Production Composition: A Simple Theory of Complex Skill Acquisition

NIELS A. TAATGEN[1], University of Groningen, The Netherlands, and FRANK J. LEE, Rensselaer Polytechnic Institute, USA

In psychology many of theories of skill acquisition have had great success in addressing the fine details of learning relatively simple tasks, but can they scale up to complex tasks that are more typical of human learning in the real world? In this paper we describe production composition, a theory of skill acquisition that combines aspects of the theories forwarded by Anderson (1982) and Newell and Rosenbloom (1981), that we believe can model the fine details of learning in complex and dynamic tasks. We use production composition to model in detail learning in a simulated air-traffic controller task.

[1]Requests for reprints should be sent to Niels A. Taatgen, Artificial Intelligence, University of Groningen, Grote Kruisstraat 2/1, 9712 TS Groningen, The Netherlands. Email: niels@ai.rug.nl

Running Title: SKILL ACQUISITION

Keywords: skill acquisition, cognitive modeling, ACT-R,

INTRODUCTION

From controlling air-traffic to operating nuclear power plants, researchers in human factors have accumulated extensive empirical knowledge on human performance in complex and dynamic tasks. However development of detailed computational models that can explain how people are able to problem solve and learn such tasks has lagged behind. One reason is that most researchers have focused on developing computational models of learning simple tasks, while very few researchers have attempted to develop models of skill development in complex and dynamic tasks. Although we have gained deep insights from studying simple tasks, in order to move toward a complete theory of skill acquisition we need to develop and test our model against complex and dynamic tasks that are more typical of human learning in the real world. Toward this goal, we describe production composition, a theory of skill acquisition that combines aspects of theories proposed by Anderson's (1982) and Newell and Rosenbloom's (1981). We use production composition to develop a detailed model of learning in a simulated air-traffic control task.

Production composition

Skill acquisition is usually characterized as going through three stages: the cognitive stage, the associative stage, and the autonomous stage (Fitts, 1964). The three stages can be characterized as moving from conscious processing, slow and error-prone to unconscious processing, fast and error-free. Anderson (1982) explained these three stages in terms of a transition from declarative knowledge to procedural knowledge. In the cognitive stage knowledge is declarative and needs to be interpreted. Interpreting knowledge is slow and may lead to errors if the relevant knowledge cannot be retrieved at the right time. Procedural knowledge on the other hand is compiled and therefore fast and free of errors and is associated with the autonomous stage. The asso-

ciative stage is a transitional stage during which knowledge is partly declarative and partly compiled.

Newell and Rosenbloom (1981) proposed an alternate theory of skill acquisition called chunking that became an important component of the Soar cognitive architecture (Newell, 1990). Within Soar, skill acquisition process is carried out by combining production rules and by converting the current goal context into new, more specialized rules. These specialized rules summarize all the processing required to achieve a specific subgoal. If Soar later encounters a similar problem, it no longer needs a separate subgoal to process it. Instead it can use the learned rule to solve it in a single step.

Production composition incorporates aspects of both Anderson's and Newell and Rosenbloom's account of skill acquisition. It involves both the compilation of declarative knowledge into procedural knowledge and the combination of multiple production rules into new single rules. Production composition has already been successfully used to model inflection of the English past tense (Taatgen & Anderson, submitted) and the German plural (Taatgen, 2001b), and strategy development in the balanced-beam task (van Rijn, van Someren & van der Maas, submitted). We use production composition to develop a detailed computation model of learning in the Kanfer-Ackerman Air Traffic Controller (KA-ATC) Task (Ackerman, 1988; Ackerman & Kanfer, 1994). We are using the KA-ATC task, because while it is a reasonably complex task that simulates dynamic aspects of real air traffic control (e.g., planes lose fuel and weather conditions change), it is simple enough to be tractable for study. In addition, Ackerman has collected data from over 3500 subjects on the KA-ATC task and has made them available on a CD-ROM (Ackerman & Kanfer, 1994) to the Office of Naval Research (ONR).

THE TASK

The KA- ATC task is composed of the following elements displayed on the screen: (a) 12 hold positions, (b) 4 runways, (c) information on current score, landing points, penalty points, conditions of the runways, and wind direction and speed, (e) a queue of planes waiting to enter the hold, and (f) 3 message windows, 1 for notifying of weather changes, 1 for providing feedback on errors, and 1 for displaying of the rules of the task in response to information requests by the participants. The 12 hold positions are divided into 3 levels corresponding to altitude, with hold level 3 being the highest and hold level 1 being the lowest. A typical display of the KA-ATC task is presented in Figure 1.

Six rules govern participant's actions in this task: (1) Planes must land into the wind, (2) Planes can only land from hold level 1, (3) Planes can only move 1 hold level at a time, but to any open position in that level, (4) Ground conditions and wind speed determine the runway length required by different plane types. In particular, 747's always require long runways, DC10's can use short runways only when runways are dry or wet, and wind speed is less than 40 knots, 727's can use short runways only when the runways are dry or wind speed is 0-20 knots, and PROP's can always use short runways, (5) Planes with less than 3 minutes of fuel remaining must be landed immediately, and (6) Only one plane at a time can occupy a runway.

Participants can execute three actions in this task: (a) they can accept a plane from the queue into an open hold-position, (b) they can move a plane between the three hold-levels, and (c) they can land a plane on a runway. They can accomplish these actions by using four keys: the Up-arrow and the Down-arrow keys, ↑ and ↓; the F1 function key, **F1**; and the Enter key, ↵. They can move the cursor up and down the hold-positions and the runways using the ↑ key and the ↓ key. They can accept a plane from the queue into an open hold-position using the **F1** key. And, they

can select a plane in the hold, place a selected plane in an open hold-position (either from the queue or from another hold-position), or land a plane on a runway using the ↵ key. In addition, participants can press the number keys **1** - **6** to examine the rules 1 - 6 any time during the task. They are given 50 points for landing a plane, penalized 100 points for crashing a plane, and penalized 10 points for violating one of the six rules. A plane crashes when the fuel-level of a plane falls to 0 minutes. Planes are added to the queue approximately every 7 seconds and it takes 15 seconds for a plane to clear a runway. Once planes enter the hold position from the queue, they have between 4 - 6 minutes of fuel and begin to lose fuel in real time. In Ackerman (1988), participants performed in the fair-weather condition where the wind speed was fixed to 0 – 20 knots and the runway condition was fixed to DRY. Under this condition, Rule 4 simplifies to the rule that all planes, except 747s, can land on the short runway.

Lee and Anderson (2000) developed a model of expert performance in the KA-ATC task using ACT-R/PM (Byrne & Anderson, 1998) cognitive architecture. In their model, Lee and Anderson showed that expertise in the KA-ATC task require a substantial degree parallelism between cognition, perception, and action. Indeed, as people become skilled in the KA-ATC task, their performance was largely limited by the constraints on the motor system. In addition, Taatgen (2001a) developed an ACT-R model of the impact of individual differences on performance and learning speed. By manipulating ACT-R parameters corresponding to working memory capacity, speed of production composition and psychomotor speed, he found the same pattern of correlations between individual differences and performance as Ackerman (1988) found in his empirical analysis. The model we present in this paper builds on the model described in Taatgen (2001a). In the current model, we account for perceptual and motor interaction with the interface and build on the task analysis of the KA-ATC task by Lee and Anderson (2001).

Task Analysis

Figure 2 illustrates Lee and Anderson's (2001). decompositional task-analysis of the KA-ATC task. The task analysis is based on Card, Moran, and Newell (1983) method of unit task analysis in which a task is decomposed into increasingly specific goals, all the way down to the keystroke level of elementary cognitive and perceptual-motor goals. As can be seen in Figure 2, the KA-ATC task can be decomposed into 3 unit-tasks. They are (a) moving a plane between the hold-levels, (b) landing a plane on a runway, and (c) getting a plane from the queue into a hold-position. As Figure 2 further illustrates each unit-task can be decomposed into a number of functional-level goals. For instance the unit-task of landing a plane involves (1) finding a plane to land, (2) moving to the plane, (3) selecting the plane, (4) finding a runway to land, (5) moving to the desired runway, and (6) landing the plane. Each of these functional-level goals involves a number of keystroke-level goals, including a sequence of shifts of attention across the screen, encoding of information on the screen, and a keystroke to effect the desired action.

THE MODEL

The model of the KA-ATC task is based on the ACT-R cognitive architecture (Anderson & Lebiere, 1998). In particular, we use version 5.0 of the architecture, which incorporates many new theoretical elements including the perceptual motor extension described in Byrne & Anderson (1998). First we give an overview of the architecture after which we will proceed to discuss the model. We then proceed to discuss in more detail some aspects of ACT-R that are particularly important for our model.

An overview of the ACT-R architecture

The theoretical foundation of the ACT-R architecture is rational analysis of human cognition (Anderson, 1990). According to rational analysis, each component of the cognitive system is

optimized with respect to the demands from the environment given its computational limitations. The main components in ACT-R are a declarative (fact) memory and a production (rule) memory. ACT-R is a hybrid architecture in that it has both symbolic and sub-symbolic aspects. We describe these components informally. Further details about the ACT-R architecture can be found in Anderson and Lebiere (1998).

Items in declarative memory, called chunks, have different levels of activation to reflect their use: chunks that have been used recently or chunks that are used very often receive a high activation. This activation decays over time if the chunk is not used. In addition, chunks cannot act by themselves; they need production rules for their application. In order to use a chunk, a production rule has to be invoked to retrieve it from declarative memory and another rule to do something with it. Since ACT-R is a goal-driven theory, chunks are always retrieved to achieve some goal. In the context of the KA-ATC task there are several goals. While only one goal can be active at a time, a model may consist of elaborate strategies to switch between these goals. One of the goals may be to land a plane for which it may be necessary to hand over the control to a lower-level goal, e.g. a goal to move the arrow on the screen to the desired plane.

The behavior of production rules is also governed by the principle of rational analysis. Each production rule has a real-valued quantity associated with its expected outcome. Expected outcome is calculated from estimates of the cost and probability of reaching the goal if that production rule is chosen. The unit of cost in ACT-R is time. ACT-R's learning mechanisms constantly update these estimates based on experience. If multiple production rules are applicable for a certain goal, the production rule is selected with the highest expected outcome.

In both declarative and procedural memory, selections are made on the basis of some evaluation, either activation or expected outcome. This selection process is noisy, so the item with the

highest value has the greatest probability of being selected but other items get opportunities as well. This may produce errors or suboptimal behavior but also allows the system to explore knowledge and strategies that are still evolving. In addition to the learning mechanisms that update activation and expected outcome, ACT-R can also learn new chunks and production rules. New chunks are learned automatically: each time a goal is completed it is added to declarative memory. If an identical chunk is already present in memory, both chunks are merged and their activation values are combined. Chunks acquired through perception, information on the screen for example, are also stored. New production rules are learned on the basis of specializing and merging existing production rules. Since this process is quite crucial in our model we will examine it in more detail later on.

Description of the model

The basis for the model is the idea that instructions are represented in declarative memory and need to be retrieved and interpreted. The production rules that interpret the declarative instructions are not task-specific and can be used for other tasks as well. The declarative representation that is used in our model is a mixture of ideas expressed by Taatgen (1999) and by Anderson (2000). The declarative representation is organized at two levels: the level of goals, and the level of individual steps that have to be taken within a goal. The individual steps for a goal are organized sequentially. For example, to land a plane, the instruction is to first scan hold-level 1 to see whether there are any planes, then to scan the wind direction and then to scan the occupancy of the two runways that are in the direction of the wind. Based on this information, another goal is selected: if both runways are occupied or there is nothing in hold-level 1, a goal is selected to move a plane from hold-level 2 to hold-level 1. If only the short runway is free, a goal that searches for a non-747 in hold-level 1 is selected.

Below is an example sequence of declarative instructions, which is part of the land-strategy. In this example, land2 and land3 are identifiers for two chunks. The other lines are slots of the chunk with slot-values. Slot-values are usually references to other chunks, so "task land" means that there is a slot called task, which refers to another chunk with identifier land.

```
(land2
     isa instruction
     task land
     action scan
     arg1 wind-direction
     arg2 none
     prev land1)
```

```
(land3
     isa instruction
     task land
     action remember-string
     arg1 loc1
     arg2 none
     prev land2)
```

Each instruction refers to the goal it applies to in its task slot (in this case land). It then specifies some action in its action slot. This action may be supplied with additional arguments in the arg1 and arg2 slots. These arguments may be constants, for example wind-direction, or maybe refer to information stored or to be stored in the goal, for example loc1. Finally the prev slot links instructions into a list, specifying in this example that land2 follows land1, and land3 follows land2. In this particular example, the scan action in land2 instructs ACT-R to move its eyes to the location on the screen where the wind-direction is displayed, and to perceive what is there. The next instruction, remember-string in land3, instructs ACT-R to store the perceived string in the goal.

Each of these instructions has to be carried out by a set of production rules that interpret these instructions. For example, carrying out a scan instruction involves the following steps, each of which requires its own production rule:

1. A request is made to declarative memory to retrieve the next instruction

2. Once declarative memory produces the instruction, it is stored in the goal

3. A request is made to declarative memory to retrieve the location of the object that has to be perceived

4. Once declarative memory produces the location of the object, a command is issued to the visual system to move the eyes to that location

5. Once the visual system indicates the eyes are at the desired location, a command is issued to the visual system to perceive to object at the current eye location.

The advantage of breaking down an instruction into these steps is that each of the steps is independent of the current task. Initially, all the task-specific knowledge is in declarative memory and all the task-independent knowledge is in procedural memory. As we will see later declarative knowledge will gradually be compiled into procedural knowledge producing task-specific production rules and speeding up the process considerably.

Except for the instructions, other declarative knowledge controls the switching between goals. In general, there is declarative knowledge that specifies what to do when a goal is completed successfully, what to do when a goal fails, and what to do at explicit choice points in a goal. For example, there is a chunk that specifies that if the land goal fails (which happens when it cannot find a plane in hold-level 1), ACT-R should switch to the move goal. There is another chunk that specifies that once the land goal has arrived at the do-landing action, and its arguments are f and f, indicating that both runways are free, ACT-R should switch to the land-f-f goal. Figure 3 shows the full goal structure of the model. Finally, locations of objects on the screen are stored in declarative memory, for example the fact that hold-level-3 is in the top-left corner of the screen. The procedural knowledge contains no task-specific rules. It has rules to implement the above goal-switching scheme and can perform the operations in Table 1.

It has to be noted that the particular strategy that we use in our model is not the only possible strategy. For example, while our strategy will only get one plane from the queue and will then check whether it is possible to land a plane or move a plane, an alternate strategy is to bring a number of planes from the queue in sequence, speeding up the queuing process but potentially missing landing opportunities. In addition, for some participants declarative strategy will only develop over time. This aspect of the learning is not currently modeled.

An important feature of the model is that it interacts with the experimental software (the version used by Lee & Anderson, 2001, for their experiments) by issuing perceptual and motor commands. When the model runs, the interaction between the model and the experiment can be followed on the screen where a red circle indicates the current focus of visual attention.

Important aspects of ACT-R for the KA-ATC model

A central aspect of ACT-R 5.0 is the notion of buffers. ACT-R has a fixed set of buffers that can contain information of a specific type. There is a visual-location buffer that contains the current focus of visual attention, a visual buffer that contains the contents of the currently attended location, a manual buffer that stores the current motor command, etc. Besides buffers that are used for perceptual and motor operations, there is a buffer that contains the current goal, and a buffer that contains the chunk that has been retrieved from declarative memory most recently.

Production rules match the current contents of the buffers on the left hand side and may issue changes to the buffers on their right hand side. These changes have different effects for different buffers. For example, a change in the visual-location buffer will direct visual attention to a new location. A change in the retrieval buffer will issue a new retrieval request to declarative memory, and a change in the manual buffer will indicate a motor command that has to be carried

out. Changes in the goal may change certain slots of the goal or may replace the current goal with a new goal. If the contents of a certain buffer are replaced by something new, the old contents are stored as a chunk in declarative memory. Although the central core of ACT-R acts serially, the different subsystems, the visual system, declarative memory, the motor system, act asynchronously. For example, while declarative memory is busy with some retrieval, a production rule not involving declarative memory may issue a command to move visual attention to a new location.

Production composition

New production rules in ACT-R are learned by combining two rules that fire in sequence into one new rule. But because the power of production rules in ACT-R is constrained by the buffer structure - it is not possible for example to issue two retrieval requests at the same time - some method of modification is necessary to produce well formed rules. The prototypical case of production composition is when a first rule issues a retrieval request to declarative memory, and a second rule harvests this information and uses it to take some further action. In order to combine these types of rules, the declarative retrieval has to be factored out of the rule by substituting the variables in both production rules by the content of the retrieval. For example, in the case of scanning the wind-direction, the first two steps

1. A request is made to declarative memory to retrieve the next instruction

2. Once declarative memory produces the instruction, it is stored in the goal

along with the declarative retrieval that states the next action is to scan the wind-direction, are combined into a new rule:

a. Set the action to scan the wind-direction

The next two steps in the process,

3. A request is made to declarative memory to retrieve the location of the object that has to be perceived

4. Once declarative memory produces the location of the object, a command is issued to the visual system to move the eyes to that location

are combined into a new rule:

b. Move the eyes to the location of the wind direction

The 2 new rules are much more efficient than the 4 old rules. Not only is the number of steps decreases from 4 to 2, but also 2 retrievals from declarative memory are save. The 2 new rules can now be combined into 1 new rule:

c. Set the action to scan the wind-direction and move the eyes to the location of the wind-direction

This final rule is relatively straight forward as both parent rules do not have any conflicts in their buffer use. As the final action of this rule is to move the eyes to a new location, and the next step in the process is to do something at this new location, no further compilation is possible since ACT-R has to wait for the vision module to complete its action. While it is possible to optimize even more by using the cognitive "slack time", this issue is not explored in the current model.

Production composition has a number of properties that are important to the current model. By substituting declarative retrievals into rules, general production rules are compiled into task-specific rules, and this speeds up the process by reducing the number of production firings. But more importantly, it also reduces the number of retrievals from declarative memory, operations that are generally slow and error-prone. The diminished demand on declarative memory opens the possibility of additional parallelism in the cognitive process.

New production rules are introduced gradually into the system. Once a new rule is introduced, its expected gain parameters are based on the parameters of the parent rules but with an added penalty. So the new rules start out with a disadvantage, but if they prove to outperform their parents over time their parameters will improve and will eventually win the competition.

COMPARISON

To judge the accuracy of the model, we compare the model predictions with data from Study 2 in the ONR data set (Ackerman & Kanfer, 1994), as reported in Ackerman (1988). The data from Study 2 were from 65 college undergraduates who completed 27 trials of the KA-ATC task with each trial lasting 10 minutes. For our model comparisons we only use trials 1 through 10, all in the fair-weather condition. We compare the model predictions and the data at 3 levels of detail: (1) overall performance, (2) unit task level performance, and finally (3) keystroke level performance. In order to get model predictions, we ran the model five times for ten trials, and the results were averaged. All ACT-R parameters were set to their default values.

Overall performance

As a measure for overall performance we take the number of planes that are landed within a 10 minutes trial. Figure 4 shows the data and the predictions of the model. Initially, the model outperforms the participants by almost a factor of two. But after a few trials, the match between model and data is quite good ($R^2 = 0.97$). The initial advantage of the model is probably due to the fact that it has all the declarative knowledge it needs to do the task, while participants still have to gather some of the information. For example, Lee and Anderson (2001) attributed most of the initial speedup of participants to the fact that they started to learn where all the visual information on the screen is located, so that unnecessary eye-movements can be avoided. The present model initially has all the visual locations in declarative memory and does not need to search for them.

Also, the model starts out with a reasonably efficient strategy in declarative memory, which may be true for some of the participants but certainly not for all of them.

Performance at the unit task level

As has been indicated in the task analysis, 3 main unit tasks can be identified: landing a plane, moving a plane between hold levels, and getting a plane from the queue. Figure 5 shows the results of experiment and the model predictions for the time it takes to complete each of the 3 unit tasks. As can be seen, while the fit between the model and the data for the land unit task is very good ($R^2 = 0.98$), the model is generally slower than the participants in the experiment for the remaining 2 unit tasks. This is especially true for the move unit task where, apart from the first trial, the model is much slower. This mismatch can probably be explained by two factors. First, the strategy used by the model is to check whether a landing is possible before attempting a move unit task, and hence the model will never do consecutive move unit tasks without checking the wind direction and the occupancy of the runways first. Second, the model only uses a few move unit tasks in each trial. It only uses moves planes in the hold to clear the initial contents of hold levels 2 and 3, and only moves new planes from the queue into hold level 1. As a consequence, its experience with the move unit task remains limited while participants using other strategies may gain more experience with this particular unit task. For the queue unit task, apart from the initial trial, the participants are slightly faster than the model ($R^2 = 0.91$). This may again be due the model only doing a queue unit task after checking the unavailability of other options, i.e. land and move unit tasks. Nevertheless the match is quite reasonable especially since we are using the default ACT-R parameters.

Performance at the keystroke level

With respect to analyzing the performance at the keystroke level, we limit ourselves to the land unit task since this is the most complicated of the unit tasks. As discussed previously, there are 6 types of keystroke level goals for the land unit task: find plane, move to plane, select plane, find runway, move to runway, and select runway. Figure 6 shows the time to complete these keystroke level goals for both the data and the model ($R2 = 0.87$). If we compare the predictions of the model and the performance of the participants, the qualitative pattern is the same. That is, the keystroke times are in the right order, the slowest keystrokes for the model also being the slowest keystrokes for the participants, and also the learning patterns are very similar. The main discrepancy between the model and the data is the scale: the model is slower, particularly for the more "cognitive" keystrokes like finding and selecting a plane. This may seem odd because at the unit task level the model predicted the data quite accurately. One explanation for this is that the model does not make many unnecessary keystrokes while participants do. For example, if a participant moves the arrow a couple of keystrokes towards a certain plane and then decides to select another plane, the finding of the second plane is actually counted as "move to plane". In this sense, the "find a plane" latencies are diluted into other categories of keystrokes, mainly "move to plane" keystrokes, where the model is actually faster than the participants. Again this reflects the fact that the model already starts out with an accurate representation of what to do in the task whereas participants sometimes still have to find out at the time of the task.

CONCLUSION

Production composition is a powerful mechanism that combines task-specific declarative knowledge and general procedural knowledge into task-specific production rules. Although this has not been explored in the present model, production composition can incorporate experiences

gained during task performance. Taatgen (2001a) showed some examples of this in an earlier ACT-R model in which illegal weather/plane/runway combinations were remembered by the model and later compiled into rules that tried to avoid these combinations.

Another aspect of learning in the KA-ATC task that is not fully explored by the present model is the tight interleaving of perceptual, motor and declarative processes. The model has no strategies to use cognitive "slack time", i.e. periods of time when it is waiting for one of the external actions to finish (e.g., moving the eyes, pushing a key, retrieving something from memory). Such integration is necessary (Lee & Anderson, 2000) if the model is to reach the level of performance of the participants after 18 trials. But since the decision to use of slack time has to be fast, it has to be composed into production rules to be useful. This points to production composition as the potential learning mechanism.

In addition, the model does not fully match the data in that it already starts out with a complete strategy for doing the task. Although participants can be assumed to pick up some of this knowledge during the presentation of the instructions, it is safe to assume that they still have some discovery to do during the task, such as finding locations on the screen (Lee & Anderson, 2001). The fact that this strategy-discovery phase is not modeled also prevents the model from capturing the individual differences in strategy, as pointed out by John and Lallement (1997). They not only found that participants start out with different strategies but also that they change strategies during the task. Despite the lack of strategy evolution in our model, it does show one strategic transition: in the first few trials the model uses what John and Lallement call a sequential strategy: get one plane from the queue, land it, repeat. After practice, however, this changes into an opportunistic strategy: try to keep hold level 1 filled with planes but give priority to landing planes. The model's explanation is simple: at first the model can't keep up with the planes that are landed on the run-

way, so when it gets a plane from the queue there is a runway available to land it. With practice the model becomes faster so it is able to get more planes from the queue than it can land thereby filling up hold level 1.

A point of critique on the ACT-R theory is that it has too many free parameters that enable people to model any data they wish. In ACT-R 5.0, care has been taken to fix the value of most of the free parameters. The added constraint that the model has to interact with the actual interface, constrained by empirical knowledge embodied in the visual and motor system, prevents easy adjustments to the model to match the data. We therefore chose to base our model solely on default values for parameters and have not taken any additional steps to optimize them to fit the data.

Despite the fact that the model does not fully capture the strategy and tight perceptual and motor integration aspects discussed above, it can be extended to incorporate them. For example, the use of slack time can potentially be modeled by incorporating production rules that act in periods of slack time. In addition, other strategic knowledge can be added to the model to search for certain information on the screen instead of already incorporating it in the initial declarative knowledge. Finally, knowledge can be added to memorize experience and retrieve it at the appropriate moment. What the model currently lacks is general procedural knowledge, in order to provide even better account of the data. Regardless, production composition has been shown to be a compelling mechanism for modeling learning in the KA-ATC task.

REFERENCES

Ackerman, P. L. (1988). Determinants of individual differences during skill acquisition: Cognitive abilities and information processing. Journal of Experimental Psychology: General, 117, 288-318.

Ackerman, P. L., & Kanfer, R. (1994). Kanfer-Ackerman air traffic controller task© CD-ROM database, data collection program, and playback program. Office of Naval Research, Cognitive Science Program.

Anderson, J. R. (1982). Acquisition of cognitive skill. Psychological Review, 89, 369-406.

Anderson, J.R. (1990). The adaptive character of thought. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (2000). Learning from instructions. Proceedings of the Seventh Annual ACT-R Workshop. Carnegie Mellon University. Pittsburgh, PA.

Anderson, J.R., & Lebiere, C. (1998). The atomic components of thought. Hillsdale, NJ: Erlbaum.

Byrne, M. D., & Anderson, J. R. (1998). Perception and Action. In J. R. Anderson & C. Lebiere (Eds.), The atomic components of thought (pp. 167-200). Mahwah, NJ: Erlbaum

Card, S. K., Moran, T. P., & Newell, A. (1983). The psychology of human-computer interaction. Hillsdale, NJ: Erlbaum.

Fitts, P.M. (1964). Perceptual-motor skill learning. In A.W. Melton (Ed.), Categories of human learning. New York, NY: Academic Press.

Newell, A. (1990). Unified theories of cognition. Cambridge, M.A.: Harvard University Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), Cognitive skills and their acquisition (pp. 1-55). Hillsdale, NJ: Erlbaum.

John, B.E. & Lallement, Y. (1997). Strategy use while learning to perform the Kanfer-Ackerman air traffic controller task. In M.G. Shafto & P. Langley (Eds.), Proceedings of the Nine-

teenth Annual Conference of the Cognitive Science Society (pp. 337-342). Mahwah, NJ: Erlbaum.

Lee, F.J. & Anderson, J.R. (2000). Modeling Eye-Movements of Skilled Performance in a Dynamic Task. In N.A. Taatgen, & J. Aasman (Eds.), Proceedings of the Third International Conference on Cognitive Modeling. Veenendaal, The Netherlands: Universal Press.

Lee, F.J. & Anderson, J.R. (2001). Does learning a complex task have to be complex? A Study in Learning Decomposition. Cognitive Psychology, 42, 267-316.

Taatgen, N.A. (1999). Learning without limits. Ph.D. Thesis, University of Groningen, The Netherlands.

Taatgen, N.A. (2001a). A model of individual differences in learning Air Traffic Control. In E.M. Altmann, A. Cleeremans, C.D. Schunn, & W.D. Gray (Eds.), Proceedings of the Fourth International Conference on Cognitive Modeling. Mahwah, NJ: Lawrence Erlbaum Associates.

Taatgen, N.A. (2001b). Extended the past tense debate: a model of the German plural. In K. Stenning & J. Moore (Eds.), Proceedings of the Twenty-Third Annual Meeting of the Cognitive Science Society. Mahwah, NJ: Erlbaum.

Taatgen, N.A. & Anderson, J.R. (submitted). Why do children learn to say “Broke”? A model of the past tense without feedback. Submitted to Cognition.

van Rijn, H., van Someren, M, & van der Maas, H. (submitted). Modeling developmental transitions on the balance scale task.

Table 1.

Operations that can be performed by task-independent procedural knowledge

Operation	Description
Scan Scan-seek Scan-empty	The scan operations move visual attention to a certain region on the screen, focus on an object in that region, and perceive that object. Scan-seek will only focus on objects that have not been recently attended, and Scan-empty will focus on empty space instead of text (necessary to find empty slots in the hold-levels.
Remember-loc Remember-string Remember-status	The remember operations store information from the current visual location or object in the goal. Remember-loc stores the current visual location. Remember-string parses the currently attended word and stores it in the goal. Remember-status parses a runway-string and stores its status (free or occupied) in the goal.
Press-enter Press-F1	Operations to press the ↵ and **F1** keys, respectively.
Move-to-loc	Operation to move the cursor to a specified position on the screen by repeated pressing of the arrow keys.
Compare-restart	Compare whether the two arguments of the action are equal. If this is the case, restart the present goal, else continue. This operation is used in combination with scan-seek to find a non-747 in hold-level 1.

Figure Captions

Figure 1. The Kanfer-Ackerman Air Traffic Controller Task.

Figure 2. A hierarchical task decomposition of the Kanfer-Ackerman ATC task.

Figure 3. Structure of the goals used by the model.

Figure 4. Number of planes landed by the participants and the model.

Figure 5. Unit task performance by the participants and the model for (a) land unit task, (b) move unit task, and (c) queue unit task.

Figure 6. Keystroke level performance on the landing unit task of the participants (top) and the model (bottom).

Figure 1.

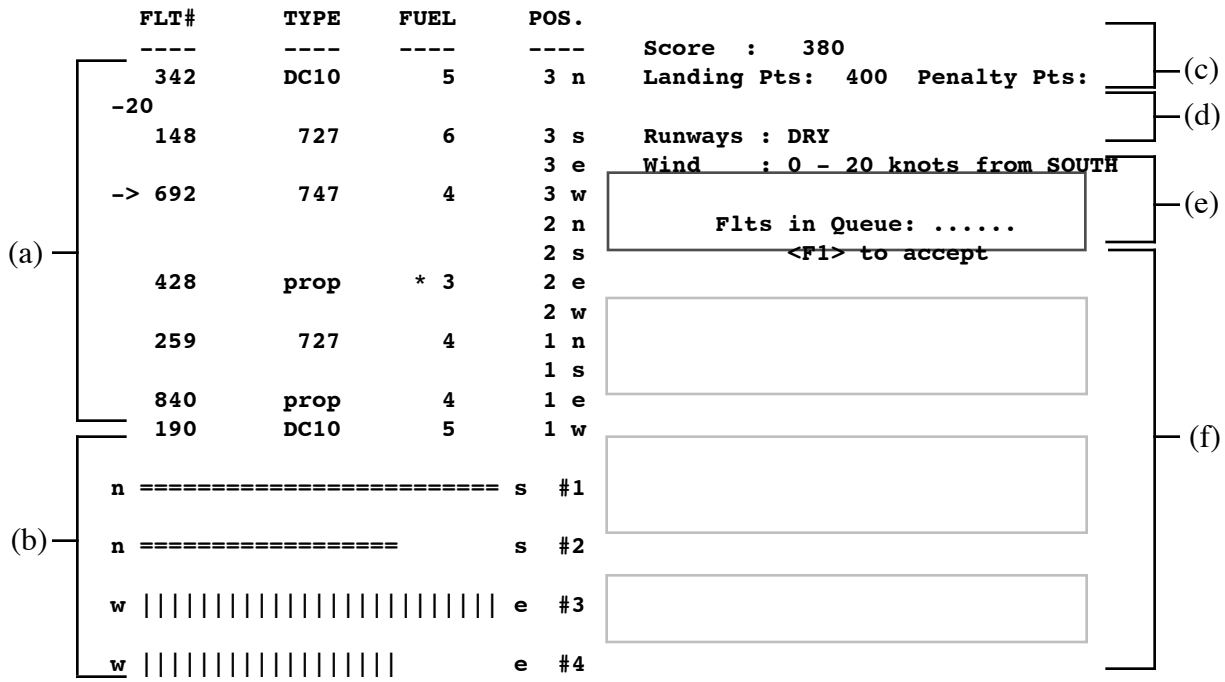

FLT# TYPE FUEL POS.
342 DC10 5 3 n
-20
148 727 6 3 s
3 e
-> 692 747 4 3 w
2 n
2 s
428 prop * 3 2 e
2 w
259 727 4 1 n
1 s
840 prop 4 1 e
190 DC10 5 1 w
n ========== s #1
n ========== s #2
w ========== e #3
w ========== e #4
Score : 380
Landing Pts: 400 Penalty Pts:
Runways : DRY
Wind : 0 - 20 knots from SOUTH
Flts in Queue: ......
<F1> to accept
(a)
(b)
(c)
(d)
(e)
(f)

Figure 2.

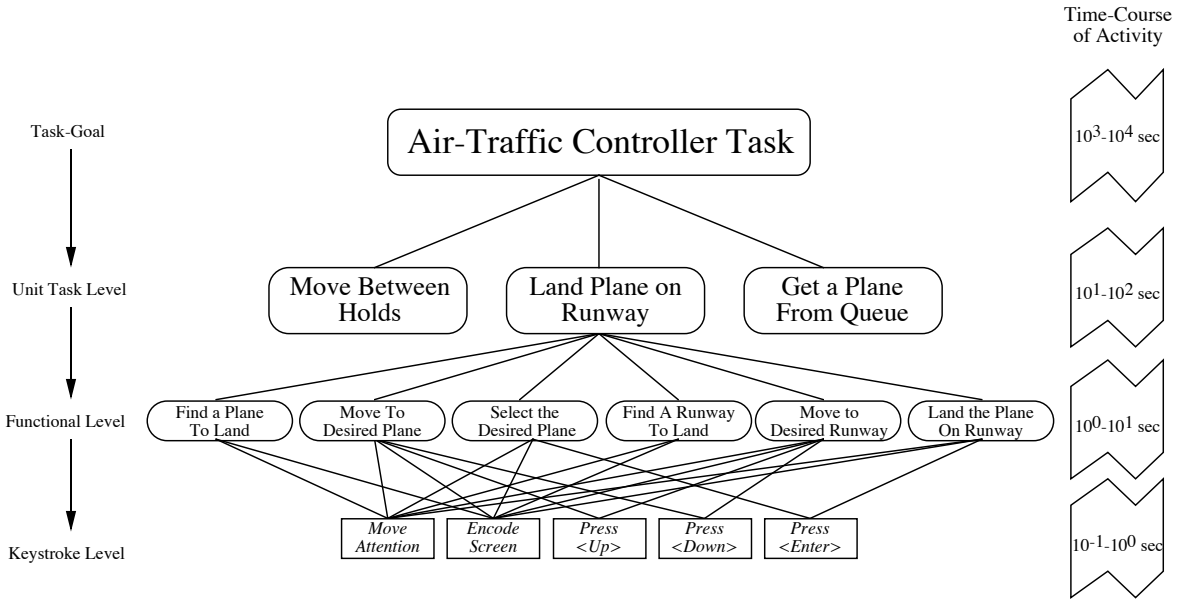

Time-Course of Activity
Task-Goal
Unit Task Level
Functional Level
Keystroke Level
Air-Traffic Controller Task
Move Between Holds
Land Plane on Runway
Get a Plane From Queue
Find a Plane To Land
Move To Desired Plane
Select the Desired Plane
Find A Runway To Land
Move to Desired Runway
Land the Plane On Runway
Move Attention
Encode Screen
Press <Up>
Press <Down>
Press <Enter>
10^3-10^4 sec
10^1-10^2 sec
10^0-10^1 sec
10^{-1}-10^0 sec

Figure 3.

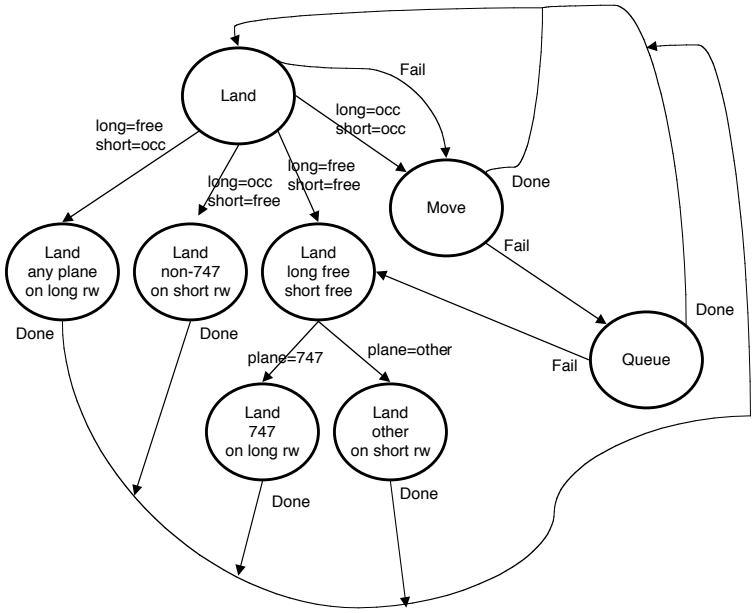

Land
Fail
long=free
short=occ
long=occ
short=occ
long=occ
short=free
long=free
short=free
Move
Done
Land
any plane
on long rw
Land
non-747
on short rw
Land
long free
short free
Fail
Done
Done
Done
plane=747
plane=other
Fail
Queue
Land
747
on long rw
Land
other
on short rw
Done
Done

Figure 4.

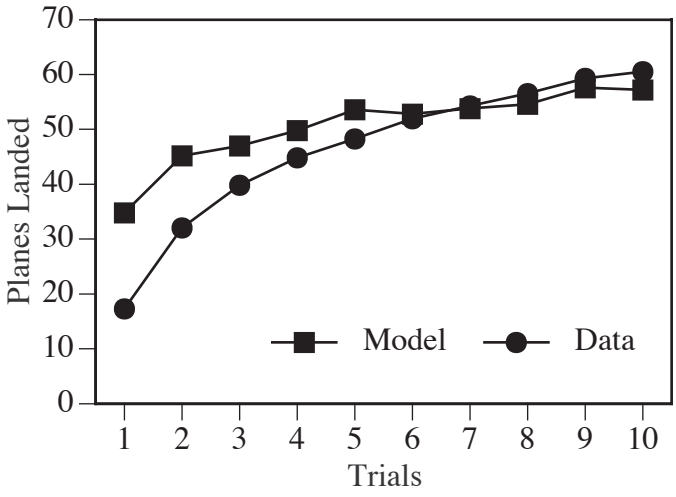

70
60
50
40
30
20
10
0
Planes Landed
Model
Data
1
2
3
4
5
6
7
8
9
10
Trials

Figure 5

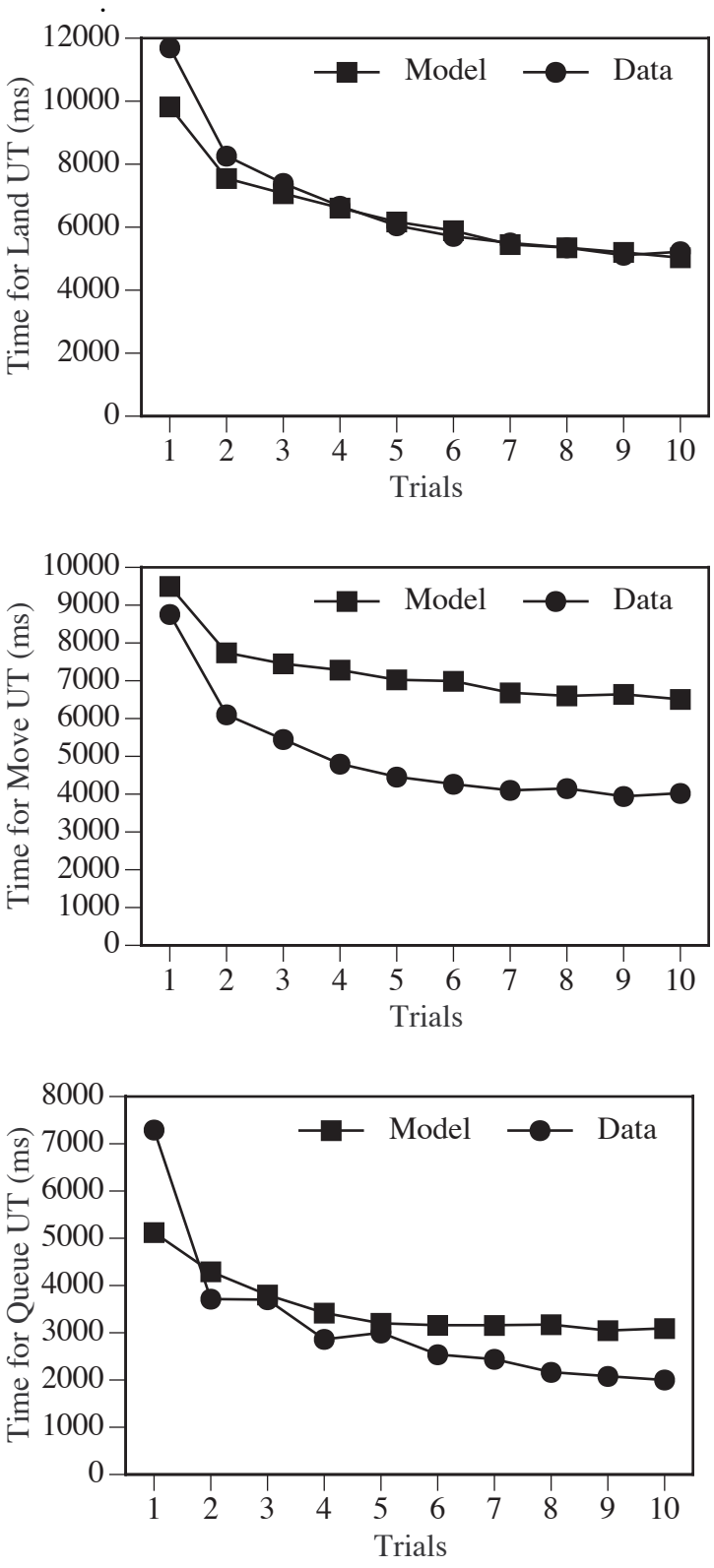
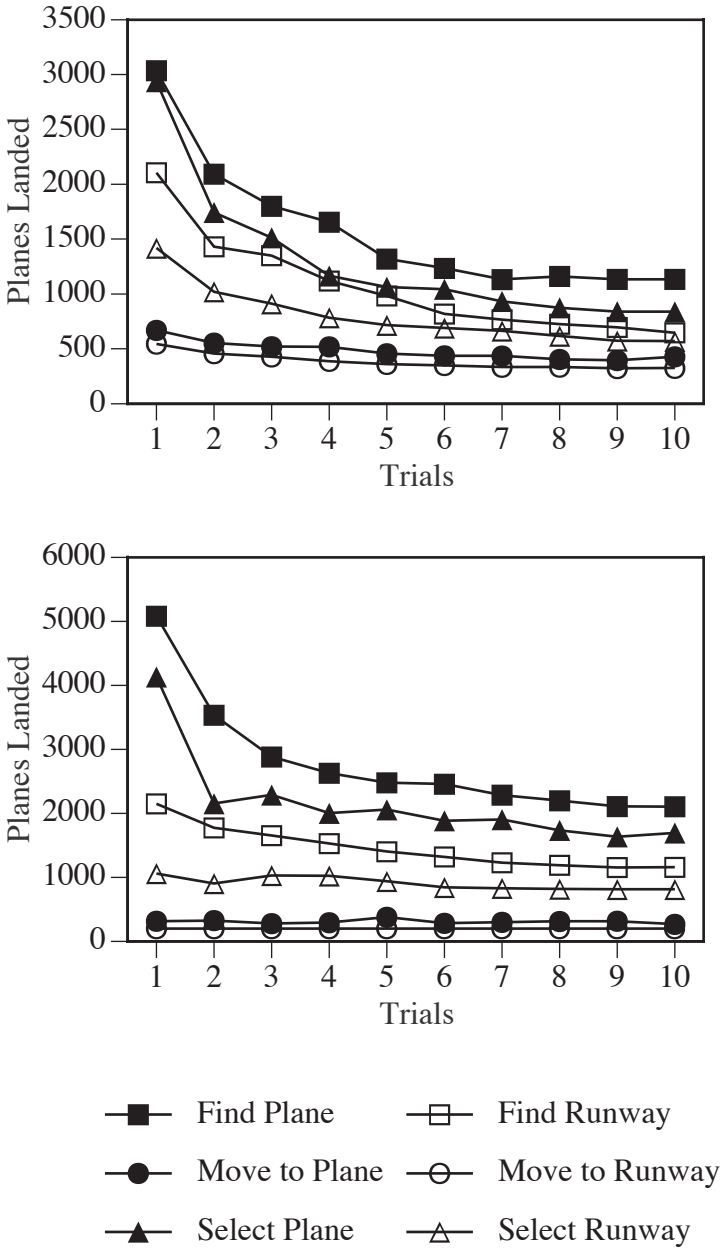

Time for Land UT (ms)
12000
10000
8000
6000
4000
2000
0
Model
Data
1 2 3 4 5 6 7 8 9 10
Trials
Time for Move UT (ms)
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
Model
Data
1 2 3 4 5 6 7 8 9 10
Trials
Time for Queue UT (ms)
8000
7000
6000
5000
4000
3000
2000
1000
0
Model
Data
1 2 3 4 5 6 7 8 9 10
Trials

Figure 6.


3500
3000
2500
2000
1500
1000
500
0
Planes Landed
1 2 3 4 5 6 7 8 9 10
Trials
6000
5000
4000
3000
2000
1000
0
Planes Landed
1 2 3 4 5 6 7 8 9 10
Trials
Find Plane
Find Runway
Move to Plane
Move to Runway
Select Plane
Select Runway

NIELS A. TAATGEN, University of Groningen, The Netherlands, and FRANK J. LEE, Rensselaer Polytechnic Institute, USA

NIELS A. TAATGEN received a Masters degree in Computer Science in 1989 and a Masters degree in Psychology in 1991 from the University of Groningen, Netherlands. After that, the helped setting up the Cognitive Science (now Artificial Intelligence) department at the University of Groningen, while doing his Ph.D. research. He received his Ph.D. in psychology in 1999. Since then, he has held a faculty position in the department of Artificial Intelligence at the University of Groningen, where he is currently a University Teacher (roughly equivalent to an assistant professor with tenure). His research interests are cognitive modeling, skill acquisition, language acquisition, procedural learning, problem solving and cognitive development.

FRANK J. LEE received an A.B. degree in Cognitive Science from University of California, Berkeley in 1994. He received a Ph.D. in Cognitive Psychology from Carnegie Mellon University in 2000. Since 2001, he has held a faculty appointment in the Department of Cognitive Science at Rensselaer Polytechnic Institute. He is currently an assistant professor. His research interests include human-computer interaction and development of synthetic agents.