Aleven, V., & Sewall, J. (2016). The frequency of tutor behaviors: A case study. In A. Micarelli, J. Stamper, & K. Panourgia (Eds.), P*roceedings of the 13th International Conference on Intelligent Tutoring Systems, ITS 2016* (pp. 396-401). Springer International Publishing. doi:10.1007/978-3-319-39583-8_47

The Frequency of Tutor Behaviors: A Case Study

Vincent Aleven and Jonathan Sewall

Human-Computer Interaction Institute
Carnegie Mellon University

aleven@cs.cmu.edu, sewall@cs.cmu.edu

Abstract: For cross-pollination between ITS authoring tools, it may be useful to know the prevalence of tutoring behaviors crafted with these tools. As a case study, we analyze the problem units of *Mathtutor,* a web-based intelligent tutor for middle-school mathematics built, as an example-tracing tutor, with the Cognitive Tutor Authoring Tools (CTAT). We focus on tutoring behaviors that are relevant to a wide range of tutoring systems, not just example-tracing tutors, including behaviors not found in VanLehn's (2006) taxonomy of tutor behaviors. Our analysis reveals that several tutor behaviors not typically highlighted in the ITS literature were used extensively, sometimes in unanticipated ways. Others were less prevalent than expected. This novel insight into the prevalence of tutor behaviors may provide practical guidance to ITS authoring tool developers. At a theoretical level, it extends VanLehn's taxonomy of tutor behavior, potentially expanding how the field conceptualizes ITS behavior.

Keywords: ITS authoring tools; behavior of tutoring systems; authoring data

1 Introduction

Versatile, robust, easy-to-use, and easy-to-learn tools for authoring ITSs are an important development [1][2][3] and may be key to making ITS widespread. In developing an ITS authoring tool, a key question is: What tutor behaviors should the tool support? VanLehn's classic taxonomy of tutor behaviors [4] provides one possible answer. This taxonomy was induced by theoretically analyzing six ITSs. On the other hand, ITS authoring tools may provide a unique practical perspective that may not be fully captured in this taxonomy. This may be so especially if the tool has had a long life and seen widespread use; it may gradually have acquired features aimed at supporting a wide range of tutoring behaviors. If many tutors or tutor units have been built with the tool, we can measure the frequency of key tutor behaviors in these tutors. We present a case study, focusing on the Cognitive Tutor Authoring Tools (CTAT) [1], which support an ITS technology called example-tracing tutors. Over the years, many tutors have been built with CTAT and these tools have been honed and extended based on the needs of these projects. It is thus an interesting question which tutor behaviors are prevalent in CTAT-built tutors. We focus on one such tutor, *Mathtutor*, [5], one of a number of web-based ITS for middle-school mathematics (cf. *AS-*

SISTments [3] and *Wayang Outpost* [6]). A distinguishing characteristic may be that *Mathtutor* supports more complex problem-solving scenarios.

Our investigation focuses on a set of tutor behaviors commonly found in many ITSs and not specific to example-tracing tutors. It includes some behaviors not found in VanLehn's taxonomy [4]. Some of these behaviors were described in our prior publications [1], but we have not previously undertaken a systematic analysis of their use or frequency, nor are we aware of any other projects reported in the ITS literature that did so. Baker et al. created a taxonomy of tutor features to investigate students' gaming behaviors [7], but this taxonomy was too fine-grained for current purposes, nor did it focus on tutor behavior exclusively.

The work contributes to the ITS literature both at a practical and theoretical level. At a practical level, insight into the prevalence of tutoring behaviors may provide guidance for developers of ITS authoring tools. At a theoretical level, our analysis enriches theoretical accounts of tutor behavior by extending VanLehn's (2006) taxonomy of ITS behaviors.

2 Overview of *Mathtutor* and CTAT

Mathtutor [5] covers five content strands for mathematics in grades 6 through 8: (1) numbers and operations, (2) algebra, (3) data analysis, (4) geometry and (5) ratios and proportional reasoning. It is a re-implementation, as an example-tracing tutor, of a set of Cognitive Tutors for middle-school mathematics created prior to CTAT's inception. *Mathtutor* offers 65 units, each comprising between 8 and 30 problems for students to solve. So far, *Mathtutor* has been used by 2,215 students, who completed a total of 31,918 problems in 1,258 hours of work. *Mathtutor* was built by a team of authors that included professional staff, many student interns, and teachers. A goal was to reproduce the tutor behaviors of the original Cognitive Tutors, adhering to a model of tutoring that is encoded in eight Cognitive Tutor principles [8]. This model prescribes making thinking visible by breaking problems into steps and providing step-level guidance such as next-step hints and feedback.

Example-tracing tutors can be built with CTAT through a combination of end-user programming techniques such as drag-and-drop interface building, programming by demonstration, Excel-like formula writing, and template-based problem generation [1]. An author first decides for which problems to provide tutoring and conducts cognitive task analysis to identify solution steps, common major and minor strategy variations, and common errors (although given that *Mathtutor* is a reimplementation of existing tutor units, this information was instead gleaned from the existing units). She then creates a user interface for each of the targeted problem types, which lays out the steps of the problems. Using CTAT's Behavior Recorder, the author creates a "behavior graph" that defines acceptable solution strategies. An author can generalize a behavior graph in a number of ways, so that it can stand for a wider range of problem-solving behavior than literally just what is recorded in the graph. The author also writes hints and feedback messages. At student run time, the tutor uses the graph to interpret student problem-solving behavior and to provide hints and feedback.

3 Analysis of tutoring behaviors supported in *Mathtutor*

Our analysis focuses on the following inner-loop (i.e., within-problem, step-level) tutor behaviors: error-specific feedback, multiple solution paths; dynamic interfaces; accepting complex input, notational variants, and minor step dependencies; input substitution; partial ordering of steps; and optional and repeatable steps. Of these, only error-specific feedback is included in VanLehn's taxonomy [4]. We analyzed the 897 behavior graphs that make up the 65 *Mathtutor* units. We ran awk scripts over the behavior graphs, generating a table with information about 33,950 behavior graph links. We then used Excel PivotTables to compute the statistics reported below.

3.1 Error feedback messages

First, we investigated the prevalence of error-specific feedback messages. These messages react to specific student errors and explain for example why the error is an error. We found that error-specific feedback messages are present in 38 out of 65 *Mathtutor* units (58% of units). Across all tutor units, 21% of links represent errors (as opposed to correct problem-solving steps). Thus, although error-specific feedback messages are used frequently, it is clear that the *Mathtutor* authors made no attempt to systematically cover the majority of errors. If they had, there would be many more error links than correct action links. In *Mathtutor,* students can rely on on-demand hints, rather than error-specific feedback, if they do not understand how to solve a step. Nonetheless, the high prevalence of error-specific feedback suggests that ITS authoring tools should support them.

3.2 Multiple solution paths

Next, we investigated to what degree, in the *Mathtutor* units, the tutor is capable of following students with respect to multiple strategies *within a single problem* [13]. Surprisingly, the ability to support multiple strategies within a given problem is not mentioned in many theoretical accounts of intelligent tutors (e.g., [4]), possibly because it is assumed to be present. Not all ITSs however appear to support multiple strategies or solution paths within a problem, so this ability should not be taken for granted. Example-tracing tutors offer two main ways of authoring tutors that can accept multiple solution strategies within a problem. First, an author can create multiple paths in a behavior graph. For example, in a *Mathtutor* unit dealing with proportional reasoning, the tutor recognizes two major strategies, Equivalent Fractions and Cross Multiplication. These major strategy variations are captured as two separate branches in the behavior graph. Second, as discussed below, an author can use formulas, regular expressions, or numeric ranges to capture minor strategy variations,

Approximately 30% of *Mathtutor* units have behavior graphs with multiple solution paths. This percentage was lower than expected, especially when one considers that multiple paths were often used to capture notational variants rather than genuinely different strategies. It may be that from a pedagogical perspective, accommodating multiple strategies within a single problem is not always high priority or even desira-

ble. It is often difficult for students to practice a single strategy to mastery, let alone multiple. Also, even when the goal is for students to learn multiple strategies, the tutor may still need to offer single-strategy problems, to make sure all strategies are practiced. Nonetheless, we recommend that ITS and ITS authoring tools be able to accommodate multiple solution strategies [9].

3.3 Dynamically adjusting the tutor interface to the state of problem solving

Next, we consider dynamic interfaces, that is, interfaces that change at specific points in the problem-solving process. Using CTAT, authors can create dynamic interfaces without programming, by adding links in the behavior graph that capture "tutor-performed actions" (TPAs) [1]. Dynamic interfaces are used in 35% of *Mathtutor*'s units, for a variety purposes. Often they are used to manage limited screen real estate, when there is not enough space to accommodate all required interface components simultaneously. Another common use of dynamic interfaces is to reveal the steps in tutor problems gradually, as the student progresses through the problem, rather than displaying all the steps from the start, to enforce on orderly problem-solving process.

3.4 Variable steps, including dependencies among steps

A third category of behaviors comprises variable (or non-literal) steps, which an author can create by attaching formulas, regular expressions, and other matchers to behavior graph links. Formulas were used far more extensively than we anticipated, namely, in 54 out of 65 units (83% of the units). Their most common use in *Mathtutor* is to capture notational variants of student input. For instance, on steps where students enter an arithmetic expression, a formula is needed to deal with the range of equivalent expressions that students enter. In other tutor units, formulas were used to accept notational variations such as "40" and "40%." Formulas were also used to express dependencies among steps. For example, in a unit dealing with proportional reasoning, students compared two proportions (e.g., what is a better deal, buying 12 tickets for $18 or 20 of the same tickets for $25?) by first choosing a suitable "comparison number" (e.g., a number of tickets, such as 4) and then scaling the proportions to this comparison number. Formulas were used to capture the multiple options for the comparison number and also to capture how later steps depend on that number.

3.5 Input substitution

Input substitution refers to the behavior in which the tutor replaces student input by a different expression of that input, when the input is accepted as correct. A common use is to replace text typed by the student by a spelling-corrected version, or to replace an arithmetic expression by the value to which it evaluates. The latter form of input substitution makes the cell function as a simple calculator, for example in units in which the student masters arithmetic and the instructional objectives focus on other aspects of mathematics. Input substitution is used in 21 of the 65 *Mathtutor* units

(32% of units). In addition to evaluating arithmetic expression, input substitution was used for formatting student input (e.g., avoiding many decimals, making sure a percent sign is included, and money notation). The prevalence of input substitution in *Mathtutor* suggests that this functionality is important in a real-world ITS.

3.6 Partial ordering of steps

In creating a tutor, it is often desirable to constrain the order in which students carry out problem steps, although without restricting students to a single ordering of steps. In some mathematical procedures, the order of steps matters (e.g., order of operations, or processing columns right-to-left in multi-column addition). At other times, the order of steps does not matter mathematically, but it matters for creating an effective tutor, for example because it can be difficult to give good hints for a step when prior steps to which the hint refers have not been completed. In CTAT, an author can set whether overall the tutor should treat a problem as ordered or unordered. In addition, an author can define groups of links and designate them as ordered or unordered. The tutor only accepts steps that conform to the author-specified ordering constraints. Of the 65 *Mathtutor* units, the authors defined ordered or unordered groups in 40 units. Thus, it is clear that authors often want to define a partial ordering of problem steps.

3.7 Optional and repeatable steps

In tutored problem solving it is often desirable to make steps optional, meaning that they are not required for completing the problem. Similarly, it is useful to make steps repeatable, meaning that they can be, or have to be, done multiple times within a given tutor problem. In CTAT, authors can create optional and repeatable steps by specifying a lower and upper bound on the number of times a link in a behavior graph can be "traversed" as the student solves the given problem. Optional links are used in 15 units, or 23% of *Mathtutor* units, repeatable links in only 3 of the 65 units. Optional links are used primarily to provide optional scaffolding within a problem (i.e., extra steps with tutor guidance that may be helpful but not necessary for all students). Sometimes, optional links were used for actions that are mathematically correct but not strictly necessary, such as entering leading or trailing zeros for decimal numbers.

4 Conclusion

To the best of our knowledge, this paper is the first that reports on the frequency of tutor behaviors in an ITS. We focus on a set of common inner-loop behaviors including some that are not included in VanLehn's taxonomy [4] and that are rarely if ever mentioned in theoretical accounts of ITSs. A striking finding is the frequent use of formulas (over 80% of *Mathtutor* units use them) to capture input variations and (less frequently) dependencies among steps. We also found that dynamic interfaces are used frequently, that great attention is paid in *Mathtutor* to being able to accept notational variations in input and to replace student input with a different expression of it

(input substitution). On the other hand, flexibility in following students with respect to multiple problem solution paths was more rare than expected, even if it is still a highly desirable tutor behavior that ITS authoring tools should support.

The tutor behaviors discussed in this paper are not specific to example-tracing tutors; they are likely to cut across many types of tutors. At the same time, it seems likely that the reported prevalence of these behaviors is somewhat specific to mathematics at the middle-school level. Further, the particular frequencies may be somewhat specific to the tutoring paradigm used, based on Cognitive Tutor principles. It is an interesting question how much variability there is among authors in terms of what tutoring behaviors are used. We do not, however, have data to answer that question.

A limitation of the work is that it involves only a single tutoring system and only a single authoring tool, albeit a comprehensive tutoring system that has seen substantial classroom use, and an authoring tool whose range of tutoring behaviors may be wide and shaped substantially by demands from the field. It will be useful to repeat this type of analysis across many tools and tutor-building projects.

Practically, the work might provide guidance to developers of ITS authoring tools. At a theoretical level, the work elaborates the range of inner loop functionality identified by VanLehn [4], advancing our field's conceptualization of tutor behaviors.

5 References

1. Aleven V., McLaren B.M., Sewall J., van Velsen M., et al: Example-Tracing tutors: Intelligent tutor development for non-programmers. Int Jrnl of AI in Ed 26, 224-269 (2016)
2. Mitrovic A., Suraweera P., Martin B., Zakharov K., et al: Authoring constraint-based tutors in ASPIRE. In: Ikeda M., Ashley K.D., Chan T.W. (eds.) Proc of the 8th Int Conf on Intelligent Tutoring Systems, ITS 2006, pp. 41-50. Springer, Berlin (2006)
3. Razzaq L., Patvarczki J., Almeida S.F., Vartak M., et al: The Assistment Builder: Supporting the life cycle of tutoring system content creation. IEEE Transactions on Learning Technologies 2, 157-166 (2009)
4. VanLehn K.: The behavior of tutoring systems. Int Jrnl of AI in Ed 16, 227-265 (2006)
5. Aleven V., McLaren B.M., Sewall J.: Scaling up programming by demonstration for intelligent tutoring systems development: An open-access web site for middle school mathematics learning. IEEE Trans. on Learning Techn. 2, 64-78 (2009) _
6. Arroyo I., Woolf B.P., Burleson W., Muldner K., et al: A multimedia adaptive tutoring system for mathematics that addresses cognition, metacognition and affect. Int Jrnl of AI in Ed 24, 387-426 (2014)
7. Baker R.S., de Carvalho A., Raspat J., Aleven V., et al: Educational software features that encourage and discourage "gaming the system". In: Dimitrova V., Mizoguchi R., du Boulay B., Graesser A. (eds.) Proc of the 14th Int Conf on AI in Ed, pp. 475-482. IOS Press, Amsterdam (2009)
8. Koedinger K.R., Corbett A.T.: Cognitive tutors: Technology bringing learning sciences to the classroom. In: Sawyer R.K., editors. The Cambridge Handbook of the Learning Sciences, pp. 61-78. Cambridge University Press, New York (2006)
9. Waalkens M., Aleven V., Taatgen N.: Does supporting multiple student strategies lead to greater learning and motivation? Investigating a source of complexity in the architecture of intelligent tutoring systems. Computers & Education 60, 159 - 171 (2013)