Using a Cognitive Architecture to Plan Dialogs for the Adaptive Explanation of Proofs

Armin Fiedler
FB Informatik, Universitat des Saarlandes
Postfach 15 11 50, D-66041 Saarbriicken, Germany
afiedlerQcs.uni-sb.de

Abstract

In order to generate high quality explanations in technical or mathematical domains, the presentation must be adapted to the knowledge of the intended audience. Current proof presentation systems only communicate proofs on a fixed degree of abstraction independently of the addressee's knowledge.

In this paper we propose an architecture for an interactive proof explanation system, called *Prex.* Based on the theory of human cognition ACT-R., its dialog planner exploits a cognitive model, in which both the user's knowledge and his cognitive processes are modeled. By this means, his cognitive states are traced during the explanation. The explicit representation of the user's cognitive states in ACT-R allows the dialog planner to choose a degree of abstraction tailored to the user for each proof step to be explained. Moreover, the system can revise its assumptions about the user's knowledge and react to his interact ions.

1 Introduction

A person who explains to another person a technical device or a logical line of reasoning adapts his explanations to the addressee's knowledge. A computer program designed to take over the explaining part, should also adopt this principle.

Assorted systems take into account the intended audience's knowledge in the generation of explanations (see e.g. [Cawsey, 1990; Paris, 1991; Wahlster *et a*/., 1993]). Most of them adapt to the addressee by choosing between different discourse strategies. Since proofs are inherently rich in inferences, the explanation of proofs must also consider which inferences the audience can make [Horacek, 1997; Zukerman and McConachy, 1993]. However, because of the constraints of the human memory, inferences are not chainable without costs. Explicit representation of the addressee's cognitive states proves to be useful in choosing the information to convey [Walker and Rainbow, 1994].

While a mathematician communicates a proof on a level of abstraction that is tailored to the audience, state-of-the-art proof presentation systems such as *PROVERB* [Huang and Fiedler, 1997] verbalize proofs in a nearly textbook-like style on a fixed degree of abstraction given by the initial representation of the proof. Nevertheless, *PROVERB* is not restricted to presentation on a certain level of abstraction. Adaptation to the reader's knowledge may still take place by providing the appropriate level of abstraction in the initial representation of the proof.

Drawing on results from cognitive science, we are currently developing an *interactive proof explanation system,* called *Prex* (for *proof explainer).* In this paper, which extends the work reported in [Fiedler, 1998], we propose an architecture for its dialog planner based on the theory of human cognition ACT-R [Anderson and Lebiere, 1998]. The latter explicitly represents the addressee's knowledge in a declarative memory and his cognitive skills in procedural production rules. This cognitive model enables the dialog planner to trace the addressee's cognitive states during the explanation. Hence, for each proof step, it can choose as an appropriate explanation its most abstract justification that is known by the addressee. Moreover, the system can revise its assumptions about the users knowledge and react to his interactions.

The architecture of P.rex, which is sketched in Section 3, is designed to allow for multimodal generation. The dialog planner is described in detail in Section 4. Since it is necessary to know some of the concepts in ACT-R to understand the macroplanning process, the cognitive architecture is first introduced in the next section.

2 ACT-R: A Cognitive Architecture

In cognitive science several approaches are used to describe the functionality of the cognitive apparatus, e.g. production systems, mental models or distributed neural representations. Production systems that model human cognition are called *cognitive architectures.* In this section we describe the cognitive architecture ACT-R [Anderson and Lebiere, 1998], which is well suited for user adaptive explanation generation because of its conflict resolution mechanism. Further examples for cognitive architectures are SOAR [Newell, 1990] and EPIC [Meyer and Kieras, 1997].

ACT-R. has two types of knowledge bases, or *memories,* to store permanent knowledge in: declarative and procedural representations of knowledge are explicitly separated into the declarative memory and the procedural production rule base, but are intimately connected.

Procedural knowledge is represented in production rules (or simply: *productions)* whose conditions and actions are defined in terms of declarative structures. A production can only apply if its conditions are satisfied by the knowledge currently available in the declarative memory. An item in the declarative memory is annotated with an activation that influences its retrieval. The application of a production modifies the declarative memory, or it results in an observable event. The set of applicable productions is called the *conflict set.* A *conflict resolution* heuristic derived from a rational analysis of human cognition determines which production in the conflict set will eventually be applied.

In order to allow for a goal-oriented behavior of the system, ACT-R manages goals in a goal stack. The current, goal is that on the top of the stack. Only productions that, match the current, goal are applicable.

2.1 Declarative Knowledge

Declarative knowledge is represented in terms of *chunks* in the declarative memory. Below is an example for a chunk encoding the fact that F C G, where subset-fact is a concept and F and G are contextual chunks associated to factFsubsetG.

```
fact FsubaetG
    isa    subset-fact
    setl   F
    set2   G
```

Chunks are annotated with continuous activations that influence their retrieval. The activation .A_i, of a chunk C_i, consists of its base-level activation B, and the weighted activations of contextual chunks. In B_i-, which is defined such that it decreases logarithmically when C_i is not used, ACT-R models the forgetting of declarative knowledge. Note that the definition of the activation establishes a spreading activation to adjacent chunks, but. not further; multi-link-spread is not supported.

The constraint on the capacity of the human working memory is approached by defining a retrieval threshold r, where only those chunks C_i, can be matched whose activation A_t is higher than T. *C*hunks with an activation less than r are considered as forgotten.

New declarative knowledge is acquired when a new chunk is stored in the declarative memory, as is always the case when a goal is popped from the goal stack. The application of a production may also cause a new chunk to be stored if required by the production's action part.

2.2 Procedural Knowledge

The operational knowledge of ACT-R is formalized in terms *of productions.* Productions generally consist of a condition part and an action part, and can be applied if the condition part is fulfilled. In ACT-R both parts are defined in terms of chunk patterns. The condition is fulfilled if its first chunk pattern matches the current goal and the remaining chunk patterns match chunks in the declarative memory. An example for a product ion is

IF the current goal is to show that x £ S_2 and it is known that x E S_i and $S\backslash$ C S_2

THEN conclude that x 6 #2 by the definition of C

Similar to the base-level activation of chunks, the strength of a production is defined such that it decreases logarithmically when the production is not used. The time spent to match a production with a chunk depends on the activation of the chunk.[1] It is defined such that it, is negative exponential to the sum of the activation of the chunk and the strength of the production. Hence, the higher the activation of the chunk and the strength of the production, the faster the production matches the chunk. Since the activation must, be greater than the retrieval threshold r, r constrains the time maximally available to match a production with a chunk.

The conflict resolution heuristic starts from assumptions on the probability P that the application of the current production leads to the goal and on the costs C of achieving that goal by this means. Moreover G is the time maximally available to fulfill the goal. The net-utility E of the application of a production is defined as

$$E = \quad PG-G. \tag{1}$$

We do not go into detail on how P, G and C are calculated. For the purposes of this paper, it is sufficient to note that G only depends on the goal, but not on the product ion.

To sum up, in ACT-R the choice of a production to apply is as follows:

1. The conflict set is determined by testing the match of the productions with the current goal.
2. The production p with the highest utility is chosen.
3. The actual instantiation of p is determined via the activations of the corresponding chunks. If no instantiation is possible (because of r), p is removed from the conflict set and the algorithm resumes in step 2. otherwise the instantiation of p is applied.

ACT-R provides a learning mechanism, called *production compilation,* which allows for the learning of new productions. We are currently exploring this mechanism for its utility for the explanation of proofs.

3 The Architecture of *P. rex*

P. rex is planned as a generic explanation system that can be connected to different theorem provers. It, adopts the following features of the interactive proof development environment ΩMEGA [Benzmuller *et at.,* 1997]:

- Mathematical theories are organized in a hierarchical knowledge base. Each theory in it may contain axioms, definitions, theorems along with proofs, as well as proof methods, and control rules how to apply proof methods.
- A proof of a theorem is represented in a hierarchical data structure. This representation makes explicit, the various levels of abstraction by providing several justifications for a single proof node, where each justification belongs to a different level of abstraction. The least abstract, level corresponds to a

[1] In this context Ttime does not mean the CPU time needed to calculate the matchT but the time a human would need for the match according to the cognitive model.

oroof in Gentzen's natural deduction (ND) calculus Gentzen, 1935]. Candidates for higher levels are proof plans, where justifications are mainly given by more abstract proof methods that belong to the theorem's mathematical theory or to an ancestor theory thereof.

An example for a proof is given below. Each line consists of four elements (label, antecedent, succedent, and justification) and describes a node of the proof. The *label* is used as a reference for the node. The *antecedent* is a list of labels denoting the hypotheses under which the formula in the node, the *succedent,* holds.[2] This relation between antecedent and succedent is denoted by K

Label	*Antecedent*	*Succedent*	*Justification*
L_0		$\vdash a \in U \vee a \in V$	J_0
H_1	H_1	$\vdash a \in U$	HYP
L_1	H_1	$\vdash a \in U \cup V$	$\mathrm{Def}\cup(H_1)$
H_2	H_2	$\vdash a \in V$	HYP
L_2	H_2	$\vdash a \in U \cup V$	$\mathrm{Def}\cup(H_2)$
L_3		$\vdash a \in U \cup V$	$\cup\text{-Lemma}(L_0)$
			$\mathrm{CASE}(L_0, L_1, L_2)$

We call $\Delta \vdash \varphi$ the *fact* in the node. The proof of the fact in the node is given by its *justification.* A justification consists of a rule and a list of labels, the *premises* of the node. J_t denotes an unspecified justification. HYP and DefU stand for a hypothesis and the definition of U, respectively. L_3 has two justifications on different levels of abstraction: the least abstract justification with the ND-rule CASE (i.e. the rule for case analyses) and the more abstract justification with the rule U-Lemma that stands for an already proven lemma about a property of U. By agreement, if a node has more than one justification, these are sorted from most, abstract to least abstract.

The proof is as follows: From $a \in U \vee a \in V$ we can conclude that $a \in U \cup V$ by the U-Lemma. If we do not know the U-Lemma, we can come to the conclusion by considering the case analysis with the cases that $a \in U$ or $a \in V$, respectively. In each ca.se, we can derive that $a \in U \cup V$ by the definition of U.

A formal language for specifying proofs is the interface by which theorem provers can be connected to *P. rex.* An overview of the architecture of *P.rex* is provided in Figure 1.

The crucial component of the system is the *dialog planner.* It is implemented in ACT-R, i.e. its operators are defined in terms of productions and the discourse history is represented in the declarative memory by storing conveyed information as chunks (details are given in Section 4). Moreover, presumed declarative and procedural knowledge of the user is encoded in the declarative memory and the production rule base, respectively. This establishes that the dialog planner is modeling the user.

In order to explain a particular proof, the dialog planner first assumes the user's supposed cognitive state by updating its declarative and procedural memories. This is done by looking up the user's presumed knowledge

[2] As notation we use Δ and T for antecedents and φ and ψ for succedents.

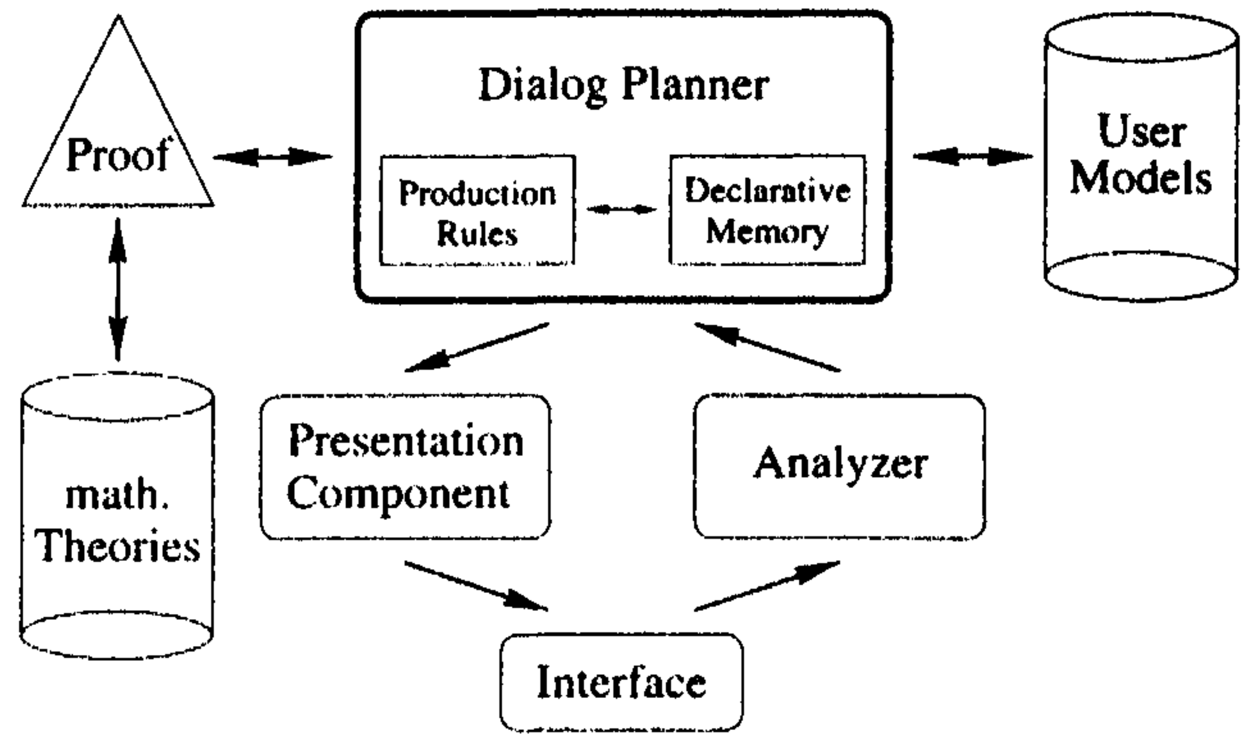

Figure 1: The Architecture of *P.rex*

in the user model, which was recorded during a previous session. An individual model for each user persists between the sessions.

The individual user models are stored in the database of *user models.* Each user model contains assumptions on the knowledge of the user that are relevant to proof explanation. In particular, it makes assumptions on which mathematical theories the user knows, which definitions, proofs, proof methods and mathematical facts he knows, and which productions he has already learned.

After updating the declarative and procedural memories, the dialog planner sets the global goal to show the conclusion of the proof's theorem. ACT-R tries to fulfill this goal by successively applying productions that decompose or fulfill goals. Thereby, the dialog planner not only produces a multimodal dialog plan (see Section 4.1), but also traces the user's cognitive states in the course of the explanation. This allows the system both to always choose an explanation adapted to the user (see Section 4.2), and to react to the user's interactions in a flexible way: The dialog planner analyzes the interaction in terms of applications of productions. Then it plans an appropriate response.

The dialog plan produced by the dialog planner is passed on to the *presentation component.* Currently, we use *PROVERBS* microplanner [Huang and Fiedler, 1997] to plan the scope and internal structure of the sentences, which are then realized by the syntactic generator TAG-GEN [Kilger and Finkler, 1995].

An *analyzer* receives the user's interactions and passes them on to the dialog planner. In the current experimental stage, we use a simplistic analyzer that understands a small set of predefined interactions.

4 The Dialog Planner

In the community of NLG, there is a broad consensus that the generation of natural language should be done in three major steps [Reiter, 1994]. First a *macroplanner (text planner)* determines what to say, i.e. content and order of the information to be conveyed. Then a *microplanner (sentence planner)* determines how to say it, i.e. it plans the scope and the internal structure of the sentences. Finally, a *realizer (surface generator)* produces the surface text. In this classification, the dialog planner is a rnacroplanner for managing dialogs.

As Wahlster *et al.* argued, such a three-staged architecture is also appropriate for multimodal generation [Wahlster *et ai,* 1993]. By defining the operators and the dialog plan such that they are independent of the communication mode, our dialog planner plans text, graphics and speech.

Since the dialog planner in *P. rex* is based on ACT-R, the plan operators are defined as productions. A goal is the task to show the fact in a node *n* of the proof. A production fulfills the goal directly by communicating the derivation of the fact irl *n* from already known facts or splits the goal into new subgoals such as to show the facts in the premises of *n.* The derivation of a fact is conveyed by so-called mathematics communicating acts (MCAs) and accompanied by storing the fact as a chunk in the declarative memory. Hence the discourse history is represented in the declarative memory. ACT-R's conflict resolution mechanism and the activation of the chunks ensure an explanation tailored to the user. The produced dialog plan is represented in terms of MCAs.

4.1 Mathematics Communicating Acts

Mathematics communicating acts (MCAs) are the primitive actions planned by the dialog planner. They are derived from *PROVERBS proof communicative acts* [Huang, 1994]. MCAs are viewed as speech acts that are independent of the modality to be chosen. Each MCA at least can be realized as a portion of text. Moreover some MCAs manifest themselves in the graphical arrangement of the text.

In *P. rex* we distinguish between two types of MCAs:

- MCAs of the first type, called *derivational MCAs,* convey a step of the derivation. An example for a derivational MCA with a possible verbalization is:

  (Derive :Reasons $(a \in U, U \subseteq V)$
  :Conclusion $a \in V$
  :Method Def$\subseteq$)

  "Since a is an element of *V* and U is a subset of V', *a* is an element of V by the definition of subset."

- MCAs of the second type, called *structural MCAs,* communicate information about the structure of a proof. For example, case analyses are introduced by:

  (Case-Analysis **:Goal** ψ
  :Cases (φ_1, φ_2))

  "To prove ψ,, let us consider the two cases by assuming φ_1 and φ_2

4.2 Plan Operators

Operational knowledge concerning the presentation is encoded as productions in ACT-R that are independent from the modality to be chosen. The proof explaining productions are derived from *PROVERB'S* macroplanning operators [Huang, 1994]. Each of those corresponds to one or several productions in *P. rex.*

Each production either fulfills the current goal directly or splits it into subgoals. Let us assume that the following nodes are in the current proof:

Label	*Antecedent*	*Succedent*	*Justification*
P_1	Δ_1	$\vdash \varphi_1$	J_1
		.	
P_n	Δ_n	$\vdash \varphi_n$	J_n
C	Γ	$\vdash \psi$	$R(P_1, \ldots, P_n)$

An example for a production is:

(PI) IF The current goal is to show $\Gamma \vdash \psi$
and *R* is the most abstract known rule justifying the current goal
and $\Delta_1 \vdash \varphi_1, \ldots, \Delta_n \vdash \varphi_n$ are known
THEN produce MCA (Derive :Reasons $(\varphi_1, \ldots, \varphi_n)$:Conclusion ψ :Method R)
and pop the current goal (thereby storing $\Gamma \vdash \psi$ in the declarative memory)

By producing the MCA the current goal is fulfilled and can be popped from the goal stack. An example for a production decomposing the current goal into several subgoals is:

(P2) IF The current goal is to show $\Gamma \vdash \psi$
and *R* is the most abstract known rule justifying the current goal
and '$\Phi = \{\varphi_i | \Delta_i \vdash \varphi_i$ is unknown for $1 \leq i \leq n\} \neq \emptyset$
THEN for each $\varphi_i \in \Phi$ push the goal to show $\Delta_i \vdash \varphi_i$

Note that the conditions of (PI) and (P2) only differ in the knowledge of the premises φ_i for rule *R.* (P2) introduces the subgoals to prove the unknown premises in Φ. As soon as those are derived, (PI) can apply and derive the conclusion.

Now assume that the following nodes are in the current proof:

Label	*Antecedent*	*Succedent*	*Justification*
P_0	Γ	$\vdash \varphi_1 \vee \varphi_2$	J_0
H_1	H_1	$\vdash \varphi_1$	HYP
P_1	Γ, H_1	$\vdash \psi$	J_1
H_2	H_2	$\vdash \varphi_2$	HYP
P_2	Γ, H_2	$\vdash \psi$	J_2
C	Γ	$\vdash \psi$	$\mathrm{CASE}(P_0, P_1, P_2)$

A specific production managing such a case analysis is the following:

(P3) IF The current goal is to show $\Gamma \vdash \psi$
and CASE is the most abstract known rule justifying the current goal
and $\Gamma \vdash \varphi_1 \vee \varphi_2$ **is known**
and $\Gamma, H_1 \vdash \psi$ **and** $\Gamma, H_2 \vdash \psi$ **are unknown**
THEN **push the goals to show** $\Gamma, H_1 \vdash \psi$ **and** $\Gamma, H_2 \vdash \psi$
and produce MCA (Case-Analysis :Goal ψ
:Cases (φ_1, φ_2))

This production introduces new subgoals and motivates them by producing the MCA.

Since more specific rules treat common communicative standards used in mathematical presentations, they are assigned lower costs, i.e. $C_{(P3)} < C_{(P2)}$ (ef. equation 1).

Moreover, it is supposed that each user knows all natural deduction (ND) rules. This is reasonable, since ND-rules are the least abstract possible logical rules in proofs. Hence, for each production *p* that is defined such that its goal is justified by an ND-rule in the proof, the probability P_p that the application of *p* leads to the goal to explain that proof step equals one. Therefore, since CASE is such an ND-rule, $P_{(P3)} = 1$

Before examining more closely an example explanation of a proof, we look at. a production reacting to a user interaction. Consider the case that the user informs the system that he did not understand a step of the derivation. The analyzer receives the user's message and pushes the goal to backtrack to the node *n* whose explanation was not understood. This goal can be fulfilled by the following production:

(P4) IF The current goal is to backtrack to node *n*
THEN push the subgoals to re-explain the fact in *n* and to revise the assumption, that the justification *J* used in its last explanation was known.

A further production (P5), which is omitted here due to space restrictions, performs the revision by decreasing the base-level activation of./.

In order to elucidate how a proof is explained by- *P.rex* let us consider the following situation:

- The following nodes are in the current proof:

Label	*Antecedent*	*Succedent*	*Justification*
L_0		$\vdash a \in U \vee a \in V$	J_0
H_1	H_1	$\vdash a \in U$	HYP
L_1	H_1	$\vdash a \in U \cup V$	Def$\cup$(H_1)
H_2	H_2	$\vdash a \in V$	HYP
L_2	H_2	$\vdash a \in U \cup V$	Def$\cup$(H_2)
L_3		$\vdash a \in U \cup V$	$\cup$-Lemma(L_0) CASE(L_0, L_1, L_2)

- the current goal is to show the fact in L_3,
- the rules HYP, CASK, Def$\cup$, and $\cup$-Lemma are known,
- the fact in L_0 is known, the facts in H_1, L_1, H_2, and L_2 are unknown.

The only applicable production is (PI). Since U-Lemma is more abstract than CASE and both are known, it is chosen to instantiate (PI). Hence, the dialog planner produces the MCA

(Derive :Reasons $(a \in U \vee a \in V)$
:Conclusion $a \in U \cup V$
:Method U-Lemma)

that can be verbalized as "Since $a \in U$ or $a \in V$, $a \in U \cup V$ by the U-Lemma."

Suppose now that the user interrupts the explanation throwing in that he did not understand this step. The analyzer translates the user's interaction into the new goal to backtrack to L3, which is pushed on the goal stack. This goal is processed by (P4) pushing the subgoals to re-explain the fact in L3 and to revise the assumption, that U-Lemma is known. The latter is fulfilled by (P.')) by decreasing the base-level activation of U-Lemma below the retrieval threshold. This leaves the goal to (re-)cxplain the fact in L_3 on the top of the goal stack.

Now, since CASE is the most abstract known rule justifying the current goal, both decomposing productions (P2) and (P3) are applicable. Recall that the conflict resolution mechanism chooses the production with the highest utility E (cf. equation 1). Since $P_{(P3)} = 1$ and $P_p \leq 1$ for all productions />, $P_{(P3)} \geq P_{(P2)}$. Since the application of (P2) or (P3) would serve the same goal, $G_{(P3)} = G_{(P2)}$. Since (P3) is more specific than (P2), $C_{(P3)} < C_{(P2)}$. Thus

$$E_{(P3)} = P_{(P3)}G_{(P3)} - C_{(P3)} > P_{(P2)}G_{(P2)} - C_{(P2)} = E_{(P2)}$$

Therefore, the dialog planner chooses (P3) for the explanation, thus producing the MCA

(Case-Analysis :Goal $a \in U \cup V$
:Cases $(a \in U, a \in V)$)

that can be realized as "To prove $a \in U \cup V$ let us consider the cases that $a \in U$ a $ra \in V$ " and then explains both cases. The whole dialog takes place as follows:

> ***P.rex:*** Since $a \in U$ or $a \in V \Gamma a \in U \cup V$ by the $\cup$-Lemma.
>
> **User:** Why does this follow?
>
> ***P.rex:*** To prove $a \in U \cup V$ let us consider the cases that $a \in U$ and $a \in V$. Let $a \in U$. Then $a \in U \cup V$ by the definition of $\cup$. Let $a \in V$. Then $a \in U \cup V$ by the definition of $\cup$.

This example shows how a production and an instantiation are chosen by *P.rex*. While the example elucidates the case that a more detailed explanation is desired, the system can similarly choose a more abstract explanation if needed. Hence, modeling the addressee's knowledge in ACT-R allows *P. rex* to explain the proof adapted to the user's knowledge by switching between the its levels of abstraction as needed.

Having in mind that the MCAs and the explaining productions are derived from *PROVERBS* macroplanner, it is no surprise that *Prex's* dialog planner produces text plan equivalent to *PROVERB.* But while the proof must be provided to *PROVERB* on an appropriate level of abstraction to satisfv the user, *P. rex* determines for each proof step which level of abstraction it considers as the most appropriate for the respective audience. Moreover, P *rex* can react to interactions by the user and revise both its assumptions about the addressee and its planning decisions.

5 Conclusion and Future Work

In this paper, we proposed to combine the traditional design of a dialog planner with a cognitive architecture in order to strive for an optimal user adaptation. In the interactive proof explaining system *Prer,* the dialog planner is based on the theory of cognition ACT-R.

Starting from certain assumptions about the addressee's knowledge (e.g. which facts does he know, which definitions, lemmas, etc.) built up in the user model during previous sessions, the dialog planner decides on which level of abstraction to begin the explanation. Since ACT-R traces the user's cognitive states during the explanation, the dialog planner can choose an appropriate degree of abstraction for each proof step to be explained. Furthermore, it can react to user interactions and revise the user model. The rationale behind this architecture should prove-to be useful for explanation systems in general.

Moreover, since this architecture can predict what is salient for the user and what he can infer, it could be used as a basis to decide whether or not to include optional information [Walker and Rainbow, 1994].

P. rex is still in an experimental stage. It goes already beyond *PROVERBS* capabilities, since it can not only produce textbook style proofs but also plan explanations tailored to the respective user and react to interactions. We plan to extend the presentation component to multimodality supporting graphics, text, and speech. It should consist of the following subcomponents:

A *multimodal microplanner* plans the scope of the sentences and their internal structure, as well as their graphical arrangement. It also decides, whether a graphical or a textual realization is preferred. Textual parts are passed on to a *linguistic realizer* that generates the surface sentences. Then a *layout component* displays the text and graphics, while a *speech system* outputs the sentences in speech. Hence, the system should provide the user with text and graphics, as well as a spoken output. The metaphor we have in mind is the teacher who explains what he is writing on the board.

Currently, we are examining the knowledge compilation mechanism of ACT-R that could enable the system to model the user's acquisition of proving skills. This could pave the way towards a tutorial system that not only explains proofs, but also teaches concepts and proving methods and strategies.

Moreover, we are planning experiments with users of different levels of expertise in mathematics to evaluate the system.

Acknowledgements

Many thanks go to Jorg Siekmann, Frank Pfenning, Ken Koedinger, and Christian Lebiere for their help in my research. Frank Pfenning, Carsten Sehurmann and Chris Scarpinatto read earlier drafts of this paper. I also want to thank the anonymous reviewers for their useful comments.

References

[Anderson and Lebiere, 1998] J. R. Anderson and C. Lebiere. *The Atomic Components of Thought.* Lawrence Erlbaum, 1998.

[Benzmuller *et a*/., 1997] C. Benzmuller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarsehmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant,. In W. McCune, editor, *Proceedings of the 14th Conference on Automated Deduction,* number 1249 in LNAI, pages 252-255, Townsville, Australia, 1997. Springer Verlag.

[Cawsey, 1990] A. Cawsey. Generating explanatory discourse. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation,* number 4 in Cognitive Science Series, pagers 75 101. Academic Press, San Diego, CA, 1990.

[Fiedler, 1998] A. Fiedler. Macroplanning with a cognitive architecture for the adaptive explanation of proofs. In *Proceedings of the 9th International Workshop on Natural Language Generation,* pages 88-97, Niagara-on-the-Lake, Ontario, Canada, 1998.

[Gentzen, 1935] G. Gentzen. Untersuchungen uber das logische Schlieften I & II. *Mathematische Zeitschrift,* 39:176-210, 572 595, 1935.

[Horacek, 1997] H. Horacek. A model for adapting explanations to the user's likely inferences. *User Modeling and User-Adapted Interaction,* 7:1-55, 1997.

[Huang and Fiedler, 1997] X. Huang and A. Fiedler. Proof verbalization as an application of NLG. In M. E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (1JCAI),* pages 965-970, Nagoya, Japan, 1997. Morgan Kaufmann.

[Huang, 1994] X.Huang. Planning argumentative texts. In *Proceedmgs of the 15th International Conference on Computational Linguistics,* pages 329-333, Kyoto, Japan, 1994.

[INLG, 1994] *Proceedings of the 7th International Workshop on Natural Language Generation,* Kennebunkport, ME, USA, 1994.

[Kilger and Finkler, 1995] A. Kilger and W. Kinkier. Incremental generation for real time applications. Research Report RR-95-11, DKKI, Saarbrucken, Germany, 1995.

[Meyer and Kieras, 1997] D. E. Meyer and D. E. Kieras. EPIC: A computational theory of executive cognitive processes and multiple-task performance: Part 1. *Psychological Review,* 104:3 65, 1997.

[Newell, 1990] A. Newell. *Unified Theories of Cognition.* Havard University Press, Cambridge, MA, 1990.

[Paris, 1991] C. Paris. The role of the user's domain knowledge in generation. *Computational Intelligence,* 7:71-93, 1991.

[Reiter, 1994] E. Reiter. Has a consensus NL generation architecture appeared, and is it psyeholinguistieally plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation* [1994], pages 163 170.

[Wahlster *et al,* 1993] W. Wahlster, E. Andre, W. Kinkier, H.-J. Profitlich, and T. Rist. Plan-based integration of natural language and graphics generation. *Artificial Intelligence,* 63:387 427, 1993.

[Walker and Rambow, 1994] M. A. Walker and 0. Hambow. The role of cognitive modeling in achieving communicative intentions. In *Proceedings of the 7th International Workshop on Natural Language Generation* [1994], pages 171 180.

[Zukerman and McConachy, 1993] I. Zukerman and R. McConachy. Generating concise discourse that addresses a user's inferences. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI),* pages 1202 1207, Chambery, Prance, 1993. Morgan Kaufmann.