Chief Chefs of Z to Alloy: Using a Kitchen Example to Teach Alloy with Z

Sureyya Tarkan and Vibha Sazawal*

University of Maryland, Department of Computer Science,
College Park, Maryland, 20742, USA
{sureyya,vibha}@cs.umd.edu
http://www.cs.umd.edu/users/sureyya/

Abstract. Z is a well-defined and well-known specification language. Unfortunately, it takes significant expertise to use existing tools (such as theorem provers) to automatically check properties of Z specifications. Because Alloy is substantially similar to Z and the Alloy Analyzer offers a relatively simple method of model checking, we believe that Alloy should be largely employed in classes that teach Z. To this end, we present an online tutorial especially designed to help students transition from Z to Alloy. The tutorial includes both the classic Birthday Book example and a large real-world scenario based on a Kitchen Environment. Our experiences with novices studying the tutorial suggest that the tutorial helps students learn both Z and Alloy. In addition, novices can answer questions correctly about the approximately 500-line Kitchen Environment model after only a few hours of study.

Keywords: Formal Methods, Formal Specification, Z, Model Checking, Alloy.

1 Introduction

When teaching Formal Methods, it is important to choose a widely understood notation as the mathematics inherent in concepts should be unambiguous. Z [22] is a well-defined and well-known specification language which emphasizes the mathematical parts of formal definitions. A number of successful textbooks have been written using Z [9,27] suggesting that educators prefer this specification language over others available to them.

Although teaching the necessary mathematics to define software systems formally is helpful in software engineering education, mathematics alone is not enough. Verification and model checking tools are becoming increasingly popular [14]. Specifications are large and cannot be verified by manual inspection alone; thus, automatic checking of specification properties is essential. Educators are also expected to incorporate this change in their curriculum. Many software tools are introduced to facilitate this process but it is clear that model checking

* Special thanks to John C. Knight for his help on this paper.

J. Gibbons and J.N. Oliveira (Eds.): TFM 2009, LNCS 5846, pp. 72–91, 2009.

is one of the winners [13]. Students who know mathematical notation may now need to learn a separate notation to write and check their models. Given the problem of lack of interest already present in Formal Methods classes [7,16,19], it is unreasonable to expect students to be highly motivated to learn various notations and be able to apply all of them later on.

As Z is one of the most popular formal notations, students should learn a technique that enables them to master tool usage besides mathematical knowledge. However, this should be accomplished without the burden of learning a new language. We believe that Alloy, in fact, provides such a functionality and is substantially similar to Z as noted by Jackson [8]:

> "The language, Alloy, is deeply rooted in Z. Like Z, it describes all structures (in space and time) with a minimal toolkit of mathematical notations, but its toolkit is even smaller and simpler than Z's."

The advantages of using Alloy in class can be summarized as follows: (i) very roughly, Alloy can be viewed as a subset of Z [2], (ii) Alloy and Z are both based on logic and set theory, (iii) Alloy, unlike many theorem provers for Z, performs fully automatic analysis without any guidance from an experienced user (as students are obviously novices), (iv) Alloy Analyzer [1] is consistently maintained by a group of researchers at MIT, (v) unlike some other model checkers, Alloy is free and can be used in the classroom, (vi) Alloy users share materials on the website forums in an online community format.

One would expect that Alloy would have been employed often in classes that teach Z. However, we see that there is a lack of examples and educational material for the transition from Z to Alloy. In particular, educators develop examples and material for either Z or Alloy yet there is insufficient courseware for Z to Alloy. Moreover, the examples that have the two versions (e.g. the Hotel locking example found in Jackson [8]) do not adequately address the relationship and differences between these languages. We suggest that this gap should be filled in with interesting examples and well-documented lesson plans.

However, there are a number of challenges in making Z to Alloy comprehensible for students. We think that employing educational theories in the preparation of these materials plays an important role. For example, the Montessori method of directing students' interests with the increasing complexity of the material [15] is helpful. Moreover, it is essential to point out both the similarities and differences between the languages. Accordingly, developing user-friendly interfaces for educational documents is also crucial to support navigation through content, for example, from reviewing discrete math and logic background, to showing formal notations, and finally, to using a tool.

As the previous paragraphs emphasize, to populate the educational materials for Z to Alloy, we suggest the use of a running example that is easy for students to understand and yet self-explanatory and comprehensive to illustrate the required processes involved in the transition. We argue that it would not be difficult for students to learn Alloy particularly if we employ our Kitchen Environment project [23]. Our contributions in this paper are twofold:

1. First of all, we present our online tutorial to teach Z to Alloy with the Kitchen Environment real-world example. In order to do that, we implement a conversion technique that simplifies this process.
2. Finally, we share our experiences with novice students using the tutorial.

The rest of the paper is organized as follows. First, we survey previous work in the area. Second, we show our full tutorial implementation. Third, we explain our case studies with novice students. Then, we discuss the results and implications of our design. Finally, we conclude with future work.

2 Related Work

The related work can be investigated from three different perspectives. First, we talk about the interactive application of Formal Methods. Next, we look at educational materials similar to ours. Finally, we mention how content-wise we employ previous work in our tutorial.

Dean [5] talks about the development of an interactive case e-study. Formal Methods materials usually reside on conventional paper and there are some difficulties in converting them to interactive electronic versions. Therefore, Dean has developed a hyperlink structure, which is a mixture of HTML files and PDF documents. One advantage is that there is no need to save paper so the material can easily be broken up into manageable sections that are interrelated. The downside is it becomes essential to match the dimensions of the computer screen. In addition, the most important aspect of the design is navigation through the document. We mostly agree with these comments but we do not make use of PDF in our implementation since the level of expected interactivity is high in our case and PDF format cannot completely fit our needs. We have used images due to the heavy mathematical content, however, there are packages like T$_T$H [24] that can convert LaTeX files easily into HTML.

Similarly, Pandora [4] is a tool developed for teaching first order natural deduction. It contains a friendly e-tutor component that provides hints, explanations, warnings, and counterexamples for corresponding student actions. This context sensitive e-tutorial has help and various other facilities for saving, loading, and printing proofs by exporting them to LaTeX. Pandora has been extensively used in class by students for their coursework and exams. When the e-tutorial is started, four tutorials of propositional exercises are available. The first consists of a fixed set of exercises and is useful in laboratory sessions. For the others, there are three levels for "easy," "medium," and "hard" each with five exercises that get randomly selected at run time by the e-tutor. Our tutorial, on the other hand, is not developed for the purpose of teaching natural deduction. Moreover, our interactive tutorial implementation is not part of such a tool, although it can easily be incorporated into one as it runs from the web.

Rosa [20] suggests the use of Piaget's theory [17] in Formal Methods education. More specifically, the paper proposes a shift of focus from the development of calculation skills to the encouragement of active participation in discrete math education. To do that, new epistemological frameworks are necessary. One such

framework is provided by the genetic epistemology theory of Piaget [18], which claims that acquiring knowledge is governed by the laws of human biological development. The paper presents an application of this theory to students, who design algorithms for solving problems. Finally, it argues that supporting instructional material is essential. In accordance, we borrow similar ideas from the pedagogical work of deBry [6] inspired by Kolb's learning theory [10]. We particularly make use of the iterative process involved in the learning of humans via employing an immediate feedback mechanism integrated into our tutorial.

Apart from such a framework, Rudall [21] develops a module from Z to SPIN. The module is divided into two parts, whose first part deals with formal specification whilst the second part deals with formal verification. The main focus of the formal specification is the Z language. The topics covered are schemas, relations, functions, and sequences; however, not every detail, e.g. the refinement and proof, of Z notation is shown. In the second part of the module, at first, temporal logic is introduced so that the Promela [25] modeling language and the SPIN [26] model checker can be taught. Promela is used to model complex process interactions and SPIN is used to verify these models in the lab sessions. The paper indicates that the examples and theory are worked through on the board in class mainly with student interaction but for practice, students are expected to follow the tutorials using software tools in the labs on their own. It is believed that to occupy the interest of students, a broad base course is more useful than a narrow focus. Although the author clearly identifies the reason for the selection of Z, she admits that the Promela language is chosen subjectively. We depart from this study in the following ways. We make use of the Alloy modeling language and tool as it is very similar to the Z notation. Furthermore, we embed the in-class interaction within the tutorial.

Many studies reflect on the fact that real-life examples make learning of Formal Methods more interesting for the students [3,11,12,19]. Brakman [3] uses a project about a Bluetooth communication protocol to increase the "fun-factor," and henceforth, the attendance rates of the classes. Similarly, to motivate students with hands-on experience, Larsen [11] assigns students interesting class projects to work on. Moreover, Lightfoot [12] proposes an interesting group competition based on the vote-recording software of the Eurovision Song Contest. Finally, Reed [19] stresses that the use of small, simple, and practical examples (Fibonacci numbers, integer division, invariants, the Needham-Shroeder protocol) is helpful for novice programmers in the early stages. Inspired by the successes of these prior studies, in our tutorial, we make use of first a simple well-known example, the Birthday Book [22], and later a comprehensive but everyday-life example that we developed, the Kitchen Environment.

3 Z to Alloy Tutorial

In this section, we describe the details of our online interactive tutorial, which is publicly available at our website [28].

3.1 Tutorial Content

The content is prepared to teach the concept of "model checking formal specifications" to novice students. It is organized as follows:

– **Introduction.** This section covers the motivation for what formal methods are, what they are used for, and some examples that illustrate disastrous cases when they are not properly applied in practice.

– **Birthday Book.** This simple example is presented in detail with two separate implementations (one for the formal Z specification and the other for the Alloy model) to give the mathematical and logic background as well as to explain the details of Z and Alloy languages.

– **Alloy Analyzer.** The tool is reviewed to show what its capabilities are.

– **Kitchen Environment.** This comprehensive example is developed using a conversion technique from the Z specification to the Alloy model. This model is pretty long compared to the specifications that are covered in textbooks and therefore successfully relates the concept to real software specifications used in industry.

– **Quiz.** A quiz is given to assess the student's understanding of the material. The questions ask the student to complete and fix an Address Book model [8]. This is similar to the Birthday Book with a difference that it saves people's addresses.

We explain the Birthday Book and Kitchen Environment examples in more detail in the sections that follow.

The Birthday Book. Our Z specification of the Birthday Book is adapted from Spivey [22] and our Alloy model is directly taken from the Alloy distribution of the sample models [1]. The Birthday Book records people's birthdays (a name and a date for each person), and places a reminder when the appropriate day comes. It can be populated and depopulated, and there is a search option.

The Birthday Book specification includes the type declarations, the data objects, the state space, the initial state, and the operations – add, delete, find, and remind – as schemas. Within this specification, we review concepts from set theory and more specifically, the operators and symbols that are commonly used in the Z notation.

The corresponding Alloy model is then used to automatically check the subsequent assertions:

1. Adding a birth date to the book indeed works.
2. Deletion is an undo of addition.

To explain the Alloy model, we introduce the Alloy notation, i.e. modules, signatures, atoms, relations, relational product operators, multiplicity markings, predicates, assertions, and checks, as well as how to interpret the output.

The Kitchen Environment. This example is adapted from Tarkan [23]. The task is to simulate the actions of a kitchen chef to direct other cook(s) in the preparation of a dish (given the recipe). The primitive object types are `cook`,

ingredient, kitchen_item, and measurement and the built-in functions are Bake, Clean, Cut, Knead, Mix, Preheat, and Put.

The complications in the specification are identified in the following manner. Multiple cooks can function concurrently, some kitchen items work autonomously, multiple ingredients get composed to make others, ingredients change their states from Raw to Baked, Cut, Kneaded, Mixed, or Processed. To this end, an event-driven programming approach is suggested. For this purpose, we first write an approximate Z specification and then convert it into a full Alloy model that enables the student to perform automatic analysis. We do not develop a full Z specification since it is going to be modified to meet the Alloy language requirements. Lastly, we identify the following list of assertions as crucial for our system.

1. Item is the same tool after getting Cleaned.
2. Preheat always precedes Bake.
3. Preheat is the only way to heat an item.
4. A HeatedItem should always be available for use to protect against fire.
5. A UsedIngredient never becomes available again.
6. A newly created AvailableIngredient[1] is composed of only UsedIngredient(s), which were available prior to the creation of this ingredient.
7. Raw ingredients are never a composition of other ingredients.
8. All AvailableIngredients that are not Raw are compositions of other ingredients.

Given the problem statement, we show how to convert sets in the Z specification to signatures in the Alloy model. We present the sets of cooks, kitchen items, ingredient names, measurements, and event identifiers in the Z notation. Apart from these, an ingredient is represented with a schema that has fields for its name, amount, and its constituents that change over time. The subsets of ingredients are also defined for Baked, Cut, Kneaded, Mixed, Processed, and Raw. Direction saves all the events called so far and the arguments to these calls. We first convert the sets for cook and kitchen item to signatures. At this point, we tell the student what the limitations of model checking are. For instance, some sets that do not affect the state are not declared in the Alloy model to prevent extensive memory usage. We also explain the limited support for integers in Alloy as they are infinite and introduce our Time signature that is represented with an integer in Z. We introduce built-in modules such as ordering supported by Alloy. We show them the object-oriented nature of Alloy as opposed to Z with the Baked, Cut, Kneaded, Mixed, Processed, and Raw disjoint subsets of the ingredient type. In this manner, we clearly point out the differences and similarities between these approaches.

Next, we show the state specification in Fig. 1. We again first talk about the Z schema and reflect on our decisions throughout its definition, especially the relationship to time is stressed and how this is employed in the form of relations within the Z version. We finally talk about the invariants and their importance

[1] This ingredient cannot be raw but can only be in one of the following states: baked, cut, kneaded, mixed, or processed.

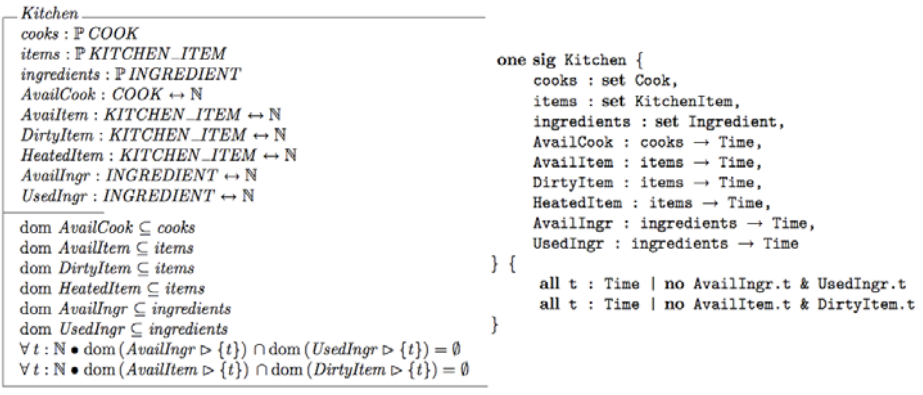
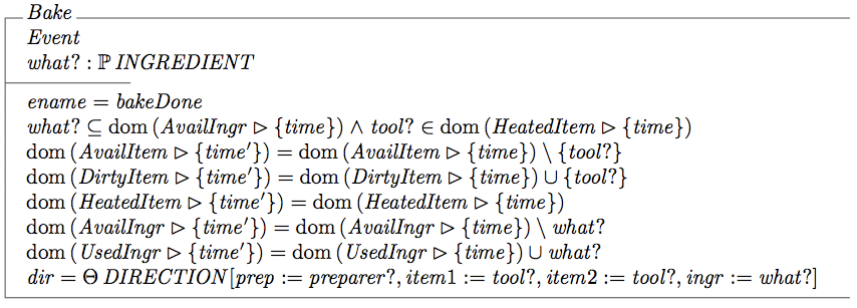


Fig. 1. Kitchen schema and signature. Kitchen acts as a database in the system. There are cooks, items, and ingredients in this world. Kitchen records available cooks, items, and ingredients, dirty and heated items as well as used ingredients. All of these entities depend on time. There are two important invariants of the system: an item cannot be both available and dirty at the same time and analogously available and used ingredients are disjoint at all times.

to our system. With this, we complete our Z schema and start converting it into an Alloy model. We immediately comment on Alloy's capabilities for the multiplicity of signatures. After simply saying that the body is very similar, we note the fact statement that is appended to the signature. We introduce how facts can be used to specify the invariants. The final step is to call to the student's attention the absence of the `Timer` schema in the Alloy model. Because a timer is a mechanism that keeps track of incremental properties like time and event count, after explaining all the details related to this schema, we inform the student that such entities will be handled with an equivalent but space-saving "trick" in Alloy.

As we are finished with the state specification, we move to the state initialization for the Z version and talk about the details in a similar fashion as the previous one. As soon as we start writing the Alloy model, we highlight the difference between Z, which represents operations with schemas, and Alloy, which uses a distinct notation for the dynamism, i.e. predicates. Afterwards, we proceed to the conversion.

One of the complications in this example is the need for events that are the cause of concurrency. We split events into the first call and the announcement of completion. Therefore, we have events for `Bake`, `Clean`, `Cut`, `Knead`, `Mix`, `Preheat`, and `Put` but also `BakeDone`, `CleanDone`, `CutDone`, `KneadDone`, `MixDone`, `PreheatDone`, and `PutDone`. When we are done with the state space and initialization, we convert the events defined as schemas in the Z notation to predicates in the Alloy version. We first start with `Event` and `EventDone` schemas that will be reused in the specific event calls and thus, contain definitions common to all of them. We review all of their definitions. Moreover, as



Bake

Event

$what? : \mathbb{P}\, INGREDIENT$

$ename = bakeDone$

$what? \subseteq \mathrm{dom}\,(AvailIngr \rhd \{time\}) \land tool? \in \mathrm{dom}\,(HeatedItem \rhd \{time\})$

$\mathrm{dom}\,(AvailItem \rhd \{time'\}) = \mathrm{dom}\,(AvailItem \rhd \{time\}) \setminus \{tool?\}$

$\mathrm{dom}\,(DirtyItem \rhd \{time'\}) = \mathrm{dom}\,(DirtyItem \rhd \{time\}) \cup \{tool?\}$

$\mathrm{dom}\,(HeatedItem \rhd \{time'\}) = \mathrm{dom}\,(HeatedItem \rhd \{time\})$

$\mathrm{dom}\,(AvailIngr \rhd \{time'\}) = \mathrm{dom}\,(AvailIngr \rhd \{time\}) \setminus what?$

$\mathrm{dom}\,(UsedIngr \rhd \{time'\}) = \mathrm{dom}\,(UsedIngr \rhd \{time\}) \cup what?$

$dir = \Theta\, DIRECTION[prep := preparer?, item1 := tool?, item2 := tool?, ingr := what?]$

```
pred Bake[t, t' : Time, preparer : Cook,
        tool : KitchenItem, what : set Ingredient] {
    preparer in Kitchen.AvailCook.t
    tool in Kitchen.HeatedItem.t
    tool in Kitchen.AvailItem.t
    some what and what in Kitchen.AvailIngr.t
    Kitchen.AvailCook.t' = Kitchen.AvailCook.t - preparer
    Kitchen.AvailItem.t' = Kitchen.AvailItem.t - tool
    Kitchen.DirtyItem.t' = Kitchen.DirtyItem.t + tool
    Kitchen.HeatedItem.t' = Kitchen.HeatedItem.t
    Kitchen.AvailIngr.t' = Kitchen.AvailIngr.t - what
    Kitchen.UsedIngr.t' = Kitchen.UsedIngr.t + what
    noComposedChangeExcept[t, t', none]
}
```

Fig. 2. Bake schema and predicate. Bake expects available ingredients and an available and heated item. At the end of the call, the ingredients are used and the tool becomes a dirty item.

the cook operates separately from the tools, `CookDone` is introduced in the same manner. The simple predicate `noComposedChangeExcept` is introduced before the conversion process as it is included in many predicate bodies. We refer to the `Directions` schema to explain the unavailable state of the preparer in the Z version. We again draw attention to the differences between two versions when we include our pre-specified predicate in others. At last, we defer, to later, the discussion about time ordering using a factual statement.

When the general schemas are ready, we specify more concrete events. The first one is `Bake` in Fig. 2. As `Bake` schema is introduced, we underline the differences and similarities with its model. The first one is the `some` multiplicity marking on the ingredient. We say that some important facts like these are not imposed in the Z specification as it serves as an initial draft for our Alloy model. However, the Alloy model is iteratively refined after analyzing the results from the previous steps and hence, is more accurate. When we get to `BakeDone` conversion, one key note is its name, which is `BCKMPDone` in the Alloy version. We say that all `BakeDone`, `CutDone`, `KneadDone`, `MixDone`, and `PutDone` affect the state similarly so they are combined in the implementation. Consequently, those differences related to generalizing it are listed. More specifically, we say that we can only assert that the type of the ingredient produced cannot be

```
fact DoneAfterEvent {
    all t'' : Time - TO/last[], c : Cook | some i, k : KitchenItem, w : Ingredient |
        let t' = TO/prev[t''], t = TO/prev[t'], t''' = TO/next[t''] |
            CookDone[t'', t''', c] ⇒ (Bake[t, t', c, i, w] or Clean[t, t', c]
            or CKMP[t, t', c, i, k, w]  or Preheat[t, t', c, i])
}
```

Fig. 3. DoneAfterEvent fact. CookDone event only takes place after one of Bake, Clean, CKMP, or Preheat events.

Raw. Moreover, HeatedItem is not handled here because different done events act upon it differently. Clean event does not need further explanation as it is very similar to Bake. However, with Cut event we see that in the conversion we combine Cut, Knead, Mix, and Put together under CKMP as they all do same things, too. We do not further discuss Knead, Mix, and Put schemas as they are same with Cut. Our final schema is Preheat. We say that the input to this schema is of type measurement and there is no need to model that in the Alloy as it does not change the state. PreheatDone is also parallel with BCKMPDone.

Later, the schemas related to the event handler in the Z specification are represented with fact statements in the Alloy model. For this, we first recall our deferred discussion on the fact statements that were supposed to constrain the system. We initially write a fact statement called Traces that takes care of the ordering of the states. We summarize that we cannot advance time by one unit at each tick and to circumvent this, we jump from the end time of one task directly to the start time of the next task. We review the body of the fact statement and any new notation that is encountered for the first time. We conclude with the remark that there is still no event handling and any event can take place at any point in time. In that regard, we formalize a guarded implementation. This concept is presented as follows. An event happens only when those events that must precede it already have taken place and those that must succeed it are guaranteed to take place. Thus, instead of depending on the time increment, the system jumps between events. As an example, we show DoneAfterEvent in Fig. 3, which stipulates that CookDone comes after Bake, Clean, CKMP, or Preheat. Similarly, BakeBeforeDone requires that every Bake implies there is a BCKMPDone event that will happen and a CookDone event succeeds it. DoneAfterBCKMP states that BCMKPDone follows Bake or CKMP but it introduces some more constraints, which were omitted beforehand in the predicate definitions, about the relationship between the first and second events. It is pointed out that in the Z implementation, we prepared an extra schema called Directions to record each call and Timer schema keeps track of all the event handling but in Alloy, we use such facts to fill in these details. CleanBeforeDone, PreheatBeforeDone, DoneAfterCP, and CKMPBeforeDone facts are written analogously.

Finally, to convince the student that our model actually satisfies our assertions, we carry out the final step of writing the verbal assertions above formally and checking in the Alloy modeling language. We first show DirtyAndCle- anSame and explain why the check command needs more time steps than any other

instances of sets for this assertion. We emphasize the distinction between under-constraining and over-constraining and that if the range is not assigned properly, the Analyzer will fail to find counterexamples. We note that it is better to be on the safe side and check with the maximum possible range without burdening the tool too much. Subsequently, `PreheatBeforeBake`, `HeatUnheated`, `NoFire`, `UsedNeverAvailable`, `AvailableOnlyUsed`, `RawNonComposed`, and `NonRawComposed` are developed as are the rest of the assertions.

Although the conversion seems straightforward, some complications arise and these are addressed and documented in the tutorial. In this way, we expect the students to be able to understand both the similarities and differences between the two versions so that when they work on their own specifications and models, they will have acquired the necessary background. At the end of the tutorial, we encourage the students to think and write more assertions that makes them feel satisfied with their expectations. Our final Alloy model is fairly long (≈ 500 lines) but we clearly state that once a rough Z specification is available, Alloy development is not very cumbersome.

3.2 Implementation

The implementation of the online tutorial consisted of two phases.

Webpage Design. As can be seen from Fig. 4, we decided to split the tutorial into the following sections, which are shown on the upper left-hand side as a list box. The numbers in the parentheses below show the number of questions in each section, which will be covered in the subsequent paragraph.

(1) Introduction
(2) Birthday Book Specification (20)
(3) Alloy Analyzer Walkthrough
(4) Birthday Book Model (26)
(5) Kitchen Environment
(6) Sets as Signatures (8)
(7) State Specification & Initialization (11)
(8) Events as Predicates (23)
(9) Handler as Facts (7)
(10) Assertions (9)
(11) Conclusion
(12) Questions (4)

The interactivity of the pages is supported as follows. Although Section 3.1 presented as if the tutorial expects the students to be passive readers of the tutorial, this is indeed not the case in our tutorial. According to Kolb's theory, the student takes some information, which gets combined with the existent knowledge of the student. With questions that provoke further reasoning, the educator can immediately act upon the student's understanding to correct the misunderstandings. The reaction of the educator also becomes part of the student's prior knowledge so the educator asks more questions to steer the student's

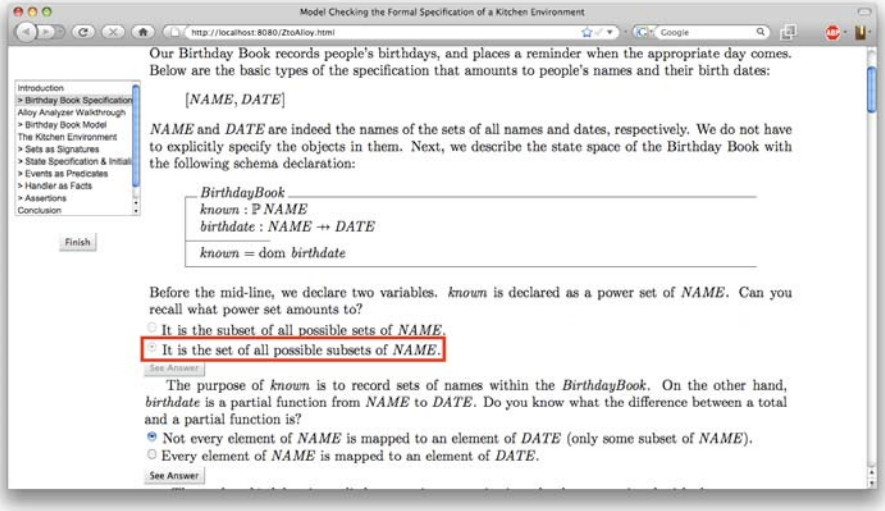


Fig. 4. The Online Interactive Tutorial. The webpage contains teaching material as well as some questions to trigger student's learning. The radio buttons are placed for multiple-choice questions while *See Answer* buttons are used for submitting student's answer and seeing the expected answer. When revealed, the correct answer is highlighted with a red border and the interactivity is disabled. The sections are listed on the left-side panel to enable easy navigation within the document besides the conventional "Prev" and "Next" buttons located at the end of each page. The sections whose names start with a ">" symbol are interactive. When all of the questions within a section are answered, this symbol disappears from the name of that section so that the student knows they have completed the section. The *Finish* button yields the end of the tutorial.

learning. With this theory, we decided to implement multiple-choice questions in the tutorial. Most of these questions would ask the student to think beyond what is given rather than passively reading the text. The students are given the option to reveal the correct answer once they are satisfied with their answer. Consequently, the student has the chance to correct their mistakes further on. Apart from this interactive content, the last section, Questions, is organized as a quiz whose questions are free-fill. Every question contains a line that must be filled in and submitted by the student.

Finally, we embedded two surveys within the tutorial. The pre-survey asks 17 questions on the student's background with Formal Methods and the post-survey with 30 questions asks for feedback. These surveys are prepared using the Likert scale from 1 ("Strongly Disagree") to 7 ("Strongly Agree"), and include an option for the "Don't Know" case. There are also some open-ended questions within these surveys.

Server to Database Connection. Because we wanted to make the tutorial accessible from the Internet through students' computers, we implemented a server component that records client answers in a database and sends the correct answers to the client. The students should be given an identifier and a password by the instructor to access the tutorial. Each student is assigned a user identifier of 5 characters, e.g. 'usr09.'

4 Case Studies with Novice Students

In this section, we report the experiences of our novice students with the tutorial.

4.1 Task and Procedure

In a single session in our lab, the students were asked to read the tutorial and to answer all of the multiple-choice and free-fill type questions on the tutorial. They were also asked to fill out the two surveys. Consequently, they responded to 104 multiple-choice, 4 free-fill, and 47 survey questions in total. Furthermore, the online interface and its functionality were explained before the session. The students were allowed to ask questions during the entire session and also take a break at any time. We used a single computer and the same web browser for all of the sessions. Each student spent between 1 to 3 hours on the tutorial and we accepted verbal comments from the students during or after the tutorial to receive their feedback.

4.2 Student Background

In total we had 8 students (2 females and 6 males) take our tutorial. They were aged between 20 to 30 with an average of 25.75. All of them were Computer Science students with one who had a minor in Mathematics. One of them was a sophomore, two were Master's students, and five were Ph.D. students. They strongly agreed that they have experience with small- and medium-sized software projects (with an average of 6.88/7 and 6.25/7, respectively). They also had an above average (5.38/7) discrete math and logic background. They moderately disagreed that they have strong formal methods background (mean = 3/7 excluding one student who answered they did not know the answer) and they were neutrally interested in formal methods (with a mean of 3.6/7 excluding three students who replied that they don't know the answer). Four of the students who answered the questions thought that formal methods are useful (mean is 4.75/7) and that they are difficult to learn and understand (average is 4/7). Out of the five people who answered the question, the students were willing to make use of formal methods in the future (mean = 5/7). Six students responded that they have seen formal specifications written in English (average = 4.33/7). Seven students strongly agreed that they have not had much experience with formal methods with an average of 6.14/7.

On the open ended questions, the students replied that they use unit testing (e.g. JUnit), smoke testing, nightly builds, code review, trial-and-error, debug printing, incremental development, debuggers (e.g. gdb), running the code, writing test cases, and user studies to detect errors in their code. For their class background, they mentioned an undergraduate level discrete math course or a graduate level programming language or software engineering course. Four people did not report any class background. Only one person indicated that he or she had used SPIN as a formal software verification tool in the past.

4.3 Results

In this section, we report the results from two points of view. First, we present the results in terms of the tutorial's comprehensibility. Next, we report them based on each student's performance.

Tutorial Questions. Here, we talk about the students' answers for the multiple-choice questions. All of these questions would require the student to decide between two answers. At first, we would like to further split our questions into two parts: (i) Birthday Book, and (ii) Kitchen Environment questions.

For the Birthday Book example, there were 46 questions in total. As we introduced the Z specification of the Birthday Book, some initial questions were directed to review the mathematical background of students, but later as students gradually improved their understanding of the notation, more and more questions became related to the specifics of the Birthday Book example. A similar approach was taken for the Birthday Book model. We started asking questions related to the mathematical interpretations of Alloy and as we moved further into the details of the model with a better understanding of the student, we elaborated on the questions. Overall, students did well in the Birthday Book example and we were satisfied that they understood the material. Here we report all of the eight students' results. For the Z specification of the Birthday Book, they answered 17 to 19 questions correctly (on average 18.25/20). The Birthday Book model was well-received, as well. Students gave between 20 and 24 correct answers (mean = 22.25/26) to the questions. Overall for the Birthday Book, their correct answers ranged between 38 and 43, with a mean of 40.5/46.

The Kitchen Environment example contained a total of 58 questions. For this part of the tutorial, we did not ask for any mathematical notation but instead we initially asked questions about the student's recollection of the concepts taught in the Birthday Book. As this example was developed side-by-side with the Z and Alloy versions, we distinguish them according to our conversion technique. When converting sets to signatures, all eight students got 5.88/8 on average the questions right (a minimum of 4 and a maximum of 7). However, when they got to the State Specification & Initialization section of the tutorial, one of the students dropped out of the session because they have not recently studied mathematics and would need to go over the material more than once to truly comprehend it. Therefore, seven students remained to report on this section. Students answered 7 to 10 questions right with the mean being 9/11. By the

end of the section that converts events to predicates, another student decided to stop continuing the tutorial because they were too tired that day. However, they had been able to answer 19 of the questions and got 9 of them right. Excluding this person, six students gave 14 to 21 correct answers on this section and their average was 16.5/23. For the rest of the Kitchen Environment example, we only provide the results of the six remaining students. Their correct answers on the conversion into fact statements was between 4 and 7 with a mean of 5.67/7. These students had 6 to 8 correct answers (with a 7.17/9 average) for the last section that showed Alloy assertions. Excluding the students who could not finish this example, on average, there were 45/58 correct answers (51 at most and 41 the least).

For those six students who were able to complete both of the examples successfully, out of all the multiple-choice questions, they had a mean of 85.5/104 correct answers (93 was the maximum while 79 was the minimum number).

Assessment. The assessment was based on the 4 quiz questions. The questions asked the student to type one line of Alloy code to complete and correct a different model than the ones that are presented throughout the tutorial. The first question asked the student to write the body of a predicate, the second one asked the student to write the body of an assertion while the next one asked for its check statement. The final question tested the student's ability to modify the code so that it excludes ill-formed cases.

As noted in the previous section, because of the students who stopped early, we have only six students whose results we can present. Moreover, some students missed this section as it was after the Conclusion section within the tutorial and they went directly into the post-survey questions. Thus, there were some blank answers. The questions were evaluated based on legitimate rationale rather than exact syntax. Thus, minor syntax errors were disregarded. Based on this grading key, our students got between 0 and 3 answers right with a mean of 2/4.

We also briefly mention that overall, on both the tutorial and the quiz, the correct answers ranged between 80 to 96 (with a mean of 87.5/108) for those six students who finished it.

Student Performance. Instead of reporting each student's performance, we classified the students based on their performance on each section into three clusters as can be seen from Table 1. More specifically, we identified three students whose performance was the best on the tutorial. Additionally, three students did well in answering the questions but not as well as the best. Apart from these, as previously mentioned, two students finished the tutorial with an incomplete status.

As the standard deviations are small, it is obvious that those clustered together have performed similarly. To do a comparison between three different student performances, good students performed slightly worse than the best students in the Birthday Book. On the Kitchen Environment example, this trend seemed to continue. On the Questions section, the correct answers were equal implying that they all learned the concepts. The totals indicate that above 90

Table 1. The summary of the results for those students who achieved best performance, and who did well, and those who could not complete the tutorial

	Best Performance		Good Performance		Incomplete Tutorial	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
Birthday Book specification	18.67	0.58	17.33	0.58	19	0
Birthday Book model	23.33	1.15	21.67	0.58	21.5	2.12
Birthday Book (total)	42	1	39	1	40.5	2.12
Sets as signatures	6.67	0.58	6	0	4.5	0.71
State specification & initialization	9.67	0.58	9	1	7	N/A
Events as predicates	18	2.65	15	1	9	N/A
Handler as facts	6.33	0.58	5	1	N/A	–
Assertions	7.33	1.15	7	1	N/A	–
Kitchen Environment (total)	48	2.65	42	1	12.5	–
Birthday Book & Kitchen Environment	90	2.65	81	2	53	10.6
Questions	2	1.73	2	1	N/A	N/A
Total	92	3.61	83	3	53	8.49

is the best, while above 80 is good. An interesting result comes out for the incomplete group. They did better in the Birthday Book example than the good students, however, because they were not able to continue, we do not know how they would have performed in the Kitchen Environment example. We believe that they could in fact perform better again giving us better results for our tutorial.

4.4 Student Feedback

The post-survey asked for each student's feedback. However, we have only records for seven of the students because one quit before filling it out. Also, some students answered some of the questions as they don't know the answer and these were not counted. All the numbers reported in this section range from 1 to 7 (the Likert scale).

First of all, we had questions related to Formal Methods, i.e. the appreciation of their benefits, confidence in proficiency, credence to automatic verification tools. For these, students moderately agreed and gave 4.5, 3.29, and 4.57 points on average, respectively. Next, we asked questions about the student's opinion about formal specification (their comfort with the Z language and ability in writing Z specifications) and students were neutral (with means of 4.86 and 3.14, respectively). We also directed questions on model checking – learning of model checking, usefulness, competency in Alloy, ease-of-use for the Alloy Analyzer. Students moderately appreciated model checking (with means for learning as 5, usefulness as 4.57, and ease-of-use as 3.67) but they did not feel competent enough yet (mean = 2.86).

Apart from these, there were questions on the Z to Alloy transition. More specifically, we asked students about the ease of learning Alloy after Z, whether Alloy is similar to Z, their ability to convert Z to Alloy, the straightforwardness of the Z to Alloy conversion, their preference to write Z beforehand, and the favorability of completely transitioning to Alloy. Students strongly agreed with our statements on the ease of transitioning from Z to Alloy with the tutorial (receiving 4.57, 5.29, 4.43, 4.86, 4.33, and 5.8 averages, respectively).

Finally, there were questions about the teaching material. We asked whether it was a meaningful (5), efficient and effective (5.86), and motivating material (5.14), whether its organization facilitated learning (4.71), whether it included enough math and logic background (5.67), the appropriateness of each example's level of complexity (4.29) and mental effort (4.71), and interest (4). The students responded that they agree with these statements as shown with the means given in the parentheses. Moreover, some questions asked if they knew similar examples as the Birthday Book (2), the comprehensiveness of the Kitchen Environment example (6.14) and its presentation's level of detail (5), and the success of the conversion (5.14). As the averages between the parentheses state, students agreed that Kitchen Environment example was good even though they have not been exposed to simple examples like the Birthday Book. Finally, we directed questions at the student's ability to design their own specifications and model check them (3.57), their satisfaction with the tutorial (4.71). The averages indicate that the students gained some moderate capability at model checking and that they were satisfied.

At the end of the post-survey, we expected them to share what they liked/disliked and their opinions on how to improve the tutorial with a last open-ended question. A lot of ideas were related to the web layout of the tutorial.

– "i had to do too much scrolling."
– "Questions about figures should come after the figures themselves."
– "1. make the listbox larger 2. put prev next below the listbox. . . "
– "the figures for the questions were sometimes far beneath the questions themselves."
– "Referring to multiple models on other pages was distracting – especially since the page did not either reset the scroll bar or return to where you were when you previously viewed that page."

Moreover, some students stated both in their verbal comments and in the post-survey that they wanted visualizations (pictures, diagrams, and tables).

– "I would have liked to see a better comparison of English/Z/Alloy – maybe a few figures where they were compared and explained step by step next to each other for full effect. . . I think a more succinct table of some of the Alloy examples would have helped me understand better."

Some of the students said that they needed more intermediate examples.

– "kitchen example was very complex and maybe a too big big step from the very simple birthday book."

– "The kitchen example lost me halfway through. The birthday book was simple and made sense; the kitchen example introduced too much too quickly. It was well written and presented, but was too difficult for me to quickly grasp. More intermediate examples should have been presented."

There were cases when the students thought that one tutorial was not enough for them.

– "I felt the need of a some math and logic background. Moreover, I don't think it was easy to grasp all the definitions in the first go."

Some students found the (English) language in the tutorial hard to understand.

– "The language in the tutorial is hard to understand in several places."
– "Some of the Alloy conventions were hard to understand."
– "I felt that the question choices often stated two sides of the same thing – perhaps I did not quite understand the intent of the statements. While the step-by-step conversion process was nice the grammatical structure of some of the sentences throughout this process seemed confusing to me."

Although they stated that they liked the use of everyday examples, they complained that just examples are not enough.

– "trying to explain the importance of formal methods using simple day-to-day examples was quite motivating."
– "I feel that this tutorial brought about an appreciation for model-checking in a roundabout way; rather than coherently explaining the process it seemed to jump immediately into examples and never really get into a broad understanding of the topic."

In their verbal comments, one student stated that although he/she thinks it is certainly useful to perform model checking, it is expensive and onerous to write for his/her own projects. Similarly, another student commented that he/she does not intend to make use of model checking, because as researchers, they are satisfied with software that is "good enough." However, both of them agreed that they clearly appreciate the necessity in critical aspects (security, concurrency, etc.) of a big software system.

5 Discussion

In this section, we share what we learned from our studies and what could have been improved and in what way.

Generally, speaking we are encouraged with the results. Given the weak background of the students on formal methods and their lack of interest in the formal methods topics, the results show that the tutorial brought some appreciation for formal specification, model checking, and verification tools. Looking at the survey responses, we are also pleased that the students thought Alloy is similar

to Z and that they appreciated our conversion method in compliance with our expectations. However, because none of them had prior Z background, they were inclined to like Alloy more than Z as it enables them to directly jump to programming. Our Kitchen Environment example was comprehensive but compared to the simple Birthday Book, the student performance did not show a huge drop. This can be seen from the percentages of correct answer means for the Kitchen Environment (77.5%) and the Birthday Book (88%). It is clear that students were able to comprehend even such a complicated example.

What we learned from this study is that students do seem to appreciate paper specifications but they want to be able to practice and correct their designs using automatic tools. In that respect, they believe that Z and Alloy are similar and can be taught together. However, there needs to be some appropriate material in this field. The available textbooks teach Z or Alloy but not both and most of them only use toy examples of 10 to 50 lines. We tried to approach this problem by making use of a conversion technique to facilitate Z to Alloy transition with a comprehensive example; however, students needed well-documented intermediary examples, too. Compared to most of the textbooks we encountered, our complex example was 500 lines. The students surely felt that this was a huge gap that needs to be filled in by the instructors. Finally, they liked the idea of real-world examples but requested pictures to make them more appealing. Educators should take this into account when designing their materials.

One downside of the web component was that it was offered as one large tutorial instead of being broken up over two or more lectures. This was the reason for some students feeling too overwhelmed to finish it in a single session. There were also some problems with the online version of the tutorial that students candidly addressed in their complaints about the interface. Even with such problems, students did not criticize the content of the tutorial. This suggests that they were satisfied that they learned something new.

Moreover, our student pool was not representative of real software developers. We mostly conducted studies with graduate students, who do not often implement extensive software projects. Moreover, these students had not studied discrete math and logic in several years, and their interests were specialized in other areas of the computer science. This gave rise to issues in recalling the concepts. Since the studies were conducted at the end of the semester and during the final exam week, we could not attract many undergraduate students to participate. The long duration of the study and lack of compensation deterred some students who were initially interested in the study. Some of the students who took our tutorial did not take the questions on the quiz seriously enough nor did they complete all sections. However, looking at the in-tutorial answers, we are satisfied that participants did put enough effort into reading and answering the questions as none of them had correct answer rates below 70%. In terms of time, it took more than an hour for each student to complete the tutorial. Also, due to time limitations, we could not run enough pilot studies to correct the implementation errors and thus, some answers got unrecorded.

Finally, we allowed students to have access to the entire content throughout the session and did not save times along with the answers. Had this been implemented in our tutorial, we think we could have revealed some common patterns in the learning of Z to Alloy that could be useful for instructors.

6 Conclusion

In this paper, we first presented our online tutorial to teach Alloy with Z using the Kitchen Environment real-world example. In order to do that, we presented our conversion method that is expected to simplify this process. We also shared our experiences with novice students using our tutorial.

According to our findings, we recommend that educators who teach the Z language also focus on the Alloy tool in their classes. In addition, it is necessary to provide well-explained and interesting intermediate-level examples for the Z to Alloy transition. Our tutorial provides both a concrete Z-to-Alloy transition process and the intermediate-level Kitchen Environment example. Initial experience with novices suggests that students are able to learn Z and Alloy and answer questions about an intermediate-sized example after only a couple hours of study. Interested readers may access this tutorial through our website [28].

References

1. Alloy Analyzer 4, `http://alloy.mit.edu/alloy4/` (2009-07-15)
2. Alloy FAQ, `http://alloy.mit.edu/faq.php` (2009-07-15)
3. Brakman, H., Driessen, V., Kavuma, J., Bijvank, L.N., Vermolen, S.: Supporting Formal Method Teaching with Real-Life Protocols. In: Formal Methods in the Teaching Lab: Examples, Cases, Assignments and Projects Enhancing Formal Methods Education, pp. 59–68. McMaster University, Canada (2006)
4. Broda, K., Ma, J., Sinnadurai, G., Summers, A.: Friendly e-tutor for Natural Deduction. In: TFM: Practice and Experience. BCS-FACS, London (2006)
5. Dean, N.: Development of an Interactive Case e-Study. In: TFM: Practice and Experience Workshop, pp. 13–20. BCS-FACS, Oxford Brookes University (2003)
6. deBry, R.: Learning exercises for the rest of the brain. J. Comput. Small Coll. 20(1), 291–296 (2004)
7. Duke, R., Miller, T., Strooper, P.: Integrating Formal Specification and Software Verification and Validation. In: Dean, C.N., Boute, R.T. (eds.) TFM 2004. LNCS, vol. 3294, pp. 124–139. Springer, Heidelberg (2004)
8. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press, Cambridge (2006)
9. Jonathan, J.: The way of Z: Practical programming with formal methods. Cambridge University Press, NY (1996)
10. Kolb, D.A.: Experiential learning: Experience as the source of learning and development. Prentice-Hall, Englewood Cliffs (1984)
11. Larsen, P.G.: Two courses on VDM++ for Embedded Systems: Learning by Doing. In: Formal Methods in the Teaching Lab: Examples, Cases, Assignments and Projects Enhancing Formal Methods Education, Canada, pp. 21–26 (2006)

12. Lightfoot, D.: Voici les votes! – formal specification as light entertainment: An example of audience participation in developing a specification. In: TFM: Practice and Experience Workshop, pp. 71–75. BCS-FACS, Oxford Brookes U. (2003)
13. Mandrioli, D.: Advertising Formal Methods and Organizing Their Teaching: *Yes, but...* In: Dean, C.N., Boute, R.T. (eds.) TFM 2004. LNCS, vol. 3294, pp. 214–224. Springer, Heidelberg (2004)
14. Martin, J.M.: Teaching Formal Methods: An Industrial Perspective. In: TFM: Practice and Experience Workshop, pp. 35–39. BCS-FACS, Oxford Brookes U. (2003)
15. Montessori, M.: The Montessori Method, Schocken (1988)
16. Pepper, P.: Distributed Teaching of Formal Methods. In: Dean, C.N., Boute, R.T. (eds.) TFM 2004. LNCS, vol. 3294, pp. 140–152. Springer, Heidelberg (2004)
17. Piaget, J., Mays, W., Beth, E.W.: Mathematical Epistemology and Psychology. D. Reidel Publishing Company, Dordrecht-Netherlands (1966)
18. Piaget, J., Garcia, R.: Psychogenesis and the History of Sciences. Columbia University Press, New York (1980)
19. Reed, J.N., Sinclair, J.E.: Motivating Study of Formal Methods in the Classroom. In: Dean, C.N., Boute, R.T. (eds.) TFM 2004. LNCS, vol. 3294, pp. 32–46. Springer, Heidelberg (2004)
20. Rosa, S.: Designing Algorithms in High School Mathematics. In: Dean, C.N., Boute, R.T. (eds.) TFM 2004. LNCS, vol. 3294, pp. 17–31. Springer, Heidelberg (2004)
21. Rudall, J.: From Z to SPIN in One Module. In: TFM: Practice and Experience Workshop, pp. 71–75. BCS-FACS, Oxford Brookes U. (2003)
22. Spivey, J.M.: The Z Notation: A Reference Manual, 2nd edn. Prentice-Hall International, UK (1992)
23. Tarkan, S.: The Formal Specification of a Kitchen Environment. Master's scholarly paper, University of Maryland (2009)
24. TEXto HTML translator, `http://hutchinson.belmont.ma.us/tth/` (2009-07-15)
25. The Promela Language, `http://en.wikipedia.org/wiki/Promela` (2009-07-15)
26. The SPIN Model Checker, `http://spinroot.com/` (2009-07-15)
27. Woodcock, J., Davies, J.: Using Z: specification, refinement, and proof. Prentice-Hall, Upper Saddle River (1996)
28. Z to Alloy Tutorial, `http://ztoalloy.cs.umd.edu/` (2009-07-15)