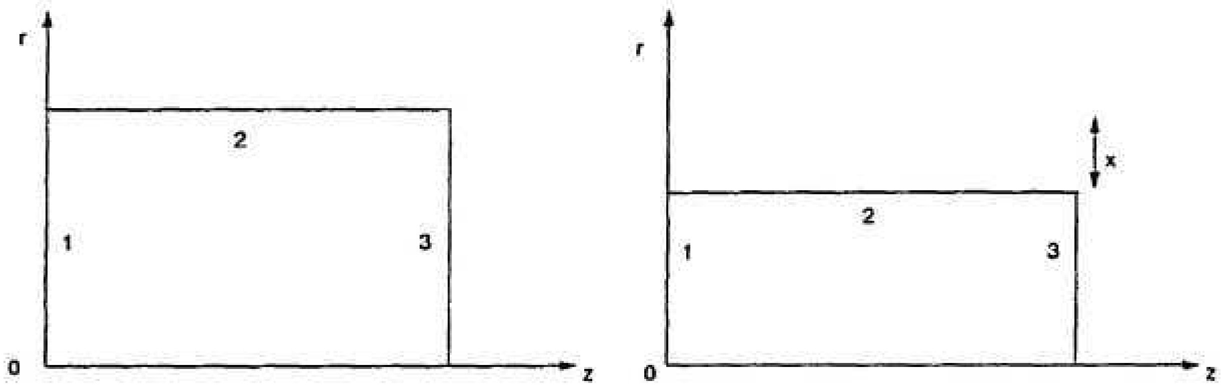

Figure 4: Rough cutting applied to surface 2

```
forall(x,forall(y,forall(z,forall(d,
              Knowledge-source(match,
                                (selected-operator(current(x,d)).
                                 &.theory(operator-hierarchy,y).
                                 &.member(instance(x,z),y).
                                 =>.selected-subplan(z,d)))))))))
```

Figure 5: match knowledge source definition

junction of SURFACE predicates defines the two dimensional cross-section of the workpiece. By applying temporal rules we can derive a temporal model which specifies the cross-section of the workpiece at each point in time. A successful plan is a sequence of operators whose temporal model begins with the initial workpiece cross-section and ends with the goal cross-section[2]. The ENVIRONMENT predicate defines the configuration of the machine on which the workpiece is mounted. This configuration may change during the plan and the execution of cutting operations is dependent upon this configuration.

3.2. The inference and task layers

Knowledge sources specify the inferences that can be made and the procedures that can be applied in problem solving. It is an important feature of the KADS methodology that knowledge sources should be specified independently of the control strategy that will ultimately be used. A knowledge source defines how the contents of one meta-class are modified by a knowledge theory and the contents of one or more meta-classes. The configuration of knowledge sources defines the inference structure. The configuration of knowledge sources for HSPR is illustrated in Figure 1 and their purpose has been described in Section 2. This section describes the formalisation of the inference structure for the match knowledge source. The other knowledge sources are defined in a similar fashion.

The formalisation of the match knowledge source is given in Figure 5. The predicate Knowledge-source takes the knowledge source name and a list as arguments. selected-operator and selected-subplan are meta-classes, theory refers to a named knowledge theory and member is the standard list processing function. The structure defines a rule which states that if x is the current selected operator and y is the operator-hierarchy then z is the selected-subplan if instance(x,z) is a member of y. The symbols & and => simply delimit the arguments and conclusions of the rule in an intuitive manner. The interpretation of the rule (the list structure) is defined at the task level.

The purpose of the task layer is to apply the reasoning steps defined by the knowledge sources. These reasoning steps must be carried out according to the control strategy of

[2]It should be noted that all effects of each operator (at the concrete level) are defined, thus all facts about the future state are deduced from the current state, through the rules which define the concrete operator. No assumption of persistence is made, and the non-linear assumption is made. Therefore, if subgoals interact this will be reflected in the descriptions of the world. If action-1 causes tool-e to be fitted to the machine, action-2 requires tool-h, tool-h is not in the description of the world then the preconditons of action-2 are not met, and it cannot occur. It is assumed that the skeletal plan and the concrete sub-plans are totally ordered.

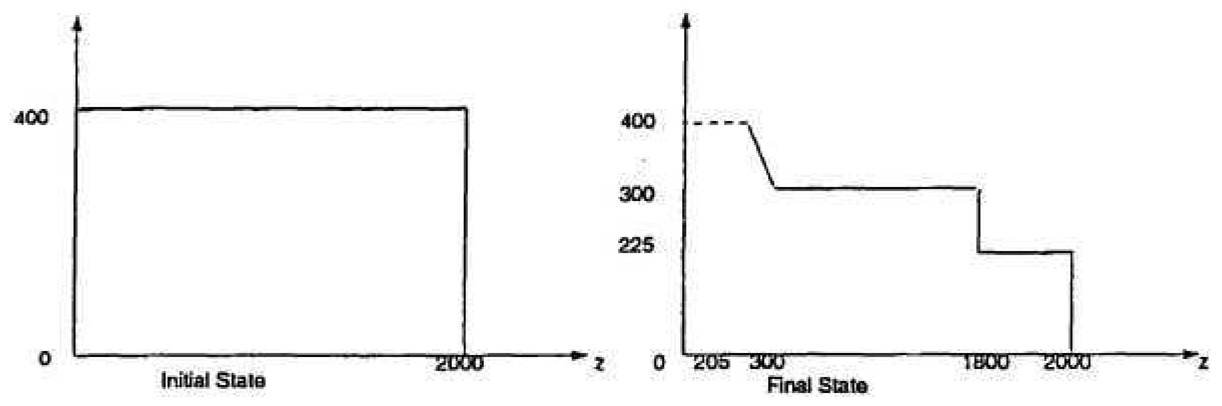

Figure 6: Initial and goal states of the workpiece

the problem solving process. For example the match knowledge source defined above may be applied such that all instances of an operator are found or such that one instance is selected, and future applications of this knowledge source select a new instance. The task layer must be capable of representing and reasoning about the knowledge source rule. The task layer formalisation we adopt explicitly defines the instantiation of a meta-class at a particular state in problem solving. A full account of the formalisation problem is given in Aitken et al (1993). These ideas on the structuring of domain, inference and task knowledge were implemented in Prolog to produce an operational planner. The following section describes the application of this system to a problem in the production planning domain.

3.3. An example solution

This section presents a simple example of the refinement of a skeletal plan. The results presented are program output, our sytsem is built using PCE-Prolog and runs on SUN Sparc machines. The initial state and goal state of the workpiece are shown in Figure 6 in diagrammatic form. The skeletal plan is:

```
skeletal-plan( setup-wp.
               cut-partial-45(surface(2)).
               cut-partial(surface(4)).
               re-tool(fine-cutting)
               cut-fine(surface(2)).
               de-chuck)
```

This plan specifies a sequence of six operations: setting up the workpiece (clamping and selecting a cutting tool), cutting surface 2 partially, cutting the newly produced surface 4 partially, re-tooling the machine, cutting surface 2 to produce a fine surface finish and finally de-clamping the workpiece. Each of these operators must be replaced by a concrete subplan as the temporal model and the concrete plan are constructed. Part of the sequence of knowledge level states is as follows:

Solution State	Meta-class instantiation
State I	`selected-operator(current(setup-wp,forwards))`
State J	`selected-subplan(collet-chuck(m).retool(m,tool-r1),forwards)`
	`concrete-plan(lhs(collet-chuck(m).retool(m,tool-r1)),`
	`rhs(de-chuck(m)))`
State K	`selected-operator(current(cut-fine(surface(2)),reverse))`
State L	$(\forall x)$ `selected-subplan(cut(fine,surface(2),x),reverse)`
	$(\forall x)$ `concrete-plan(lhs(collet-chuck(m).retool(m,tool-r1)),`
	`rhs(cut(fine,surface(2),x).de-chuck(m)))`

At I `setup-wp` is selected for refinement, at J the corresponding subplan is added to the concrete plan, at K `cut-fine(surface(2))` is selected for refinement and at L the corresponding subplan is added to the concrete plan. In this trace the selected subplans have successfully been applied to the temporal model, if this had not been the case then an alternative subplan would have been sought. This trace also shows that the temporal model and concrete plan are constructed from the initial state (forwards) and also from the goal state (reverse)[3].

The sequence of actions which the planner deduces as the solution are the following:

```
 verified-concrete-plan( collet-chuck(m).
                         retool(m,tool-r1).
                         cut(rough-rhs-45,surface(2),1700,100)
                         cut(rough-rhs,surface(4),200,75)
                         retool(m,tool-f1)
                         cut(fine,surface(2),5)
                         de-chuck(m))
```

In the concrete plan the first two operators were derived from one abstract operator, specific tools are named and the length and depth of the required cutting operations (in millimeters) are specified. The sequence of states that this plan produces are shown, superimposed, in Figure 7.

[3]Variables are bound when they are matched with constants in the temporal model, or when they are matched with an evaluable expression.

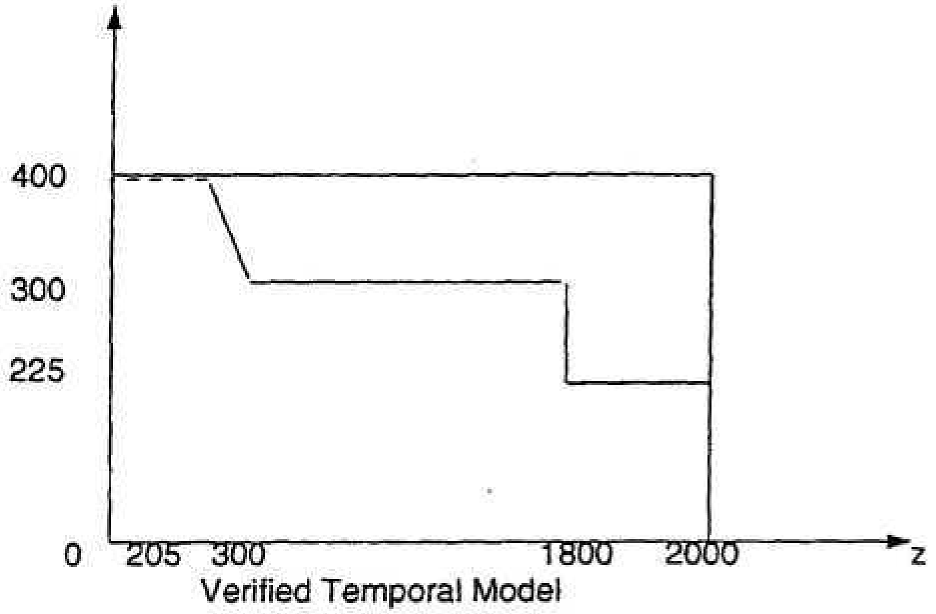

Figure 7: The sequence of states of the verified temporal model

4. Conclusions

This paper has presented an inference structure for hierarchical skeletal plan refinement and illustrated its usefullness by an example from the domain of production planning. We do not claim to have invented this approach to planning, rather our contribution is to formalise the method so that it can be used in a structured approach to KBS design.

This work has demonstrated that a conceptual model of problem solving can be used to describe the solution of a domain problem expressed in a formal calculus. The model of expertise describes the problem solving method and has a structure and specification which is distinct from the domain theory. The domain theory plays a clearly defined role in the model of expertise.

The domain theory described in this paper is sufficiently expressive to enable planning problems to be solved by deduction alone. However without the use of skeletal plans to organise the search space the planning process would be computationally intractible.

A key feature of the approach is the potential for knowledge reuse. It should be clear that the model is applicable in domains where skeletal plans can be identified (matching problem classes). The model identifies a number of domain theories, that must be acquired or be inferrable (the operator hierarchy, ordering). Concrete operators, specifying precisely the changes in the world, must also be defined. The model defines a method for solving planning problems where this information exists, or can be acquired.

We believe the combination of a conceptual model of the problem solving process with a formal domain theory to be a promising solution to the problems of reasoning in formally specified domains such as logical or mathematical theories. Meta reasoning has been applied both to planning (Corlett et al (1989)) and to mathematical reasoning (Bundy et al (1988)) and been found to be an advance on more direct approaches though not without problems (van Harmelen (1989)). The definition of a semi-formal model of the solution method, as specified in a KADS inference structure, which is clearly distinguished from the domain knowledge, and domain model, can be viewed as a kind of meta-object distinction. The KADS methodology entails that both types of knowledge are modelled separately while acknowledging the relationships between the two levels. This methodology offers the potential for the reuse of the solution method, this has not typically been the case in existing meta-level approaches.

Acknowledgements

We would like to thank the reviewers for comments on an earlier version of this paper. The research reported in this paper was carried out whilst Stuart Aitken was employed as a Research Fellow at the University of Nottingham. The work was esupported by the DFKI and by SERC grant no GR/F 35968.

References

Aitken, S. Kühn, O. Shadbolt, N. Schmalhofer, F. (1993) A conceptual model of hierarchical skeletal planning and its formalisation. *Proc. 3rd KADS User Meeting, Siemens AG Munich, March 8-9 1993 :229-247*

Bundy, A. van Harmelen, F. Hesketh, J. and Smaill, A. (1988) Experiments with proof plans for induction. *Research Report, AI Department, University of Edinburgh*

Chapman, D. (1987) Planning for conjunctive goals. *Artificial Intelligence 32 :333-377*

Corlett, R. Davies, N. Kahn, R. Reichgelt, H. van Harmelen, F. (1989) The architecture of Socrates. *in Logic based knowledge representation. Jackson, P. Reichgelt, H. and van Harmelen, F. (eds) MIT Press :37-64*

Friedland, P.E. and Iwasaki, Y. (1985) The concept and implementation of skeletal plans. *Journal of Automated Reasoning 1 (1985) :161-208*

Kühn, O. Linster, M. Schmidt, G. (1991) Clamping, COKAM, KADS, and OMOS: The construction and operationalisation of a KADS conceptual model. *DFKI Technical Memo TM-91-03*

Kühn, O. and Schmalhofer, F. (1992) Hierarchical skeletal plan refinement task and inference structures. *Proc. 2nd KADS user meeting, Siemens AG Munich, February 17-18, 1992*

Lifschitz (1986) On the semantics of STRIPS. *Proc. 1986 workshop, Reasoning about actions and plans*

Newell, A. Simon, H.A. (1963) GPS, a program that simulates human thought. *in Computers and thought. Feigenbaum, E.A. Feldman, J. (eds) :279-293*

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 Springer Verlag*

van Harmelen, F. (1989) A classification of meta-level architectures. in *Logic-based knowledge representation. Jackson, P. Reichgelt, H. and van Harmelen, F. (eds) MIT Press :13-35*

Wielinga, B.J. Schreiber, A.T. Breuker, J.A. (1992) KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition 4: 5-53*