Problem Solving in Interactive Proof: A Knowledge-Modelling Approach

James Stuart Aitken[1]

Abstract. This paper presents a model of proof discovery derived from the proof attempts of subjects who carried out interactive proofs using the HOL or Isabelle provers. Techniques of knowledge modelling, from knowledge-based system development, are used to derive a semi-formal model of the knowledge utilised by the subjects. The proposed model makes claims about the relation between the problem class, the proof plan and its implementation.

1 INTRODUCTION

Automated theorem provers are able to solve increasingly more complex problems, however, these tasks still only approximate real world verification tasks. It is becoming clear that provers developed according to the automated-proof philosophy will very often require user intervention to complete a proof. Consequently, the user must attempt to comprehend the state of a partially successful proof and the representation of its derivation. The user must then decide how the proof should be developed.

In *interactive theorem proving*, e.g. in HOL [6], the user chooses all proof steps him/herself. The user is thus responsible for all high level and low level decisions regarding the general strategy of the proof and the tactics to implement the strategy. The user is less likely to have problems orienting themselves in a proof (in [1] the frequency of user errors in proof context are quantified and found to be negligible). However, users are often uncertain about the correct proof strategy and use the theorem prover to explore conjectures about the proof.

One aim of several recent approaches to automated theorem proving is to find ways to implement humanlike proof strategies [8]. While these techniques cannot generally be equated with cognitive models, they are nonetheless the best models of problem solving in the domain of theorem proving that we have. For this reason we shall contrast our findings, which concern human problem solving, with the techniques of automated reasoning. We also relate our results to cognitive studies of programming, whose methods and findings we believe to be relevant to the study of interactive proof.

In this paper we identify levels of representation and structure in proofs found interactively. This structure is not utilised in current automated provers, hence we propose, firstly, that new methods of automated proof should utilise the types of knowledge we identify, and secondly that this structure should be extracted from automated proofs in order to generate an explanation – irrespective of how the proof was actually found.

This paper continues in Section 2 with a description of the methodology used to collect and interpret the data. An aggregate picture of the activities of interactive proof is presented in Section 3 and a knowledge-based model of problem solving is presented in Section 4. These two descriptions of the observed proof activity differ in that the former is a relatively uninterpreted view of the data, while the latter postulates the existence of various concepts and their role in problem solving. Finally, we discuss the model and its relation to automated approaches to proof and to programming.

2 METHODOLOGY

In this paper we take an observational approach to knowledge acquisition. This approach is comparable with that of [2] who note that observational studies have the advantage that they can be used to discover what the expert actually does. The disadvantage of this approach is the complexity of the analysis and the time required – in contrast with the standard approaches. We believe the observational approach is appropriate in this instance as there are insufficient existing studies which we could make use of to obtain background information, prior to, for example, using a model-based knowledge acquisition method. In the well-known study of [9] the participants had no accumulated experience in theorem proving. Their problem-solving skills in this domain were consequently undeveloped and the study is therefore not an appropriate source of data for our purposes.

The following experimental method was used. Subjects were asked to prove a given theorem. They were requested to think aloud while attempting the proof and their voice was recorded on tape. All inputs to and outputs from the prover were recorded (the shell script) as was the final proof (the proof script). After the proof was completed, or 40 minutes had elapsed, subjects were asked to complete a short questionnaire. This experimental method was intended to gather a range of data relating to the human-prover interaction in addition to collecting information required for knowledge acquisition. See [1] for more details.

All subjects were asked to prove the following theorem:

$$\begin{aligned} \vdash \forall P.(P\ [] \wedge \\ \quad (\forall x.P[x] \wedge \forall l_1\, l_2.\, (P\ l_1 \wedge P\ l_2) \supset P(\mathsf{APPEND}\ l_1\ l_2)) \\ \quad \supset\ \forall l.P\ l \end{aligned} \tag{1}$$

Transcripts were made from four think-aloud recordings. Utterances were grouped into paragraphs, or foci. The focus of an utterance was determined by the meaning of the words of the utterance and the context in which the words were spoken. This context was defined by previous focus and the point in the proof as determined from the shell script.

Each focus was described by a summary phrase. In contrast with [2], where dialogues are analysed, we cannot analyse the *function* of utterances (or foci) as we dealing with monologues. We restrict

[1] Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, Scotland. email: stuart@dcs.gla.ac.uk

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

ourselves to summarising the evidence of the transcript and the shell script into a more concise form and putting it into temporal order; focusing on the activities that are described (verbally) or were recorded. The aim of this analysis is to identify patterns of activities in interactive proof with the minimum amount of interpretation. Section 3 describes the activities of interactive proof.

The initial analysis yields a picture of proof activities, but this picture has no explanatory value. For this we need to postulate a model, such as a task model or a knowledge-based model. Following the model-based approach to describing expertise [13] we propose an *inference structure* which specifies a number of concepts and steps of reasoning which model the problem solving behaviour of the subjects. This model is presented in Section 4.

3 THE ACTIVITIES OF INTERACTIVE PROOF

In order to describe the transcripts in a consistent way, the following terminology was defined:

tactical level A description of proof step(s) in terms of tactics.

logical level A description of the proof, or of proof steps, in general, logical terms, i.e. a description which is not concerned with the tactics which realise it.

consider problem A verbalisation of the top goal statement or a reference to it.

sketch solution A general description of the solution in logical terms.

consider current goal A verbalisation of the current goal or a reference to it.

formulate proof step A description of the next proof step(s) by naming the tactic(s) (a tactical level description) or by describing the effects of the tactic (a logical level description), i.e. a specific description of *how* a tactic operates.

repeat proof step An utterance indicating that a previously formulated proof step will be repeated.

consider strategy A description of the next several proof steps in abstract, logical terms.

revise strategy A description of the next several proof steps which revises an earlier strategy.

assess A judgement of the result of a proof step with respect to the current strategy. An assessment may also be indicated indirectly, for example by commenting on the lack of response of the prover.

review proof script A modification of the proof script other than at the current subgoal.

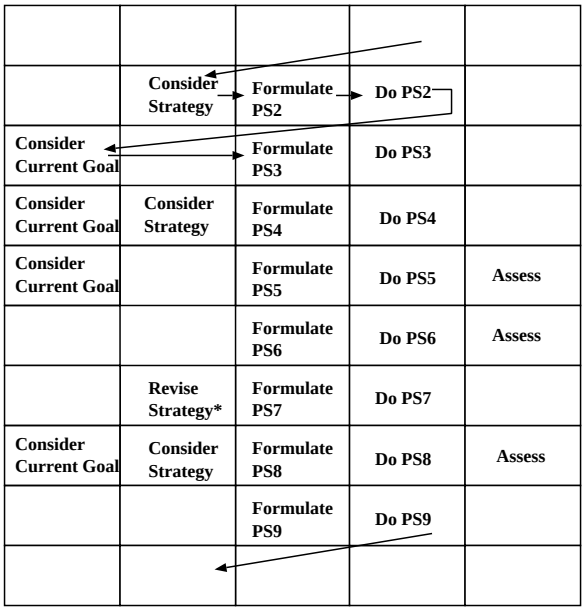
recognise context error A statement indicating an error such as backing up too far.

acquire information A statement indicating that additional information is being sought.

do proof step Execution of a proof step indicated by a statement or by the HOL shell script.

undo proof step Backing up one or more proof steps, indicated by a statement or by the HOL shell script.

Each group of utterances was summarised by one of the above phrases, with the exception of those which were irrelevant to the proof activity, e.g. comments about typing mistakes, problems with the emacs editor etc. The terminology was not defined prior to the analysis of the transcripts. Pilot studies had shown some common activities, but had not produced a sufficiently comprehensive set of terms. The sequence of activities undertaken by each subject can be examined by writing the activities in a tabular form. All subjects began by considering the problem and sketching the solution (mentally) to some degree. The activities of subject C are illustrated in Figure 1



Additional Activities: *Acquire Information

Figure 1. The activities of subject C.

which shows the formulation and execution of proof steps 2 to 9 (PS2-PS9). This diagram shows a typical pattern of activities during the proof: proof steps are formulated then executed, the result is often assessed and formulation is often preceeded by considering the current goal or by considering strategy. Figure 1 does not show every formulation of a proof step being preceeded by a perceptual activity (e.g. considering the current goal) or being followed by an evaluation activity (e.g. assessment). This is in part due to subjects not verbalising all their thoughts. It is also likely that on some occasions subjects did not take time to read the output of the prover. There were differences between subjects in the manner in which they thought aloud, and in the length of silences during rapid activity – particularly towards the end of the proof. Consequently, it is unlikely that an entirely uniform pattern of activities could emerge.

To discover what an aggregate picture of proof activity looks like we counted the transitions between activities in all transcripts. For each of the twelve activities there are eleven other activities which might follow, therefore there are a total of 132 possible transitions. In fact only 29 distinct transitions occurred. When those transitions which are unique to one subject are eliminated, 16 distinct transitions remain. These account for 111 of the original 126 transitions found in the transcripts (88% of the data) and Figure 2 illustrates the frequency of these transitions.

In Figure 2 the activities are distinguished according to whether they concern purely logical concepts (the logical level) or whether they concern concepts specific to interaction with the HOL prover (the interaction level). The *consider current goal* activity has an ambiguous status. It may describe a purely perceptual action; but the current goal is also the problem solving goal and hence the subject may be considering the goal at the logical level. To resolve this type of ambiguity it is necessary to model the problem-solving behaviour at greater depth. We now turn to this task.

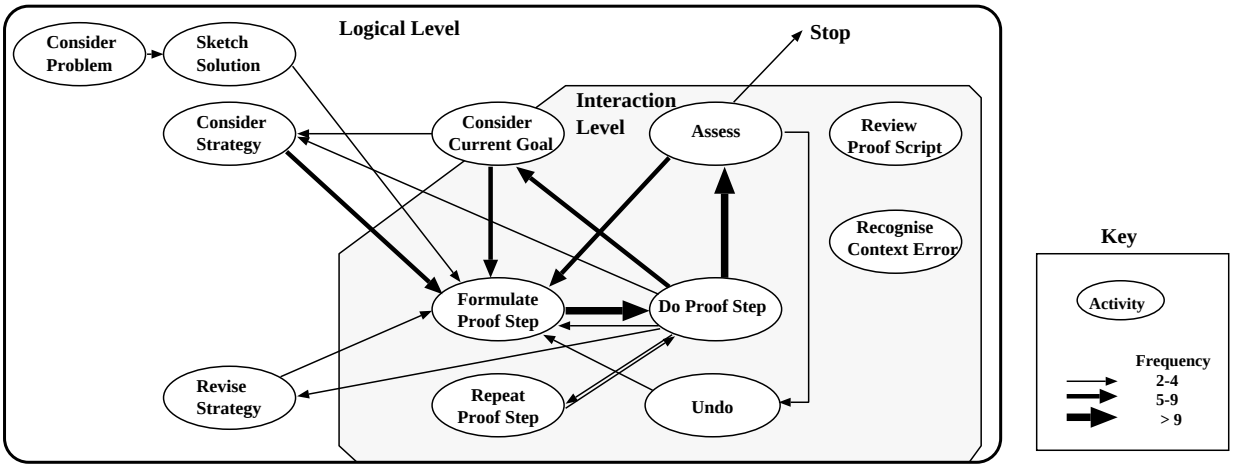


Figure 2. Activity diagram.

4 A KNOWLEDGE-BASED MODEL

We begin with an informal description of typical problem-solving behaviour, and then describe the knowledge acquisition method in detail and present the resulting knowledge-based model. We shall make use of the data from the HOL users experiment together with data from four subjects who used the Isabelle [10] theorem prover. These subjects were asked to formulate a definition and to prove the following theorem, where $\mathsf{funpow}\ f\ n\ x\ = f^n(x)$:

$$\vdash \forall x\ f\ n\ m.\ \mathsf{funpow}\ f\ (n+m)\ x = (\mathsf{funpow}\ f\ n)(\mathsf{funpow}\ f\ m\ x) \qquad (2)$$

Problem solving was always observed to begin by sketching the solution, or strategy (as defined above), of the proof. Examples include *list-induction* and *simplify-with-arith*. Two of the Isabelle users sketched their solutions on paper before beginning the interactive proof. The tangible result of selecting a strategy was a proof step (a line of input) or a sequence of proof steps. It was determined, by means of a questionnaire, that the selection of tactics was often uncertain or experimental. This supports the proposal that the relation between the theorem to be proven and the solution strategy is a heuristic one. Subjects who chose a wrong solution approach, e.g. *double-list-induction* (HOL subject D) or induction on the wrong variable (Isabelle subject 1), revised their approach upon the failure of a tactic or when progress in the proof was difficult.

In HOL it is possible to combine several tactics into one proof step; for example, one HOL subject (A) combined four tactics into one proof step, while other subjects entered these tactics in two or in three proof steps (subjects D and B respectively). In order to model the derivation of proof steps from a strategy we introduce the concept of a *plan* which is composed of *actions*. The plan is further refined to derive a sequence of tactics. In the case of subject A, the plan is a single compound action [*prepare-list-induct-assum-rewrite(l)*] which is refined into the tactic sequence:

```
GEN_TAC THEN STRIP_TAC THEN LIST_INDUCT_TAC
THEN ASM_REWRITE_TAC[];;.
```

In the case of subject B, the plan is composed of three actions: [*prepare*, *list-induct(l)*, *assum-rewrite*] and these actions are refined to the following three tactic sequences:

```
GEN_TAC THEN STRIP_TAC;;
INDUCT_THEN list_INDUCT MP_TAC;;
ASM_REWRITE_TAC [];;
```

The important distinction between these two plans (which result in essentially the same tactics) is that the second plan permits the intermediate proof states to be examined while the first does not. Subject A was more experienced than subject B and this may explain the more compact plan.

Both plans are partial plans – they solve only the first subgoals of the main goal. Subjects were not able to produce complete plans of the proof without evaluating some tactics. This is simply because it is not possible to do interactive proof solely in the head. This is not to say that subjects did not consider the solution of the step case in their initial sketch of the solution; no subject attempted to predict the exact form of the step case subgoal without interaction with the prover (or pencil and paper calculation). The plans describe stereotypical sequences of actions. The existence of such patterns has previously been noted [3].

4.1 Knowledge acquisition and knowledge modelling

Knowledge acquisition is a modelling activity where the interpretation of the data is guided by known, or hypothesised, models of problem solving. In this case the sources of data were the transcripts of the think-aloud recordings and the shell script. Concepts were identified from phrases in the transcripts and rendered into a decontextualised form where necessary. A problem-solving model was defined and the explanatory power of the model was assessed.

One candidate problem-solving model is heuristic classification [4] where solutions are heuristically associated with problems. In this model *strategies* might be identified with certain internal nodes of the solution hierarchy (a hierarchy of plans). However, this distinction between strategy and plan is rather unsatisfactory.

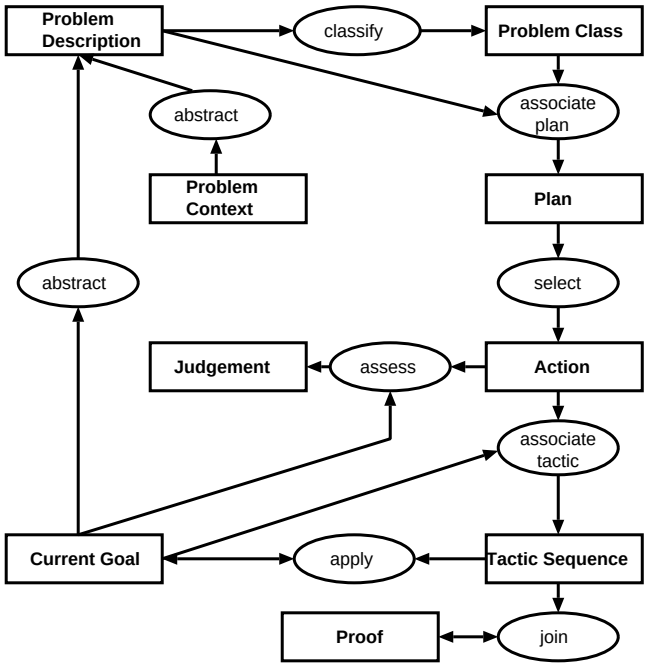


Figure 3. The *classify and associate* inference structure.

The observed pattern of reasoning can also be described by introducing the notion of a *Problem Class*. Specific problems are first classified as an instance of a particular problem class and then the solution is obtained by association [5, 7]. The heuristic aspect of problem solving is retained as classification may be an uncertain inference. The main organising principle of the problem class hierarchy is form of the solution. For example, *assumption-rewriting-problem* is now viewed as a problem class, a subclass of *rewriting-problem*. Previously, we used the term strategy to describe typical solutions of problems in these classes. Now we distinguish problem types from solutions (plans). The full model is shown in Figure 3 and we now describe each reasoning step, known as an inference step and class of concepts, known as a meta-class, at this 'meta-level'.[2]

The *Current Goal* is the goal to be proven as it is represented at the prover interface. At the start of problem solving the instance of *Current Goal* is:

```
!P.P[] /\
   (!x. P[x] /\
   (!l1 l2. P l1 /\ P l2 ==> P(APPEND l1 l2)))
    ==> (!l. P l)                                  (3)
```

Both the *Current Goal* and the *Problem Context* are abstracted to obtain the *Problem Description*. When the *Current Goal* is (3) the *Problem Description* contains equation (1), among other information. We shall develop this example.

The *classify* inference determines the *Problem Class* which is associated with the *Problem Description*. Examples of problem classes which are associated with (1) are `list-induction-problem` and `double-list-induction-problem`.

[2] We have attempted to use KADS terminology where possible, hence refinement is now called *association* which is actually a composition of two primitive steps: *specialise* and *select*.

A *Plan* is derived from the *Problem Class* by the *associate plan* inference. The plan is a pre-existing sequence of actions which forms a partial solution. The associate inference may depend on the abstracted form of the current goal or on the problem context. This will be the case where there is a choice of an important parameter such as the induction variable and the induction action needs to contain the variable name. Similarly, a rewriting action may need to specify a theorem name – an element of the problem context. Such actions can be represented in the plan as `do-induction-on(x)`. In the example case, plans include:

[`prepare-list-induct-assum-rewrite(l)`] and
[`prepare`, `list-induct(l)`, `assum-rewrite`].

Each action in the plan is then executed. This requires an *Action* to be selected from the *Plan* by the *select* inference. The *Action* must be refined to, or associated with, a *Tactic Sequence* where a tactic is an executable command. Association may make reference to the *Current Goal* as commands are often sensitive to the syntax of the goal expression. For example, an action on an assumption (e.g. specialisation) may be implemented by a tactic which may operate on the *ith* assumption of a list of assumptions or on a specific subexpression. The `prepare` action may be associated with:

```
GEN_TAC THEN STRIP_TAC;;   or with
REPEAT STRIP_TAC;;.
```

The tactic sequence is then *applied* and the result is *assessed*. If the *Judgement* is positive then the *Tactic Sequence* is *joined* to the *Proof*.

Once all the actions in a plan have been refined and successfully executed the *Current Goal* is abstracted to obtain a new *Problem Description*. This new problem must be solved by classifying it and deriving a plan; the sequence of inferences described above is repeated.

The sequence of inferences, and their dependency on the outcome of other inferences, is represented at the task level [13]. We now describe the sequence of inferences on the failure of a tactic. If a tactic fails then an alternative *Tactic Sequence* associated with the current *Action* is sought – the *associate tactic* inference is repeated. If no successful tactic sequence is found then the previous action is selected and an alternative refinement of that action is sought. The *associate plan* and *classify* inferences may be repeated until all possible problem classes have been exhausted.

4.2 Assessing the validity of the model

The validity of the model as an explanation of the observed behaviour was assessed by attempting to fit all observed problem-solving episodes to the model. The primary source of data to be explained was the temporal sequence of tactics; related sequences of tactics were termed episodes. The sequence of concepts was considered to be an important, but secondary, source of data.

The first episode in all transcripts could be described by the *classify and associate* model. All 18 subsequent problem-solving episodes identified in the transcripts of the Isabelle users were also consistent with this model. In the HOL users transcripts, 7 of the remaining 13 episodes could be described by the *classify and associate* model. Six problem-solving episodes could only be described as *planning* as the sequence of tactics could only have arisen as a result of explicit reasoning about the effects of actions on the proof state.

The model describes problem solving in the HOL trial less well than in the Isabelle trial. The explanation for this lies in the goal theorems that were proven. Equation (2) can be proven by induction and rewriting; this structure is very typical. The proof of equation (1) may require more complex reasoning in the step case if the definition of APPEND used to relate CONS to APPEND. Equation (1) may be proven in a more standard way if a theorem with the correct properties is used. However, only one subject made use of this theorem.

The *classify and associate* process we have described is one of several types of problem-solving activity which were observed. Reasoning about actions is not accounted for in the proposed knowledge model. This type of reasoning is best described by a different knowledge model, rather than by extending the proposed model. Plan synthesis and plan modification both require such reasoning.

5 REPRESENTATION AND REASONING IN PROOF AND PROGRAMMING

The levels of representation we define differ from those most often employed in automated theorem proving. Tactics, as we use the term, are the lowest level of representation of logical inferences. This level has been fixed by the HOL and Isabelle user communities over many years. Actions are meaningful proof steps which might require several tactics to implement. Actions do not have a fixed granularity, but have a complexity which is dependent on expertise. Sequences of actions, or a single compound action, constitute partial plans which describe the proof at an abstract level. In common with the proof planning approach [3] we find that many induction proofs have a stereotypical structure when viewed at this level. Abstract representations of proofs are also required if proofs are to be reused by analogy [8].

Studies of programming, e.g. [11, 12], have identified the notion of a plan as an abstraction of a program. Plans are associated with a class of problems in a problem taxonomy. In problem solving, a plan is an intermediate abstraction which links a problem to its solution, that is, to the program [12]. Plan knowledge also plays a role in program comprehension [11], as does both control flow knowledge and goal hierarchy knowledge (often termed procedural and functional knowledge respectively). Our model of interactive proof resembles these views of programming in the important respects of levels of representation and knowledge structures. This reinforces our view that proof and programming are similar activities from the problem-solving perspective. Further, studies of program comprehension can be seen to be relevant to the problem of understanding proofs, whether complete or incomplete.

6 DISCUSSION

Existing approaches to automated theorem proving put great emphasis on techniques for analysing the syntactic structure of theorems and goals. The observational studies reported on here stress the role of the subjects' knowledge of typical proof structures and the interactive nature of the discovery process. This emphasis does not exclude syntactic analysis and planning from the problem-solving process.

The knowledge-based model has several novel features. The first is that action refinement bridges the gap between the planning level and the tactic level. Naturally, this proposal rests on the aforementioned nature of actions as meaningful logical steps or as useful procedural steps which manipulate the proof state, generally to permit some logical operation. Further, conceptual modelling suggests that identifying the problem class is an important intermediate step in selecting a partial plan.

In our model, partially successful proofs can be explained at multiple levels of description and this is potentially useful for explaining the initial (presumably successful) proof steps in abstract terms while the sequence of the proof which fails can be presented in more detail. The first *proof idea* in a proof is encapsulated in the problem class associated with the top goal. Subsequent problem solving steps can also be summarised by the problem class, or by the sequence of actions that were attempted but failed, or at the lowest level, the tactics that were used to implement an action. The fact that humans discover proofs in this way gives confidence that it provides a good explanation structure.

In the examples quoted in this paper the association between the goal and a problem class could simply be stated as this knowledge was acquired from experimental trials. For the knowledge model to be the basis for an automated prover, techniques will be required for deriving higher level concepts from the syntactic form of goals and theorems. It is clearly necessary that some of this analysis be automated, while some associations must be learned from examples or obtained by conventional knowledge acquisition methods.

ACKNOWLEDGEMENTS

The author acknowledges the cooperation of researchers at the universities of Glasgow, Cambridge and Munich who participated in the experiments. This work was supported by EPSRC grant no. GR/K25038.

REFERENCES

[1] Aitken, J.S., Gray, P., Melham, T., and Thomas, M. Interactive Theorem Proving: An Empirical Study of User Activity. To appear in the *Journal of Symbolic Logic*.

[2] Belkin, N. J. and Brooks, H. M. Knowledge elicitation using discourse analysis. *International Journal of Man-Machine Studies* 27, pp. 127–144, 1987.

[3] Bundy, A., van Harmelen, F., Hesketh, J., and Smaill, A. Experiments with proof plans for induction. *Journal of Automated Reasoning* 7, pp. 303–324, 1991.

[4] Clancey, W. Heuristic Classification. *Artificial Intelligence* 27, pp. 289–350, 1985.

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

[6] Gordon, M. J. C. and Melham, T. F., eds. *Introduction to HOL: A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[7] Kühn, O. and Schmalhofer, F. Hierarchical skeletal plan refinement task and inference structures. *Proc. 2nd KADS User Meeting*, Siemens AG, München, 1992.

[8] Melis, E. How mathematicians prove theorems. *Proceedings of the 16th Annual Conference of the Cognitive Science Society*. eds. Ram, A. and Eiselt, K., Lawrence Erlbaum, pp. 624–628, 1994.

[9] Newell, A. and Simon, H. *Human Problem Solving*. Prentice Hall, 1972.

[10] Paulson, L.C. *Isabelle A generic theorem prover.* Lecture Notes in Computing Science 828, Springer Verlag, 1994.

[11] Pennington, N. Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology* 19, pp. 295–341, 1987.

[12] Soloway, E. From problems to programs via plans: The content and structure of knowledge for introductory LISP programming. *J. Educational Computing Research* Vol. 1(2), pp. 157–172, 1985.

[13] Wielinga, B.J., Schreiber, T., and Breuker. J.A. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition* 4, pp. 5–53, 1992 .