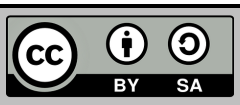
Cognitive Tutors: The End of Boredom and Confusion?

Term Paper

Lukas Gebhard
lukas.gebhard@students.uni-freiburg.de

Submitted on 01/08/2018

Albert-Ludwigs-Universität Freiburg
Department of Computer Science



Abstract

In the field of e-learning, many approaches have been tried to create adaptive digital learning environments. Based on interactions with the student, these systems make inferences about individual pre-knowledge, aptitude, affect and/or progress, and dynamically adapt the student's instruction accordingly. Cognitive tutors are an example of such systems. Based on a model of human cognition, a cognitive tutor tracks the process of knowledge acquisition and can thus select appropriate teaching contents. Keeping track of the student's cognitive states also allows it to identify common mistakes and to guide the student through exercises by giving contextual hints and feedback. In this article, I outline the general concept of cognitive tutors. In particular, I focus on cognitive tutors based on the prominent ACT-R theory to demonstrate how a cognitive tutor can be implemented. Then I move on to analyse the potential of cognitive tutors. I point out that, while they are good at adapting to individual students, they have some remarkable shortcomings. Most notably, their use is limited to teaching how to solve well-defined and well-structured problems. Moreover, they are highly domain-specific and their development is expensive. As a conclusion, a cognitive tutor's benefit largely depends on the actual use case.

Contents

1 Introduction

For practical reasons, teachers in traditional classroom settings cannot adapt teaching to each student individually. Of course, they can vary teaching methods, provide exercises of various difficulty levels, and at times even consider some students individually. However, as they have to take care of many things in a lesson (e.g., classroom management, presenting new learning contents, answering questions, assessing students, giving instructions), most of the time they need to generalise instruction to some extent. This leads to undesirable situations. For example, some students may be bored because they are already familiar with the presented topics, others may not understand what is going on due to lacking pre-knowledge, and still others may be overstrained since they require a slower pace of learning.

Intelligent tutoring systems (ITS) are a promising approach to tackle this problem. These digital learning environments adapt to each student individually by accounting for their pre-knowledge, aptitude and/or emotions and thus allowing them to learn at their own pace. Today, there is a wide range of ITSs, focussing on different application areas, based on various theories, and implemented using all kinds of techniques.

A prominent type of ITS is known as the *cognitive tutor*. What makes cognitive tutors particularly interesting is that they model cognition, which seems reasonable in the context of knowledge acquisition. Their development dates back to the mid 80s [5] when John R. Anderson, a researcher at the U.S. university Carnegie Mellon, evaluated his cognitive theory *ACT* [2]. As Anderson and his colleagues extended and improved ACT, they also continued to develop cognitive tutors based on that theory. By 2003, their cognitive tutors were brought to more than 1400 schools [20], and today they are still being improved and distributed by Carnegie Learning Inc. [8], an offspin company of Carnegie Mellon [20].

In this article we will have a closer look at cognitive tutors. More specifically, the main objectives are:

1. To outline the general concept of cognitive tutors,
2. To explain how they work using the example of ACT-R-based tutors (ACT-R [3] is the current version of ACT), and
3. To analyse their potential, especially with respect to their ability to adapt to students individually.

To that end, I first cover theoretical foundations in chapter 2. In chapter 3 I then outline what cognitive tutors are and how they work. Finally, I discuss strengths and limitations of cognitive tutors in chapter 4 and draw some conclusions in chapter 5.

2 Theoretical Foundations

Before actually turning towards cognitive tutors, in this chapter I first introduce some theoretical basics needed to understand the following chapters.

2.1 Types of Knowledge

As any other learning environment, cognitive tutors are built to facilitate the acquisition of knowledge. Since knowledge is a broad and unspecific term, many attempts have been made to provide a classification of it (e.g., [16] and [1]). In the field of educational psychology, it is common to distinguish three types of knowledge [1]: *declarative knowledge*, *procedural knowledge* and *conditional knowledge*. The terms are explained in Table 2.1.

Knowledge type	Description	Example
Declarative	The mental representation of factual information ("knowing that").	Knowing that Paris is the capital of France.
Procedural	The mental representation of skills ("knowing how").	Knowing how to solve a system of linear equations.
Conditional	The understanding of when and where the application of declarative or procedural knowledge is appropriate ("knowing when and where")	Knowing when a citation is required when writing a research article.

Table 2.1: A common classification of knowledge (based on Alexander et al. [1]).

In addition to the above three terms, *metacognitive knowledge* - knowledge about knowledge - is of key importance with respect to education. This is because successful learning is based on efficient learning strategies, and these strategies themselves must be learned as well. Metacognitive knowledge comprises all three of the above knowledge types [15]. For example, for learning to be successful it is useful to know that prior knowledge influences reading comprehension (declarative metacognitive knowledge). Moreover, it can be beneficial to know how to summarise a text (procedural metacognitive knowledge). In addition, it is important to know when it is actually a good idea to summarise a text to better understand it (conditional metacognitive knowledge).

2.2 ACT-R

ACT-R is a comprehensive model of human cognition. As such it can serve as the underlying theory of cognitive tutors. In section 3.3, I will illustrate the concept of cognitive tutors at the example of ACT-R-based cognitive tutors. For that reason, in this section I give a high-level overview of ACT-R, mostly based on Anderson et al. [4].

Originally, ACT-R consisted of "a theory of the nature of human knowledge, a theory of how this knowledge is deployed, and a theory of how that knowledge is acquired" [3]. Meanwhile, it "has evolved into a theory that consists of multiple modules but also explains how these modules are integrated to produce coherent cognition" [4]. Each of these modules is concerned with a certain aspect of cognition, such as the memory, the processing of visual information, the management of intentions, and the controlling of movements, to name only a few.

One of the most fundamental assumptions of ACT-R is that knowledge can be divided into declarative and procedural knowledge. In contrast to the threefold classification of knowledge outlined in section 2.1, ACT-R does not explicitly consider conditional knowledge. This is not contradictory, though, as ACT-R resides on a lower level of abstraction. That is, ACT-R models how information are processed, stored and retrieved in/from the brain, while the threefold classification rather emphasises high-level characteristics of each of the types of knowledge.

Furthermore, ACT-R assumes a centralised organisation of the modules. More precisely, each module exchanges information with a central component called the *production system*, but does not directly exchange information with other modules.[1]

Figure 2.1 visualises the interplay of core components in version 5.0 of ACT-R. Each component is believed to be associated with a certain brain area which is specified in parentheses. The following list provides high-level descriptions of the components shown in the figure.

Buffers To communicate with the production system, each module has an interface called the *buffer*. "[T]he content of any buffer is limited to a single declarative unit of knowledge, called a *chunk*. Thus, only a single memory can be retrieved at a time or only a single object can be encoded from the visual field." [4]

The declarative module This module implements a structured storage of declarative knowledge. Each chunk is associated with a level of *activation*. The declarative buffer always contains the most highly activated chunk. The activation of a chunk in turn depends on the relevance to the current context and its usefulness in

[1] Anderson et al. [4] admit that this clearly over-simplifies the structure of the brain as brain areas are directly connected with each other.

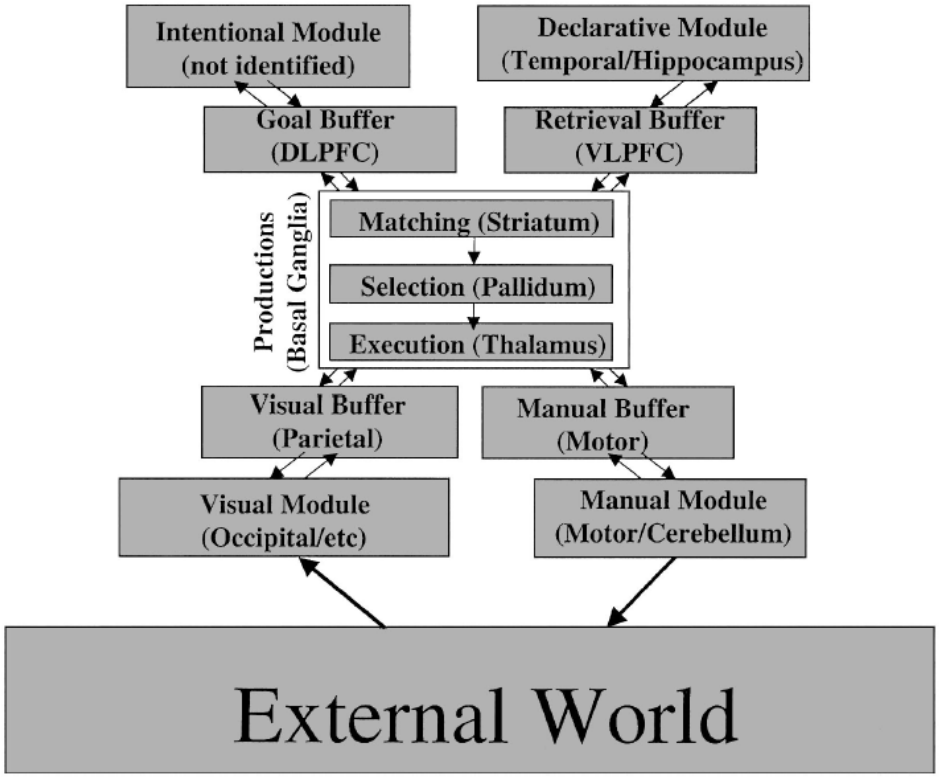


Figure 2.1: The interplay of core components in ACT–R 5.0. Image source: Anderson et al. [4]

the past. On a more abstract level, the declarative module "takes the form of a semantic net linking propositions, images, and sequences by associations" [12]. Thus, in ACT-R declarative knowledge not only comprises single facts but also an understanding of complex relationships and concepts.

The visual module The task of this module is to visually identify objects.

The manual module This module controls the hands.

The intentional module This module keeps track of goals and intentions. With that, it is also required for problem solving. The process of problem solving can be seen as the maintenance of a list of goals which have to be reached. While trying to reach one of the goals, one may identify a sub-goal which is then appended to the list. On reaching a goal, it is removed. Eventually, the overall problem is solved as soon as the list is empty.

The production system As noted earlier, the production system is ACT-R's central component. By coordinating all of the modules, it achieves coherent behaviour and thus implements procedural knowledge. This works as follows. At the core lies a set of so-called *production rules* which are basically condition-action pairs.

A condition specifies a pattern of the buffer contents whereas an action represents a modification of these contents. If a matching pattern is detected in the buffers, the corresponding action is selected as a candidate for execution. However, in each cycle only one action is executed.[2] To be able to take the most appropriate action, each production rule is associated with a continuously varying value (similar to the activation levels of chunks). These values are called *utilities* and learned from experience. The production system always selects the production rule that has the highest utility and executes the corresponding action. Thereby, it modifies the buffer contents, and with that influences the processing that occurs in the modules. Together with the continuous adjustment of utilities, it thus achieves coherent behaviour, that is, procedural knowledge.

With regard to instructional design, it is important to note how ACT-R explains the acquisition of skills. First, the learner takes the instruction for the task and converts it into declarative knowledge. Then, they convert this knowledge into production rules, thus creating procedural knowledge. After that, they do not need the declarative representation any more but can instead carry out the task based on the production rules, which leads to better performance.

[2] ACT-R models time as a sequence of cycles. In each cycle, the modules concurrently update the contents of their buffers and then the production system executes an action correspondingly.

3 Concept

Now that we have covered the basics, in this chapter we turn towards the concept of cognitive tutors. Concretely, after defining what cognitive tutors actually are (section 3.1), we have a look at their general architecture (section 3.2). Finally, in section 3.3 I outline how cognitive tutors work using the example of ACT-R-based tutors.

3.1 Definition

Cognitive tutors are "problem solving environments constructed around cognitive models of the knowledge students are acquiring" [11]. According to this definition, a cognitive tutor is a special type of the more general concept of an *intelligent tutoring system (ITS)*. The latter simply refers to "any computer program that contains some intelligence and can be used in learning" [14].

The term 'cognitive tutor' was coined by researchers of the Advanced Computer Tutoring Project at the U.S. university Carnegie Mellon in the late 80s and early 90s [5]. Their main focus of work was to design instruction "with reference to a cognitive model of the competence that the student is being asked to learn".

As a side note, *Cognitive Tutor*® is a registered trademark of the U.S. company Carnegie Learning, Inc. [7].

3.2 Architecture

As noted in section 3.1, a cognitive tutor is a special type of an ITS. Therefore, it is not surprising that its high-level architecture is mostly identical to the architecture of an ITS, which traditionally entails four components [14][18]:

- The *domain model* encodes the knowledge which is to be learned. There are various possibilities to encode that knowledge. For instance, it can be represented as a curriculum, an ontology, a set of constraints, or a set of rules [18].
- The *student model* reflects "the student's cognitive and affective states and their evolution as the learning process advances" [18].

- The *tutoring model* chooses tutoring strategies and actions based on input from both the domain model and the student model.
- The *user interface* interacts with the student and incorporates the actual learning environment.

Figure 3.1 sketches the high-level structure of a typical ITS. The domain, student, and tutoring model exchange information among each other. Together, they make up the programme logic and provide the data for the user interface to display.

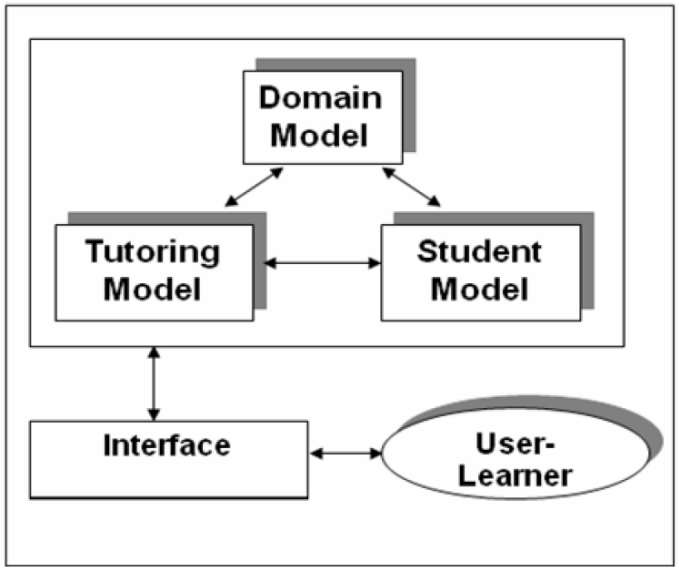


Figure 3.1: The traditional architecture of an ITS. Image source: Nkambou et al. [18]

What makes a cognitive tutor a special type of an ITS is how these components and their interactions are implemented. This is demonstrated in the following section.

3.3 Cognitive Tutors based on ACT-R

A cognitive tutor is a special type of an ITS in the sense that its domain model is based on some kind of a cognitive model. In this section, I outline the concept of cognitive tutors with respect to the cognitive model presented by Corbett et al. [9][10], which builds upon ACT-R (see section 2.2).

3.3.1 The Domain Model

As mentioned in section 2.2, ACT-R models problem solving as the maintenance of a hierarchy of goals which have to be reached one by one to eventually solve the problem.

A goal can be reached by effectively applying production rules. Correspondingly, the domain model of a cognitive tutor is made up by a set of production rules, representing "a complete, executable model of procedural knowledge in the domain" [9].

As an example, we consider the (very simple) problem of solving a linear equation of the form $ax = b$ with constants $a, b > 0$ and variable $x > 0$, $a, x, b \in \mathbb{R}$. The following pseudo-code production rules could represent the domain model:

(1)
```
IF a != 1 THEN
   adopt goal isolate_x;
```

(2)
```
IF goal == isolate_x THEN
   divide equation by a;
   drop goal isolate_x;
   adopt goal get_result;
```

(3)
```
IF goal == get_result THEN
   set result to right side of equation;
   drop goal get_result;
```

In practice, domain experts work together with cognitive psychologists to derive such a set of production rules "from a cognitive task analysis of the domain knowledge and reasoning strategies students are learning and applying in problem solving" [10].

Apart from being a source of expert knowledge, the domain model should be able to "solve the problems posed to students in the many ways that students solve them" [10]. To that end, it even includes so-called *buggy rules* [10] which support the tutoring model in detecting common mistakes. For instance, we could extend the above example by the following buggy rule:

(4)
```
IF goal == isolate_x THEN
   multiply equation by a;
   drop goal isolate_x;
   adopt goal get_result;
```

If the student applies this production rule, the domain model notifies the tutoring model which may then tell the user interface to display an error-specific hint such as 'to isolate the variable, you have to divide the equation by a, not to multiply it'.

3.3.2 The Student Model

To track the student's progress, the student model makes use of a technique known as *bayesian knowledge tracing* [9]. Basically, the rationale is to create an overlay of the production rules to be learned, where each rule is associated with the estimated probability that the student has already learned it. Each time the student applies a production

rule, the corresponding probability is updated depending on whether they applied it correctly or not. Based on these probabilities, the tutoring model can dynamically choose appropriate exercises.

Bayesian knowledge tracing is a specialisation of the concept of *hidden Markov models* [23]. The model makes the following simplifying assumptions:

- Each production rule is either in the learned or in the unlearned state. That is, it is either contained or not contained in the student's procedural knowledge.
- Production rules cannot make transitions from the learned to the unlearned state. In other words, procedural knowledge is never forgotten.
- Each production rule r is parametrised as explained in Table 3.1.

Name	Term	Description
A-priori knowledge	$p(L_0)_r$	The probability that rule r is in the learned state prior to the first opportunity to apply it.
Acquisition	$p(T)_r$	The probability that rule r transits from the unlearned to the learned state at an opportunity to apply it.
Slip	$p(S)_r$	The probability to make a mistake when applying rule r from the learned state.
Guess	$p(G)_r$	The probability of correctly applying rule r if it is in the unlearned state.

Table 3.1: The parameters in bayesian knowledge tracing. For each production rule r, these four parameters have to be set individually. Source: Adaptation of Corbett et al. [9]

Based on these values, the probability $p(L_n)_r^s$ that rule r is in the learned state of s after n opportunities to apply it is recursively given as

$$
\begin{aligned}
p(L_0)_r^s &= p(L_0)_r \\
p(L_n)_r^s &= p(L_{n-1}|E_n)_r^s + p(\neg L_{n-1}|E_n)_r^s \, p(T)_r
\end{aligned}
$$

where E_n is a binary random variable representing the n-th evidence, that is, if s correctly applied r in step n ($E_n = \text{correct}$) or not ($E_n = \text{wrong}$). Consequently, $p(L_{n-1}|E_n)_r^s$ is the posterior probability that r had already been in the learned state right before s had the n-th opportunity to apply it. We can compute this posterior via Bayesian inference.

Concretely, if s correctly applies r at the n-th opportunity, we have

$$p(L_{n-1}|E_n = \text{correct})_r^s = \frac{p(L_{n-1} \wedge E_n = \text{correct})_r^s}{p(E_n = \text{correct})_r^s} = \frac{p(L_{n-1})_r^s \, p(\neg S)_r}{p(L_{n-1})_r^s \, p(\neg S)_r + p(\neg L_{n-1})_r^s \, p(G)_r}.$$

Similarly, if s fails to correctly apply r at step n, we have

$$p(L_{n-1}|E_n = \text{wrong})_r^s = \frac{p(L_{n-1} \wedge E_n = \text{wrong})_r^s}{p(E_n = \text{wrong})_r^s} = \frac{p(L_{n-1})_r^s \, p(S)_r}{p(L_{n-1})_r^s \, p(S)_r + p(\neg L_{n-1})_r^s \, p(\neg G)_r}.$$

As a side note, instead of fitting the four parameters shown in Table 3.1 per skill only, they can also be fit per skill and per student. This yields a more accurate model which explicitly accounts for differences between students [23] [13].

3.3.3 The Tutoring Model

Without the tutoring model, the other two models are not of much use. In a process called *model tracing*, the tutoring model leverages the domain model and the student model to create an adaptive learning environment. This adaptivity is the result of several strategies:

- The tutoring model guides the student through the solution space. More precisely, it keeps the student "within a specified tolerance of an acceptable solution path" [14]. For example, a tutoring model with minimum error tolerance would reject any non-applicable action and have the user interface inform the student about their mistake immediately. The student can then correct their mistake.
- It offers situation-specific guidance. For instance, if the student requests a hint, the tutoring model could identify a production rule applicable in the given situation and suggest to perform the corresponding action.
- It takes the student's progress into account. In particular, it can select exercises covering production rules that the student model considers unlikely to be in the learned state.

3.3.4 The User Interface

The three components described above all work behind the scenes. The actual interaction with the user happens through a separate component, the user interface. It has several functions:

- It explains the domain and how to solve problems in the domain. For instance, it may provide learning material in the form of hypertext, show step-by-step solutions for sample problems, or visualise the domain knowledge using figures or animations.
- It receives the student's answers to problems at the granularity of production rules. Thus, the tutoring component can evaluate each solution step separately.
- It provides support on demand. For example, on request it may give a context-specific hint to guide the student into the right direction.
- If the student deviates too much from an acceptable solution path, it rejects their answer.
- It makes the learning progress transparent. This does not only support self-assessment but also adds an element of gamification to the system. For instance, the APT LispTutor, a cognitive tutor teaching the programming language Lisp, shows a progress bar for each skill to be learned [11] (see Figure 3.2 for a screen-shot).

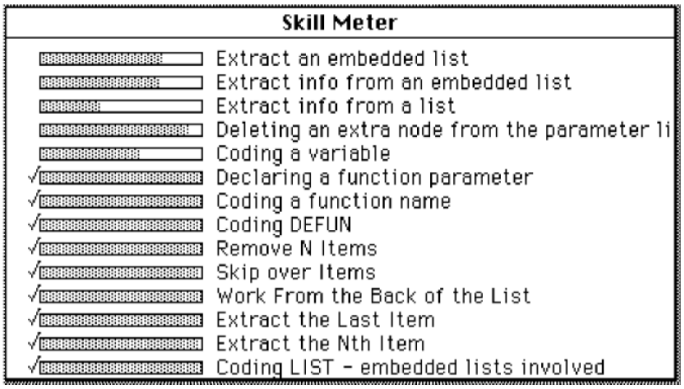


Figure 3.2: The 'skill meter' in the APT LispTutor interface. Image source: Corbett et al. [11]

4 Discussion

After having seen what cognitive tutors are and how they work, in this chapter I highlight some of their strengths and limitations.

4.1 Empirical Results

To the best of my knowledge, so far cognitive tutors have only been evaluated within the framework of blended learning curricula. Most notably, The U.S. Department of Education conducted a meta-study [22] on the effectiveness of high school Maths curricula involving a cognitive tutor developed by Carnegie Learning Inc., as compared to traditional curricula. Altogether, the authors of the meta-study reviewed a total of 27 studies but only considered 6 of them to meet their institution's evidence standards. These 6 studies include a total of about 2,500 students from 39 schools. The empirical results suggest that it is unclear if Carnegie Learning's curricula improve student performance. More specifically, one of the evaluated studies showed a statistically significant negative effect, another one showed a statistically significant positive effect, while the remaining four studies were found to show indeterminate effects.

However, cognitive tutors are not restricted to blended learning environments. They can also be used for autonomous learning, or in the context of standalone online courses. In particular, they may prove beneficial in countries with a severe shortage of qualified teachers, just as any ITS.[1] From this viewpoint, what counts is if cognitive tutors are effective in the absence of human instruction. This has yet to be investigated in detail.

4.2 Range of Application

By definition, cognitive tutors are problem solving environments. Typical application areas are well-defined and highly structured domains of subjects such as programming [11], genetics [10], and geometry [5].

However, there are numerous cases where problem solving is difficult to formalise or cannot be formalised at all. These include problems that are not solved yet, such as 'how to tackle climate change', as well as tasks with high degrees of freedom, such as 'develop a programme that can play chess'. Since such problems require creativity and can often

[1] See Nye [19] for a review of trends and approaches for educational technology in a global context.

be solved in infinitely many ways, it is not feasible to develop domain models for them. Beginners are likely to be overstrained by unstructured problems, but advanced students should be given the opportunity to practice solving such problems. Consequently, cognitive tutors might be a good choice for students with low to intermediate knowledge in the domain but not for advanced ones.

Apart from that, as cognitive tutors are restricted to problem solving (and thus focus on the acquisition of procedural and conditional knowledge), they are not suitable for the teaching of declarative knowledge. To the best of my knowledge, so far no-one has made an attempt to explicitly implement a cognitive model for the learning of facts and their complex relationships. As a consequence, today it does not make much sense to use cognitive tutors for teaching topics such as photosynthesis, the Vietnam War, capitalism, or permaculture.

4.3 Development Costs

Converting domain knowledge into a domain model (e.g., a set of production rules) is a time-consuming task [6], even though there are software tools that facilitate this work [17]. In addition, the resulting domain model is highly specific for the domain it was created for and cannot be transferred to another domain. For example, usually a domain model built for teaching a certain programming language cannot be used for teaching another one.

On the other hand, just as any other ITS, a cognitive tutor has the potential to save a tremendous amount of time once it is implemented. It can take over lots of tasks a human teacher would otherwise have to carry out. Therefore, depending on the scenario, high development costs may be negligible as compared to the added value. Furthermore, data mining seems to be a promising approach to further reduce the amount of expert knowledge needed to create a domain model [6].

4.4 Discrepancies with Constructivism

On a more fundamental level, the four-component architecture of ITSs (see section 3.2) can be criticised for not being compatible with the theory of constructivism [21]. Constructivists argue that learning is an active process involving individual interpretation of information and integration ot that information into already existing knowledge. To some degree this contradicts the typical architecture of ITSs. This is because they evaluate learning progress based on a predefined set of knowledge, the domain model. This implies that after successful learning each student should have internalised a copy of this domain model as part of their procedural knowledge. However, according to constructivism, each student would rather construct their own knowledge representation, as a result of individual interpretation and integration into pre-knowledge.

4.5 Adaptivity of Instruction

Finally, we return to the article's central question: Are cognitive tutors the end of boredom and confusion? Or, more precisely, to what extent are cognitive tutors adaptive to the student? The answer depends on the tutor's actual design. As an example, I assess the adaptivity of ACT-R-based tutors (see section 3.3) here.

Most notably, ACT-R-based tutors enable students to learn at their own pace. By applying bayesian knowledge tracing (see subsection 3.3.2, they are able to choose exercises based on individual learning progress. In addition, they can account for pre-knowledge and intelligence (especially if the model parameters are fit per student), for example by adjusting the amount of exercises a students needs to work through. Furthermore, through model tracing (see subsection 3.3.1) ACT-R-based tutors provide individual guidance. They offer problem-specific feedback and hints and prevent students from drifting away from correct solutions.

However, ACT-R-based tutors also have some shortcomings with respect to adaptivity. For one, while they are able to track the student's learning progress, they cannot identify their current affect towards the task. This could be beneficial to increase learning motivation, though, for example by giving empathic feedback. For another, ACT-R-based tutors lack adaptivity with respect to the degrees of freedom associated with exercises. This is simply because they cannot be used to teach how to solve problems with high degrees of freedom, as argued in section 4.2.

5 Conclusions

In this article, I have outlined the concept of cognitive tutors and evaluated their potential with the example of tutors based on ACT-R. To wrap it up, within the framework of problem solving cognitive tutors are a promising approach to provide individualised instruction without a human element. Based on findings from cognitive sciences, they are able to quantify learning progress with fine granularity, give problem-specific feedback and guide the student through the problem space. However, when considering to use or to develop a cognitive tutor, one has to bear in mind that its range of application is quite limited and its development expensive.

Where the future is concerned, it is not clear yet if cognitive tutors or another type of ITS will eventually prevail. Perhaps various approaches will be merged together to produce some kind of an universal ITS. Or maybe different types of ITSs will be used depending on the application area. Whatever the case, I hope that further research in the field of e-learning promotes and facilitates the access to high-quality education worldwide.

Bibliography

[1] P. A. Alexander, D. L. Schallert, and V. C. Hare. "Coming to Terms: How Researchers in Learning and Literacy Talk about Knowledge". In: *Review of Educational Research* 61.3 (1991), pp. 315–343.

[2] J. R. Anderson. *The Architecture of Cognition*. Harvard University Press, 1983.

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

[4] J. R. Anderson et al. "An Integrated Theory of the Mind". In: *Psychological Review* 111.4 (2004), pp. 1036–1060.

[5] J. R. Anderson et al. "Cognitive Tutors: Lessons Learned." In: *The Journal of Learning Sciences* 4 (1995), pp. 167–207.

[6] T. Barnes and J. Stamper. "Toward the Extraction of Production Rules for Solving Logic Proofs". In: *International Conference of Artificial Intelligence in Education*. 2007, 11–20.

[7] Carnegie Learning, Inc. *Copyright Information*. URL: `https://www.carnegielearning.com/copyright-information/` (visited on 2018-06-03).

[8] Carnegie Learning, Inc. *Product Overview*. URL: `https://www.carnegielearning.com/products/our-products/overview/` (visited on 2018-06-12).

[9] A. T. Corbett and J. R. Anderson. "Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge". In: *User Modeling and User-Adapted Interaction* (1994), pp. 253–278.

[10] A. T. Corbett et al. "A Cognitive Tutor for Genetics Problem Solving: Learning Gains and Student Modeling". In: *Journal of Educational Computing Research* 42.2 (2010), pp. 219–239.

[11] A. Corbett, M. McLaughlin, and K. C. Scarpinatto. "Modeling Student Knowledge: Cognitive Tutors in High School and College". In: *User Modeling and User-Adapted Interaction*. Kluwer Academic Publishers, 2000, pp. 81–108.

[12] R. Culatta and G. Kearsley. *ACT-R (John Anderson)*. URL: `http://www.instructionaldesign.org/theories/act/` (visited on 2018-06-03).

[13] M. Eagle et al. "Exploring Learner Model Differences Between Students". In: *Lecture Notes in Computer Science*. 2017.

[14] R. Freedman. "What is an Intelligent Tutoring System?" In: *Intelligence* 11.3 (2000).

[15] J. E. Jacobs and S. G. Paris. "Children's Metacognition about Reading: Issues in Definition, Measurement, and Instruction". In: *Educational Psychologist* 22.3-4 (1987), pp. 255–278.

[16] T. de Jong and M. G. Ferguson-Hessler. "Types & Qualities of Knowledge". In: *Educational Psychologist* 31.2 (1996), pp. 105–113.

[17] T. Murray. "Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art". In: *International Journal of Artificial Intelligence in Education* 10 (1999), pp. 98–129.

[18] R. Nkambou, R. Mizoguchi, and J. Bourdeau. *Advances in Intelligent Tutoring Systems*. Springer, 2010.

[19] B. D. Nye. "Intelligent Tutoring Systems by and for the Developing World: A Review of Trends and Approaches for Educational Technology in a Global Context". In: *International Journal of Artificial Intelligence in Education* 25 (2015), pp. 177–203.

[20] Pittsburgh Advanced Cognitive Tutor Center. *PACT Center @ Carnegie Mellon University*. URL: `http://pact.cs.cmu.edu/` (visited on 2018-06-12).

[21] J. Self. "Theoretical Foundations for Intelligent Tutoring Systems". In: *Journal of Artificial Intelligence in Education* 1.4 (1990), pp. 3–14.

[22] U.S. Department of Education, Institute of Education Sciences, What Works Clearinghouse. *High School Mathematics Intervention Report: Carnegie Learning Curricula and Cognitive Tutor®*. 2013.

[23] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. "Individualized Bayesian Knowledge Tracing Models". In: *Artificial Intelligence in Education*. Ed. by H. C. Lane et al. Springer Berlin Heidelberg, 2013, 171–180.