Formative Assessment for User Guidance In Single Stepping Systems

Alan Krempler[1], Walther Neuper[2]

University of Applied Sciences Joanneum Graz, University of Technology Graz

Key words: *e-learning, formative assessment, single stepping system, rewriting, dialog atom, dialog guidance*

Abstract:

Single stepping systems arise from re-engineering computer algebra systems to show intermediate steps for educational purposes. The most essential of these steps concern the application of a theorem to a mathematical term transforming it into a more useful one; this is called 'rewriting' in computer mathematics.
This paper gives a systematic account of possible user-interactions within such a 'rewrite step', which we call 'dialog atoms'. More than twenty of these are identified. Appropriate choice from that range of dialog atoms and their combination is concern of a dialog guide, planned to balance the flow of user interaction between challenge and support in learning – an issue calling for close cooperation with practice of mathematics education.

1 Introduction: e-learning in mathematics

Computer mathematics strives for automating tasks of mathematicians, scientists and engineers. The most general tools for such tasks are computer algebra systems (CASs, e.g. [1,2]) and computer theorem provers (CTPs, e.g. [3,4]), and these are very successful at automation. CTPs usually require some interaction, due to the complexity of the respective tasks. CASs, however, which are being used in education, almost automatically solve equations, compute integrals etc.
The CASs' ability in automation is considered harmful for education since their introduction into class rooms twenty years ago: Why should students learn to integrate, if this is done much better by a CAS? Actually, since then a major part of conferences on didactics of mathematics dedicated most vigorous discussions to this question. Shaping this question to "how to adapt education to the existence of CAS", very few authors asked the other way round: "how to adapt CAS to education" [5,6,7]. A few producers of CASs reacted to the issue, opening up their systems to 'single stepping systems' [8,9], which show intermediate steps and allow the user to direct these steps in a limited way.
Anyway, CASs do not provide the most adequate basis for educational systems in mathematics: Students do not only want to know intermediate steps and the rules justifying the steps, they could also be interested in the proof of the rules - as theorems derived within some theory, given some definitions, axioms and other theorems. For such interests CTPs are the appropriate basis. CTPs describe their knowledge in the language of mathematics, they prove theorems mechanically in a rigorous manner and involve more and more readable proof scripts [10,11]. Thus, they come along with mechanized mathematic knowledge, which can be

read by both, computers and humans - ready to be inspected by a student. Still, CTPs are rarely used as a basis for formula-based educational systems [12,13].
Indeed, at the state of the art in computer mathematics systems can be built such that they are 'interactive and transparent models of mathematics': each process can be traced down to atoms, which are elementary operations of mathematics. Such systems can model all phases of doing mathematics: specifying, modelling and solving. The systems are powerful enough to resume ideas from the early days of 'expert systems' - to establish a 'dialog between partners on an equal base' [14]: on the one side the system 'knows' how to do the next step towards a result, and on the other side the student may watch the steps or may set a step of his choice, while the system is able to check for correctness of the step input by the user.

This paper is confined to the phase of solving. Section 2 goes into detail with 'single stepping systems', introduces the notion of 'rewriting' and other technical terms. Section 3, the main part, gives a systematic account of 'dialog atoms' and discusses issues of combining them to dialog patterns. Section 4 gives some examples for sequencing dialog atoms with respect to experiences from field tests, and Section 5 provides conclusions and future work.

2 Single stepping systems and 'rewriting'

As mentioned above, single stepping systems show intermediate steps, not only the result like traditional CASs. If a system is considered a 'model of mathematics', the steps must model basic operations of mathematics.
Given a mathematical term[1] assumed to be checked by the system and thus 'correct', a **step** applies a 'tactic' resulting in another term[2] A **term** consists of elements of certain 'types', where the latter collect specific axioms, definitions and theorems. This ensures correctness of operations: The theorem $a \cdot b = b \cdot a$ may be applied to integers, reals etc, but not to matrices. A **tactic** (in the sense of CTP) is a basic operation of mathematics: de/composition of a term, case split, substitution, application of theorems or start/completion of a sub-problem.

This paper confines tactics to the application of theorems; the respective notion in computer mathematics is 'term rewriting' [15] (short **rewriting**), employed for many relevant tasks like algebraic simplification of terms, equation solving, differentiation of functions etc. Usually rewriting applies well elaborated sets of theorems (called 'term rewriting systems' having certain properties [15]). Rewriting provides a technique for checking equivalence (via 'normal forms') of terms input by the user[3].

This paper again confines rewriting to a special case: Given a term (called **given** term), apply *one* theorem (and not a set of rules; the theorem called rewrite-rule, or short **rule**) which leads to a **result** term. The rewrite rule may be a 'conditional rewrite-rule' where a predicate is evaluated with respect to the given term and the rule is applied only if the predicate evaluates to true. Thus a rule, with or without a condition, may be **applicable** to a term or not. 'Term orders' are not considered on purpose: a rewrite-rule like $a \cdot b = b \cdot a$ is applicable to any term containing multiplication (provided appropriate types).

[1] In special cases these can be *several* terms.
[2] In special cases these can be *several* terms.
[3] Another technique is 'matching' [15], which allows to check for equivalence of theorems (rules).

Despite the above restrictions of the topic, this paper still covers a most general operation in doing formal mathematics, which is also crucial in learning: the application of theorems within calculations of applied mathematics.

3 User-interaction in rewriting

One step in an interactive computation as described above reduces to the following structure:

given term → (rewrite) rule → result term

The arrows denote the application of two basic computational skills:

- find an appropriate rewrite rule
- calculate the term resulting from application of the rule

These are also the basic skills to be exercised and assessed in teaching computational dexterity.

3.1 Variants in interaction

The possibilities in user interaction can be broken down to elementary interactions called **dialog atoms** and combinations thereof called **dialog patterns** as proposed in [16].
For further investigation, we will confine ourselves to the following basic set of dialog atoms:

	dialog atom	**responsibility put on the user**
(A1)	have the user enter the correct result	analyse situation, compare to own knowledge, express conclusion
(A2)	have the user fill in blanked subterms like in a cloze test	understand the relationship between the original problem and the presented parts of the result, which may be helpful or distracting
(A3)	let the user choose from a list of variants like in a multiple-choice-test	match the choices to the situation and guess or choose a probable variant
(A4)	show a correct result (proposed by the system) to the user	compare to own reasoning or take for granted
(A5)	skip an interaction because it will be implicitly covered later	none

Table 1

These dialog atoms cover both, the **context** (F) of *finding a rewrite rule* (with and without hinting at the result) and the context (C) of *calculating the result term*:

(F)	**find a rule**	result term		**(C)**	**calculate the result term**
(A1) enter manually		(FC)	shown	(A1) enter manually	
		(F)	*not* shown		
(A2) fill in subterms		(FC)	shown	(A2) fill in subterms	
		(F)	*not* shown		
(A3) select from a list		(FC)	shown	(A3) select from a list	
		(F)	*not* shown		
(A4) see the correct answer				(A4) see the correct answer	
(A5) skip				///	

Table 2

In Table 2, each of the 5 dialog atoms (A1)...(A5) in context (F) must be followed by one of the 4 atoms in context (C) to calculate the result term; this results in 5·4 possibilities.
The 3 atoms (A1)...(A3) in column (F) allow for the variant (FC) of showing the result term in advance. The (FC) variants need *not* calculate the result term any more, i.e. they need not be combined with (C). Finally the number of combinations of dialog atoms completing a step is 5·4 + 3 = 23; in other words: there are (at least) 23 different dialog patterns for one step.

3.2 Feedback on interaction and assessment

Compared to other domains, mathematics has the advantage that decisions can be checked formally (and, in rewriting, immediately) for correctness, allowing for automated and immediate feedback.
In case of success, the only possible feedback is acknowledging the correct result.
In case of error, feedback can be varied as to the amount of additional information given:

feedback of the system	**responsibility left to the user**
result is not correct	re-analyse, draw conclusions, start over
partial hint	solve a subset of the original problem
correct result	compare to own reasoning

Table 3

Note the similarity between choice of feedback in case of error and dialog atoms. In essence, this reduces to the choice of insisting on completing the calculation with the actual dialog pattern, giving „incorrect“ as feedback and forcing the user to try again or choosing another, more supportive, dialog pattern for the next try.
Obtainable feedback in every step of a calculation depends on the amount of choice and responsibility left to the user (as detailed in the table of dialog atoms above) and therefore on the dialog pattern originally chosen. Therefore, the desired amount of support in case of error can serve as a criterion for choosing a dialog pattern.
In any case, immediate feedback is not only possible but also necessary, because sCAS are designed not to proceed if the calculation is not in a consistent, ie proven mathematically correct state.

3.3 Preview: choosing appropriate dialog patterns

Having the opportunity to choose from a wealth of different dialog patterns poses the problem of making the „right“ choice for every particular situation.
While making a random choice already offers the advantage of offering a more varied and interesting learning experience, it is to be hoped that even automated systems can do better than that.
The right choice can depend on a number of factors [17]:

- the topic presently being exercised
- the context of interaction, eg. explorative learning vs. exam
- the amount of obtainable feedback vs. possible frustration arising from excessive demands posed on the user
- the user's preferred learning strategies
- the user's knowledge and experience with the topic

All of these factors are moving targets which not only depend on the particular user of the system but also tend to change rapidly with the user's learning progress and even with the user's present mood.

Setting individual preferences on a per-user, per-topic basis may serve as a starting point, but suffers the drawback that these preferences not only change over time but, more importantly, even the user himself might be only partially aware of them.
In addition to that, to even offer a choice of preferences would require consistent classifications of

- mathematical topics
- human mathematical experience
- human preferences in learning

to express such preferences.
Assuming that suitable classifications of mathematical topics exist and the other classifications exist in their beginnings and can be refined by further didactic research, setting fine-grained individual preferences could be put to the test in experimental systems.
Based on the same assumptions, even an automated, situation-dependent choice of dialog patterns would come into reach [18]. A system choosing appropriate dialog patterns would base its decisions on assumptions about the present state of the individual user, drawn from experience gathered in interaction with the user, classified according to the aforementioned criteria [19,20]. Such a system would use the mathematical knowledge employed during a calculation, the success in doing so and the time spent to build an abstract model of the user – the very same data used for formative assessment of learning progress and already present in the system.

4 Examples and experiences

Here we give examples for some dialog atoms and dialog patterns, and add experiences gained from field-tests [17] with the experimental software [13]. Below the dialog atoms are described by tables with two columns displaying the initial and final state respectively,

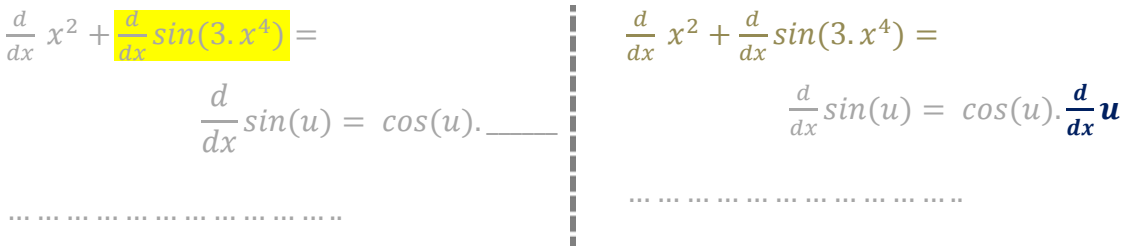
initial state	final state
given term rule_____to given	given term rule **applicable** to given result term

where ___ marks a gap to be filled by the student, ... marks an unused line (above for the result term in the initial state), **this colour** indicates input of the user, and this colour indicates output of the system. The given term is coloured like output, because it is assumed to be checked by the system.
Let us start with a dialog pattern which combines dialog atom (F.A5) with (C.A1). The “Skip” in (F.A5) (i.e. skip finding a rule) makes the user free to apply an arbitrary number of rules at once, and *not* only *one* rule; for instance, the following input should be accepted as a result term:

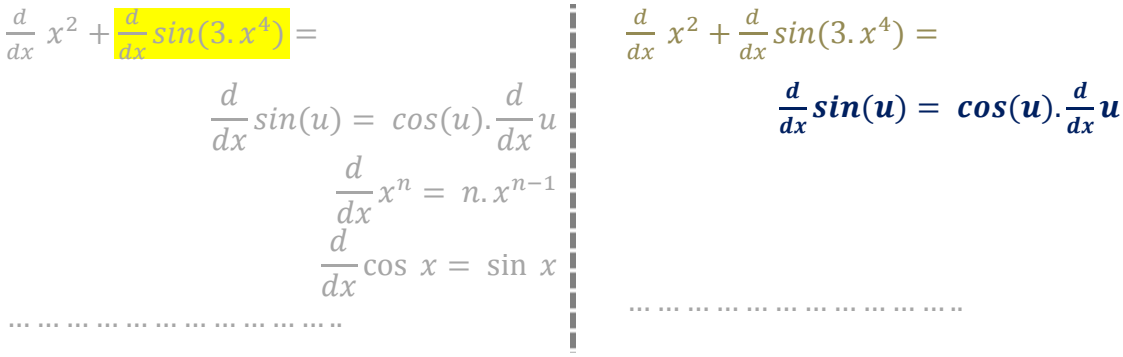
$\frac{d}{dx}\ x^2 + \frac{d}{dx} sin(3.x^4) =$	$\frac{d}{dx}\ x^2 + \frac{d}{dx} sin(3.x^4) =$ .....
.....	$\mathbf{2}.\boldsymbol{x} + \boldsymbol{cos}(\mathbf{3}.\boldsymbol{x}^{\mathbf{4}}).\mathbf{12}\ .\boldsymbol{x}$

Dialog atom (C.A1) is a ‘unique selling point’ of [13] (an open source product). With this system, implementing only (C.A1) and (C.A4), field tests [17] revealed, that these two atoms are not sufficient to lead students to deal with rules and their application – even not, when displaying the applied rules was a fixed feature within all steps.

Now, what if the student cannot provide a correct result term in the above example? The system could provide the result term by (F.A4); but a less challenging dialog atom than (C.A1) may seem a more appropriate choice, for instance (FH.A2), partially providing the chain rule:

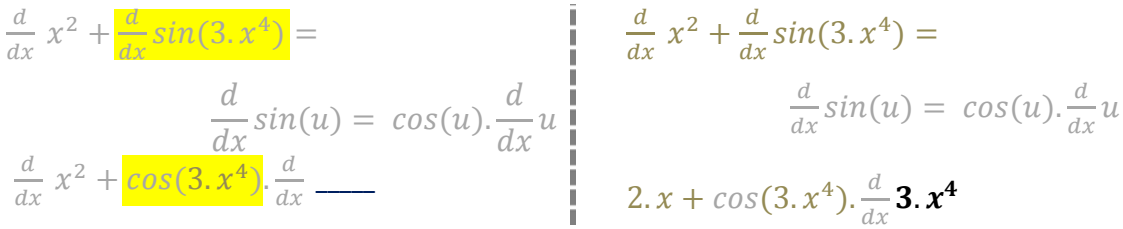


If the student cannot provide the input $\frac{d}{dx}u$, the system can provide it by (F.A4). Again, a still less challenging dialog atom could leave the initiative with the student, for instance (F.A3) selecting the rule from a list:



This example demonstrates, that formula renderers (and editors) should allow to mark sub-terms. If not, more than one choice may be correct (in the above example 2 rules).

Now an applicable rule has been interactively determined, and the next task is to determine the result term. A really smart dialog guide would remember the student's difficulties with the chain rule, and use the same pattern to create a partial result within (C.A2):



By now one single rewrite step has been completed employing 3 dialog atoms of the 23 dialog patterns. As soon as mechanized choice and combination of dialog atoms has been clarified, one may expect many other details coming up. We conclude the examples with just one such open question (which, again, poses no problems for computer mathematics): Should the following input be accepted to dialog atom (C.A1)?

$\frac{d}{dx}sin(u) = cos(u).\frac{d}{dx}u$

$\frac{d}{dx}sin(u) = cos(u).\frac{d}{dx}u$

$\frac{d}{dx}\ x^2 + cos(3.x^4).(12.x^3)$

5 Summary and future work

This paper provides new means for user-guidance in one of the most crucial skills in learning mathematics: the skill of transforming terms correctly and firmly, finally relying on the proper application of rules, and not relying on 'intuition' or some magic. The new means use a well established technology, Computer Algebra Systems (CAS), which model the application of rules (theorems) by steps of 'rewriting'. 'Single Stepping Systems' (sCAS) make the rewrite steps accessible for students.

The novel contribution of this paper is to identify five so-called dialog atoms, which allow for more than *twenty* different dialog patterns to accomplish a single rewrite step. This somewhat surprisingly large amount mirrors the many kinds of interventions a creative human tutor uses to guide a student through the steps of a difficult calculation.

To construct a meaningful dialog pattern, two elements are needed: mathematical knowledge to ask the right question and didactics knowledge to ask the question right.
Asking the right question means generating a problem consistent with the context of a calculation and checking the answer for correctness. Both can be provided by a sCAS, employing matching and rewriting to normal forms. Thus *no* human efforts for assessment, neither for formative nor for summative assessment are required. *No* repeated costs arise, either: A topic (e.g. calculus), once implemented in a sCAS, can be re-used in arbitrary applications of mathematics. Thus the primary effort concerning computer mathematics is implementing mathematics knowledge into sCAS to match the power of present CAS.

Asking the question right, i.e. in a supportive and motivating way, concerns the domain of e-learning and poses a wealth of interesting questions:
Given the wealth of dialog atoms and the large number of dialog patterns for a rewrite step, how to select the appropriate atom at a certain moment during calculation? Can the atoms be combined to 'dialog patterns' covering more than one step? How to select such 'dialog patterns' as an appropriate mechanized reaction to certain situations? Do such patterns relate to some preferences of certain users? Can a user model go beyond “student *x* applied rule *y* correctly *n* times and incorrectly *m* times” to more abstract user behaviour related to dialog patterns? What kinds of preset user models are appropriate for which courses? What is a teacher’s language to parameterise dialogues? What kinds of tools can be provided for teachers to assess the student’s success? How smoothly can standardisations be adopted (IMS QTI etc)?

The authors are convinced that tackling these questions calls for close cooperation between research in e-learning and practice of education. The extent of usability engineering and of feed-back loops required also provides possibilities to transfer knowledge between computer mathematics, e-learning and practice of education.

References:

[1] P. Adams, K. Smith, and Vyborny R. Introduction To Mathematics With Maple. World Scientific Publishing Company, 2004. ISBN: 9812560092.
[2] Stephen Wolfram. The Mathematica Book . Wolfram Research Inc., 1999.
[3] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. Isabelle/HOL, a proof assistant for high-order logic. Springer Verlag, 2008.
[4] Yves Bertot and Pierre Casteran. Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. An EATCS Series. Springer Verlag, 2004.
[5] Christopher J. Sangwin. Assessing elementary algebra with STACK. International Journal of Mathematical Education in Science and Technology, 38:987 – 1002, January 2007.
[6] Eno Tonisson. Checking the equivalence of expressions in computer algebra systems — application possibilities in mathematics education. In The Fifth International Conference on Technology in Mathematics Teaching , University of Klagenfurt, Austria, August 6 - 9 2001.
[7] Walther A. Neuper. What teachers can request from CAS designers. In The Fifth International Conference on Technology in Mathematics Teaching, University of Klagenfurt, Austria, August 6 - 9 2001. http://ftp.ist.tugraz.at/pub/projects/isac/publ/requestCAS.ps.gz.
[8] http://www.chartwellyorke.com/derive/derivefeatures.html.
[9] http://www.ti-nspire.com/tools/nspire.
[10] Wenzel Makarius. Isabelle/isar — a generic framework for human-readable proof documents. In R. Matuszewski and A. Zalewska, editors, Festschrift in Honour of Andrzej Trybulec, volume 10 of Studies in Logic, Grammar, and Rhetoric. University of Bialystok, 2007.
[11] R. Vajda, T. Jebelean, and B. Buchberger. Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis. Mathematics and Computers in Simulation , pages 1–11, 2007. to appear.
[12] http://www.leactivemath.org.
[13] http://www.ist.tugraz.at/projects/isac.
[14] Larry Press. Toward balanced man-machine systems. International Journal of Man-Machine Studies , 3(1):61–73, 1971.
[15] Franz Baader and Tobias Nipkow. Term rewriting and all that. Cambridge University Press,1998.
[16] Alan Krempler. Architectural Design for Integrating an Interactive Dialog Guide into a Mathematical Tutoring System. diploma thesis, Institute for Softwaretechnology, University of Technology, A-8010 Graz, 2005.
[17] Peter Baumgartner and Sabine Payr. Lernen mit Software. Studienverlag, Innsbruck, 1999.
[18] Timothy J. Sliski, Matthew P. Billmers, Lori A. Clarke, and Leon J. Osterweil. An architecture for flexible, evolvable processdriven user-guidance environments. In ESEC/FSE-9: Proceedings of the 8th European software engineering conference, pages 33_43. ACM Press, 2001.
[19] Linton, Deborah Joy, and Hans-Peter Schaefer. Building user and expert models by long-term observation of application usage. In UM '99: Proceedings of the seventh international conference on User modeling, pages 129_138. Springer-Verlag New York, Inc., 1999.
[20] Heimo H. Adelsberger, Markus Bick and Jan M. Pawlowski. Design principles for teaching simulation with explorative learning environments. In J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds, Proceedings of the 2000 Winter Simulation Conference, 2000.
[21] Johannes Reitinger and Walther Neuper, Begreifen und Mechanisieren beim Algebra-Einstieg. IMST Projekt 1063, Klagenfurt, 2008.

Author(s):

Alan Krempler, Dipl.Ing.
Joanneum, University of Applied Sciences
A-8020 Graz, Alte Poststraße 149
alan.krempler@mmm-komm.at

Walther A. Neuper, Dr.techn.
University of Technology, Institute for Softwaretechnology
A-8010 Graz, Inffeldgasse 16b
neuper@ist.tugraz.at