RESEARCH ARTICLE

STACK with state

Matti Harjula, Dept. Mathematics and Systems Analysis, Aalto University, Helsinki, Finland. Email: matti.harjula@aalto.fi
Antti Rasila, Dept. Mathematics and Systems Analysis, Aalto University, Helsinki, Finland.
Jarmo Malinen, Dept. Mathematics and Systems Analysis, Aalto University, Helsinki, Finland.

Abstract

The question model of STACK provides an easy way for building automatically assessable questions with mathematical content, but it requires that the questions and their assessment logic depend only on the current input, given by the student at a single instant. However, the present STACK question model already has just the right form to be extended with state variables that would remove this limitation. In this article, we report our recent work on the state-variable extension for STACK, and we also discuss combining the use of state variables with our previous work on conditional output processing. As an outcome, we propose an expansion to the STACK question model, allowing the questions to act as state machines instead of pure functions of a single input event from the student.

We present a model question using the state variable extension of STACK that demonstrates some of the new possibilities that open up for the question author. This question is based on a finite state machine in its assessment logic, and it demonstrates aspects of strategic planning to solve problems of recursive nature. The model question also demonstrates how the state machine can interpret the solution path taken by the student, so as to dynamically modify the question behaviour and progress by, e.g., asking additional questions relevant to the path. We further explore the future possibilities from the point of view of learning strategic competencies in mathematics (Kilpatrick et al., 2001; Rasila et al., 2015).

Keywords: STACK, state machine, interactive question.

1. Introduction

Various types of e-Learning systems have been used in teaching mathematics for a long time. Currently, there is a wide selection of both commercial and open source systems (such as Maple T.A., Numbas, SOWISO, STACK, etc.) that are capable not only of automatic assessment but also giving useful feedback according to student's inputs. In some of the older systems, the assessment logic is realised rather restrictively by using multiple choice questions or by string comparisons. More advanced technologies make use of symbolic computations with the aid of a Computer Algebra System (CAS) (such as Maple or Maxima) for the evaluation of students' inputs. These systems make it a relatively straightforward task to create teaching materials consisting of series of mutually independent questions, or series of questions with a common theme. It is, however, difficult to produce more multifaceted materials that are able to approach an underlying narrative from many directions in a game-like manner. This is a serious shortcoming in the age of simulations based, interactive e-learning technologies.

One way of introducing modest game-likeness to current systems is to use a series of independent questions so that the selection of the next question depends on student's earlier success. This amounts to applying Computer Adaptive Testing (CAT) techniques in material design. Coupling individual questions in this way is a rather restrictive form of storytelling since it only allows queuing pre-made questions in a changing order. It does not allow the questions to adapt to the student's answers at a finer level.

The next step is to allow the question logic to do adaptation within the question itself, resulting in a more direct, more focused, and more immediate adaptation. In order for this to be possible, the question logic needs some kind of memory (i.e., *internal state variables*) for the current and past forms of the question. Another related desirable feature is an ability to access the memory of the ambient system (such as Moodle for STACK) through *external state variables*.

STACK (Sangwin et al., 2016) has been used at Aalto University since 2006 (Sangwin, 2013), and it has been extensively improved to meet the needs of material developers during that time. However, adaptation of the question based on the student's previous answers has been difficult to implement. Adding the required memory features changes the earlier stateless question model to a stateful model. At the moment, we have an improved STACK question model (and the system executing it) where both the stateless and the stateful questions can coexist (Harjula, 2016). The *stateful question model* adds complications to the implementation and design of materials as well as to the technical side of the executing system.

2. Abstraction of the stateful question

A stateful question can be understood as a set of traditional parametric questions that **(i)** can transfer the student to another such question based on the student's actions, and **(ii)** set the parameters of the consequent question accordingly. Within a stateful question, the component parametric questions are called *scenes* that must be merged to a whole. In other words, the scenes are described in terms of traditional parametric questions. As opposed to coupling of similar questions in traditional CAT, the parameterisation of scenes can be carried of using anything that the CAS can construct (based on student's actions and the earlier state) as parameters. Traditionally, most of question coupling approaches only care about the grade, success, or failure in the preceding questions.

Scenes can be considered as states of a Finite State Machine (FSM). However, the scenes are parametric and parameter values may change during transitions between scenes or in transitions within a single scene. This leads to situations where each distinct scene (i.e., state of the FSM) and the associated parameter values give rise to what we could call the (full) state. In particular, a question with only one scene can be stateful if it has parameters that it can change based on the inputs from the student, and, hence, modify the output it presents to the student. Having just one scene is enough, e.g., for implementing delayed feedback and conditional hints.

The concise terminology of stateful questions is as follows:

state defines the values for all parameters of all scenes and keeps track of the active scene.

scene is a parametric component question that may transfer the student to another scene depending on the input received.

transition is the actual act of transferring between the states, consisting of the scenes and the associated parameter values.

transition condition defines when the inputs from the student and the current parameter values lead to a state transition.

path is the listing of previously visited scenes in sequential order, including the current scene.

In the context of the stateful STACK, the merging of scenes to a single stateful question can be done by using conditional rendering of the question presentation (i.e., text and graphics) where only the details relevant to the current state are presented. The triggering of state transitions can be done using CAS level coding within the free-code portion of the normal response generation.

3. Modifications to the question model of STACK

The original STACK handles an user input event along the following lines:

1. Reconstruct the *question variables* based on a (potentially earlier) generated, original random number seed. Using the original seed, generate all the random parameter values required by the question.
2. Identify from the input of the student which ones of the (potentially many) input fields of the question have received valid values.
3. Find those *potential response trees* (PRTs) within the question that have received valid input values for all of their input variables required for processing the PRT. Then for each such PRT:
    a. Evaluate the *feedback variables*, i.e., the free-code portion of the PRT to preprocess the input values. This portion and the PRT nodes may reference the question variables and are in that sense parametric.
    b. Traverse the nodes of the PRT starting from the root and evaluating each the binary *answer test* of each node in order to decide to which branch of the PRT to continue, until the branch ends. At each node, modify the points given and the penalties accrued for this question part for this attempt, based on the binary answer test result. Also, each node may append to the feedback presentation connected to this PRT, again based on the test result.
4. Render the question presentation based on the variables reconstructed at step 1, inserting the feedback generated by the PRTs and input validation inline to it, if required.

As this description is about a stateless model, the processing of the answer does not change the way the forthcoming processing rounds would act. Here, the general presentation of the question stays the same, and only feedback or validation messages are added to it.

The stateful extension modifies the above described outline so that the free-code portion of PRTs is allowed to write values into the state. These values can later be referenced in all other stages of question processing, including the rendering of the question presentation. In rendering, it is now possible to select the parts to be displayed based on the state. Moreover, the PRTs may evaluate the properties of the student's answer based on the state constructed from previous answers, contrary to the original STACK.

Note that the original random number seed is still in control of all random parameter values. This means that these parameter values are “frozen” to the same even if the question had cycles and the author would like to construct new random values for every time a specific scene is returned to. It would be more desirable for the author to be able to decide, based on the path, whether one would present the scene using the original random number seed or a new seed constructed from the path so as to avoid repetition.

To identify which questions have state, the proposed extension requires that all state variables (both local to this question instance and external) must be declared by the author. The declaration must also define if the variable is read-only, and set the variable's default value should it not have been initialised elsewhere. Questions without writable state variables are by definition stateless; i.e., a stateless question may have references to an external state that is used similarly as the random number seed in question initialisation. If the question, however, has a writable state variable, all CAS evaluations performed by the question must be augmented with calls that transfer the current state to the CAS. Conversely, the relevant output from CAS must be stored to a state variable.

When a question references external state variable, its value is stored as a local variable at the initialisation of the question. All references to this external variable are then rerouted to the local

variable, thus eliminating problem due to changing external state variable values while the question instance is being used. This, however, makes it difficult to change the external state as the current value is not shown directly to the question. To overcome these complications, the external state is typically handled as a data structure that can only be updated (e.g., a counter that gets incremented, a set that gets augmented, etc.) rather than entirely rewritten.

4. Three classes of stateful questions

The stateful question type makes it feasible to produce materials with which higher abilities – such as strategic competencies – can be trained and assessed instead of drilling. As examples, consider these classes of stateful questions having narratives:

1. The *constrained-path* questions demonstrate the progress of some algorithm or a more general collection of fixed rules and verify that the student's answers match the steps. The well-defined states and steps can be easily reproduced by CAS for verification. Collections of rules may leave the student some freedom of choice so as to use strategic thinking for attaining the desired goal. Erroneous values are typically not accepted in answers but the student is allowed to go in a wrong direction for a while.
2. In *sudoku* questions, some initial setting is given with incomplete details. The student is asked to gradually fill in the missing details by logical reasoning from what is already regarded as known. The next detail to fill in is chosen randomly, based on a dependency graph that describes what can be concluded by relatively few logical steps from what is already known. One may also consider accepting mistakes and allowing them to propagate in reasoning without letting the student know, but then there has to be a way for the student to fix the errors afterwards.
3. A dynamical mathematical (state space) model of some phenomenon lies in the core of *model-based* questions. The student's answer is a control input, and the state transitions are computed by the model. There are means for evaluating whether (and at what cost) the desired target state has been achieved, and there may even be dynamically generated new targets.

All these types can be realised as stateful STACK questions but their designs may get very complex in the two latter classes since many states are required to handle the storytelling logic and to separate it from the state of the mathematical model. The designs may benefit from a narrative backbone such as Interactive Storytelling (IS) taken from the game design context (Crawford, 2012). Within mathematics e-learning, storyboards have not been widely used even though various requirements for successful high-level system designs have been proposed in, e.g., Devlin (2011). We emphasise that the storytelling aspect is more important in authoring than programming skills since a stateful question must react sensibly and reasonably to students' various answers.

The three classes introduced above are not to be understood as exclusive or canonical but they help in characterising and comparing different types of possible stateful questions. Obviously, stateless questions are one end of the scale, and we could call them the *no-path* type in comparison to constrained-path questions where allowed transitions between scenes are very limited; however, not excluding infinite loops. Sudoku type questions have transitions between scenes that are best described are non-trivial directed graphs without cycles. Similarly, model-based questions are expected to lead to undirected, dense graphs as transition paths between highly parametric scenes.

The logical end point of this scale could be called the *free-path* question type where so little restrictions in transitions between scenes are posed that the student may not even notice any restrictions of freedom. As we move towards the model-based and free-path designs, we expect that methods from IS become particularly useful, starting from constrained narrative generation (Porteous and Cavazza, 2009) and leading to full-on drama management. We further expect that the design of a pedagogically relevant free-path question in mathematics to be very difficult.

5. Stateful question design principles

One example of a stateful question is given below. A prototypal design process for this kind of questions is outlined as follows:

1. Design the narrative for the question, consisting of scenes that can be represented as usual STACK exercises.
2. Describe the narrative as a FSM diagram such as Fig. 1 where each scene corresponds to one or several potential tasks. Enumerate all scenes and describe their scene transitions in response to student's answers and parameters.
3. Specify what information the scenes need as parameters. Define the necessary internal state variables to convey this information.
4. Produce the rendering code of the user interface for each scene.
5. Realise each scene like an usual STACK exercise whose response tree is activated by the student's answers. Add transition conditions and transitions to the free-code portion of the PRTs to produce transitions between the scenes.

The author should minimise the number of scenes and state variables without making the code unreadable. The generation of the question texts is challenging as the text depends on the full state, leading to many variants.

6. An example question of constrained-path type

We present an example of a constrained-path question involving integration by parts where the integrand includes a monomial. The student is allowed to split the integrand into two parts at will, and then use integration by parts to compute the antiderivative. Some strategic competency is required in addition to computation skills. Indeed, the degree of the monomial may increase as the result of a silly chosen splitting $f = u\,v'$, and the student gets further away from the right solution.

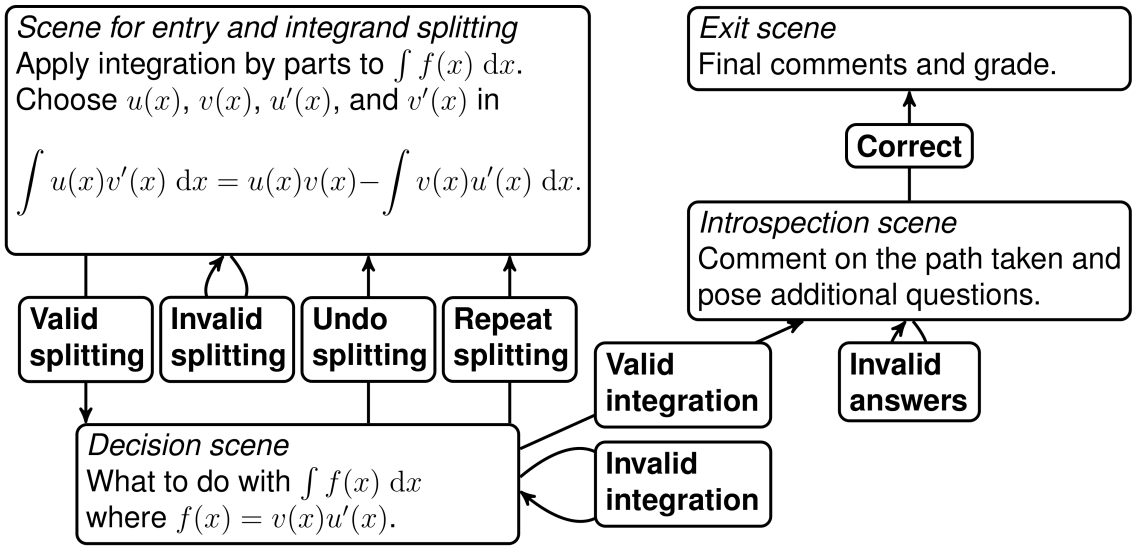

Figure 1. The state transitions of the example question having four scenes.

The question consists of four scenes as described in Fig.1, two first of which are being visited repeatedly. The user interfaces associated to these scenes are given in Fig 2–4 below. More precisely, the scenes are as follows:

Entry and integrand splitting: The initial or current integration by parts exercise is displayed by this scene, and the student is asked to split the integrand for the integration by parts process.

Decision: In this scene, the student is shown the valid splitting given earlier. Now it is possible to revert the earlier splitting, choose applying integration by parts again, or just compute the integral by other means.

Introspection: The student arrives at this scene after having produced the correct answer. Comments are given on the solution path of the student, and relevant control questions are asked concerning the solution strategy.

Exit: This scene shows the points and penalties with final comments.

The state transition rules of the question are described in Fig. 1. These allow the student to go on integrating by parts as far as they wish (within storage/memory limitations). The question is initialised with a randomised degree of the monomial in the integrand. The question state only keeps track of the two expressions generated by the process, the ejected part and the remaining integral, as a list consisting of all the values generated by previous splittings, the current scenes name, and some flags signalling which warnings have been given. The most difficult part in the authoring of this question is describing the logic for checking if the given splitting of the integrand leads to the desired direction in integration by parts process.

The stateful question type makes it possible to give specific feedback to the student after complex conditions are met. In the example question, we start to warn about going in the wrong direction only after multiple steps of integration by parts in that direction. The warnings get more direct if the student continues even further. It is also detected if the student returns to the original expression after many steps, or if the student tries to integrate by parts an expression that does not require it. A simple way of constructing the final grade is to keep track of the warnings that have been given, and this is the approach used here. More complicated evaluations can be based on the trajectory that is stored as a list of uv and vu' expressions.

Any of the four scenes in the example question can be implemented without using states. The point of example question is in joining those scenes as a dynamic storyline and evaluating the path the student takes. This is impossible to achieve without having the kind of memory that the state variable extension provides.

7. Conclusions

The state variable extension to STACK was proposed. The extension can be used to produce e-learning materials that respond more adaptively and intelligently than earlier such materials. An elementary example involving integration by part was given to show some of its technological and pedagogical potential.

As *STACK with state* gets technically more mature, the next big step is to build advanced tools to simplify the question authoring process. Drawing diagrams such as Fig. 1 and manually translating them into conditional branching statements is not the way to go when building large scale applications – it is better to have a more refined “method in madness”. The state dependent randomisation of questions is highly nontrivial, and its framework remains unspecified at the moment. Furthermore, there is need for analytical systems for **(i)** evaluating the structure of a stateful question, (i.e., static code analysis), and for **(ii)** analysing in bulk the solution paths (i.e., the state trajectories) generated by the students. The latter requirement relates to learning analytics.

In this article, we concentrated on finite states (scenes) and internal state variables. External state variables are used for accessing the global state in an ambient system (such as Moodle), providing an interface for an external learning analytics system, or used for sharing student specific information between different stateful STACK questions.

In this question we want you to apply integration by parts to

$$\int x^2\, e^{2\,x}\, \mathrm{d}x.$$

As a reminder by integration by parts we mean

$$\int u(x)v'(x)\, \mathrm{d}x = u(x)v(x) - \int v(x)u'(x)\, \mathrm{d}x.$$

To start the exercise select $u(x)$ and $v'(x)$ for the first integral $\int x^2\, e^{2\,x}\, \mathrm{d}x$.

$u(x) =$ exp(2*x) $v'(x) =$ x^2

$u'(x) =$ 2*e^(2*x) $v(x) =$ x^3/3

Check

The integration has progressed as shown below and we are still applying $\int u\, \mathrm{d}v = uv - \int v\, \mathrm{d}u$to

$$\int x^2\, e^{2\,x}\, \mathrm{d}x = \frac{x^3\, e^{2\,x}}{3} - \frac{x^3\, e^{2\,x}}{3} + \int x^2\, e^{2\,x}\, \mathrm{d}x.$$

We now continue from the previous step. In order to compute the following integral, select $u(x)$ and $v'(x)$ for it:

$$\int x^2\, e^{2\,x}\, \mathrm{d}x.$$

That integral looks oddly familiar, let's hope we do not see it again.

$u(x) =$ x^2 $v'(x) =$ exp(2*x)

$u'(x) =$ 2*x $v(x) =$ exp(2*x)/2

Check

Figure 2. The entry scene of the example question presents the currently relevant integral and asks to split the integrand. In the lower panel, the student has returned to the starting point after some unlucky choices.

Your selection placed to the formula leads to

$$\begin{aligned} u(x) &= e^{2x} \quad v'(x) = \tfrac{2x^3}{3} \\ u'(x) &= 2e^{2x} \quad v(x) = \tfrac{x^4}{6} \end{aligned} \rightarrow \int \frac{2x^3 e^{2x}}{3}\,dx = \frac{x^4 e^{2x}}{6} - \int \frac{x^4 e^{2x}}{3}\,dx$$

which means that you will still have to integrate

$$\int \frac{x^4 e^{2x}}{3}\,dx.$$

You now have a few options on how to continue. You can either just give the value of that integral, and if it is correct, this whole process ends, or you can repeat the same integration by parts on that integral and hopefully get an easier integral from it. You can also undo your selection and try again with another $u(x)$ and $v'(x)$.

Have you noticed how the order of that term in the integral grows? Surely, the integral would be simpler to solve if that order went down instead?

$\int \frac{x^4 e^{2x}}{3}\,dx =$ [ ] Repeat integration by parts Undo this selection

Check

Your selection placed to the formula leads to

$$\begin{aligned} u(x) &= \tfrac{1}{2} \quad v'(x) = e^{2x} \\ u'(x) &= 0 \quad v(x) = \tfrac{e^{2x}}{2} \end{aligned} \rightarrow \int \frac{e^{2x}}{2}\,dx = \frac{e^{2x}}{4} - \int 0\,dx$$

which means that you will still have to integrate

$$\int 0\,dx.$$

You now have a few options on how to continue. You can either just give the value of that integral, and if it is correct, this whole process ends, or you can repeat the same integration by parts on that integral and hopefully get an easier integral from it. You can also undo your selection and try again with another $u(x)$ and $v'(x)$.

You are now integrating a constant and not just any constant but zero. Perhaps now it would be a good point to stop iterating and just integrate it?

$\int 0\,dx =$ [ ] Repeat integration by parts Undo this selection

Check

Figure 3. The decision scene gives information of the current situation and allows the student decide how to proceed. The student may continue to integrate by parts even if the integrand is zero.

Well done! You have reached the end of integration. And you have shown the following facts:

$$\int x^2 e^{2x}\,dx = \frac{x^3 e^{2x}}{3} - \int \frac{2x^3 e^{2x}}{3}\,dx$$ ***A bad move, you raised the order of the term.***

$$\int \frac{2x^3 e^{2x}}{3}\,dx = \frac{x^3 e^{2x}}{3} - \int x^2 e^{2x}\,dx$$

$$\int x^2 e^{2x}\,dx = \frac{x^2 e^{2x}}{2} - \int x e^{2x}\,dx$$

$$\int x e^{2x}\,dx = \frac{x e^{2x}}{2} - \int \frac{e^{2x}}{2}\,dx$$

$$\int \frac{e^{2x}}{2}\,dx = \frac{e^{2x}}{4} - \int 0\,dx$$ **Here you should have just integrated it.**

$$\int 0\,dx = 0 - \int 0\,dx$$ **Splitting 0 to parts was probably great fun.**

$$\int 0\,dx = 0 + C$$

$$\int x^2 e^{2x}\,dx = \frac{x^2 e^{2x}}{2} - \frac{x e^{2x}}{2} + \frac{e^{2x}}{4} + C$$

For some reason you chose to apply integration by parts to $\int 0\,dx$. Why?

- [ ] Wanted to know if integration by parts can be applied for it.
- [x] Got stuck on a loop and did not notice when to stop.
- [x] Tried to see if the question can handle it.

Now some questions about the motivation for the choices. The basic idea is:

- ( ) Not answered
- ( ) Always pick the first term as $u(x)$.
- ( ) Pick $u(x)$ so that differentiating it will make it disappear

Congratulations! You have reached the end of the question and actually got some points:

100% For reaching the end.
-0% For going to the wrong direction.
-0% Saw the original integral for the second time.
-0% Explored integration by parts in the case of a constant value.
100% Total.

This is the end of this question about integration by parts and you should have learned that the selection of the parts tends to require some heuristics. If you need some guidance on how to select the parts, please explore the LIATE rule or do some tests with different selections and see where you get with them. But remember that it is not necessary to get anywhere and it might even be handy to return to the same place like with $\int e^x \cos(x)\,dx$...

You have been given 3 of the 6 possible warnings. Feel free to restart to search for more.

Restart

Check

Figure 4. The introspection scene presents multiple choice questions about matters that should have become apparent during the solution process. The exit scene gives the grading with comments.

We conclude that the most exciting aspect is the exploring the new types of mathematical questions made possible by the proposed *STACK with state*, its future modifications, and other improvements. We believe that a practically limitless *terra incognita* of possibilities opens up for game-like e-learning materials in mathematics, challenging the current technological and pedagogical paradigms.

8. References

Crawford, C., 2012. Chris Crawford on Interactive Storytelling. New Riders.

Devlin, K., 2011. Mathematics Education for a New Era: Video Games As a Medium for Learning, 1st Edition. A. K. Peters, Ltd., Natick, MA, USA.

Harjula, M., 2016. Stateful extension, frozen proof of concept code version. Available at: https://github.com/aharjula/moodle-qtype stack/tree/EAMS-frozen-state [Accessed 12 December 2016].

Kilpatrick, J., Swafford, J., Findell, B., 2001. Adding It Up: Helping Children Learn Mathematics. Mathematics Learning Study Committee, Center for Education, Division of Behavioral and Social Sciences and Education, National Research Council. Washington, DC: National Academy Press.

Porteous, J., Cavazza, M., 2009. Controlling narrative generation with planning trajectories: the role of constraints. In: Joint International Conference on Interactive Digital Storytelling. Springer Berlin Heidelberg, pp. 234–245.

Rasila, A., Malinen, J., Tiitu, H., 2015. On automatic assessment and conceptual understanding. Teaching Mathematics and its Applications 34 (3), 149–159.

Sangwin, C., 2013. Computer Aided Assessment of Mathematics. Oxford University Press.

Sangwin, C. J., Hunt, T., Harjula, M., et al., 2016. STACK, code of the master version. Available at: https://github.com/maths/moodle-qtype stack [Accessed 12 December 2016].