Increasing Engagement with Interactive Visualization: Formal Methods as Serious Games

Eduard Kamburjan[1] and Lukas Grätz[2(✉)]

[1] University of Oslo, Oslo, Norway
eduard@ifi.uio.no
[2] Technische Universität Darmstadt, Darmstadt, Germany
lukas.graetz@tu-darmstadt.de

Abstract. We present a concept to increase the interactivity of formal methods courses. To do so, we discuss how formal methods can be seen as special serious games—a set of systems that is applied successfully in other educational contexts. To close the gap between the presented theory and its formalization or implementation, we take results from interactive visualization to develop a tool that empowers the students to deepen their knowledge about the presented theory in the same terms that are used in the lecture. The concept is not only based on experiences of the formal methods community, but also on studies and theories in the educational sciences. An implementation that is used in the exercise sessions of a course teaching proof calculi is available under https://kbar.app.

1 Introduction

Motivation. Teaching methods for formal methods have received renewed attention in recent years. Several workshops [6,9] and surveys [31,34] have identified numerous subject-specific challenges. One of the identified challenges is that formal methods tools are not designed to be used for teaching: they have a steep learning curve and give little feedback [34]. As such, they are at most useful to teach how to *solve* a problem using the formal method but give little support to teach their internal *concepts*. Two further challenges worsen the situation: For one, formal methods are dry and many students feel intimidated by mathematical expressions to the point of mathematical anxiety [28]. For another, the presentation of formal methods in lectures, their implementation and their presentation in textbooks is highly non-uniform.

It is not possible to use the tool to manipulate and explore the presented material using the concepts *as presented in the lectures*. This results in low student engagement in formal method courses, as the gap between tool and concepts discourages the student.

Overview. In this work, we use the structural similarities between formal methods and serious games to develop a concept to increase student engagement

J. F. Ferreira et al. (Eds.): FMTea 2021, LNCS 13122, pp. 43–59, 2021.
https://doi.org/10.1007/978-3-030-91550-6_4

in formal methods courses. Serious games have been shown to have a positive effect on student engagement [32] and our concept aims to carry over this effect to formal methods. To see a formal method as a serious game, one needs a visualization with the following characteristics: (i) It is interactive. (ii) It visualizes the formal methods consistently with the concepts and notations in the theory part of the course.

The added value of interactive visualization is that it communicates the mental model of the expert more faithfully [18]. In contrast, formal method tools that focus on applications, need a different mental model, and thus, fail to increase student engagement. The interactivity here is crucial: non-interactive visualizations have only a small effect on student performance [19]. Note that in this concept the formal method itself is the serious game; it is *neither* an application of a formal method to games *nor* a gamification of existing tools.

This work discusses a teaching concept on how the view on formal methods as serious games can be used to develop and apply an interactive visualization tool in a formal method course. In our concept, the interactive visualization tool is used (a) as a self-study help for students and (b) in the exercise session of a course. In early learning stages, a student internalizes new concepts. Here, using the tool as a self-study can help students to explore the concepts on their own terms and on their own speed. As the tool catches mistakes and provides feedback why certain operations are not applicable, students do not reinforce misconceptions by reapplying mistakes. In contrast to pen-and-paper exercises, where they would have to wait several days for the teacher's feedback.

Proof Calculi. We present our approach to develop a tool for teaching proof calculi, e.g., different tableaux variants, based on the authors' experience with the exercises of a course on automatic theorem proving. Consistently with the identified challenges, we observe that it is challenging for students to adopt the taught calculi to prove something on new examples. We conjecture that one of the reasons, for example in tableaux methods, is that the lecture presents the proof calculus visualized as a tree, creating a cognitive gap to the formalization (where the tree is implicit behind mathematical notation) and the implementations.

We present the `KalkulierbaR` tool, which allows the student to build and change proofs in several variants of tableaux, resolution, DPLL and sequent calculi. The tool is a serious game, similar to puzzle games, where the proof rules are possible steps in the game. The goal is to close the proof.

Contributions. The main contributions of this work are (a) a concept to view formal methods as serious games to teach formal method theory, and (b) the `KalkulierbaR` tool, designed to be part of this concept to teach several proof calculi. The tool is designed by the authors, the implementation of `KalkulierbaR` was done together with two groups of students, which were implementing user stories given by the authors as part of a mandatory lab. The format is explained in detail in Sec. 4. It includes several variants of tableau, resolution, DPLL and the sequent calculus. A live instance runs under https://kbar.app and the source is available under https://github.com/kalkulierbar/kalkulierbar.

Structure. In Sect. 2 we present background on serious games and interactive visualization in education, as well as challenges specific to formal methods and related work. Sect. 3 applies these ideas for formal methods to present `KalkulierbaR`. Sect. 4 discusses our experiences and Sect. 5 concludes.

2 Background and Related Work

We discuss the background for serious games, interactive visualizations and student engagement from a formal method perspective.

We follow Roggenbach et al. [29] and understand a "formal method" as (a) a set of syntax with some (b) semantics and (c) a set of rules operating on the syntax. This includes logics with proof calculi. Under a logic we understand an abstract logic [14], i.e., a triple of (a) a set of sentences as syntax, (b) a set of models and a satisfiability relation with certain properties as semantics, and (c) a proof calculus that consists of operations on the syntax as rules.

Interactive Visualizations. Algorithm and program visualization has a long history in computer science education and is documented dating back to the beginning of the 1980s[1]. It seems intuitive that visualizations help to engage the student and increase the accessibility of taught material, and several theories on why this is indeed the case have been put forward. For example, the epistemic fidelity theory of Hundhausen argues that algorithm visualizations "*provide a faithful account (i.e., one with high epistemic fidelity) of an algorithm's execution in terms of an algorithm expert's mental model*" [18].

However, a meta-study of Hundhausen et al. [19] recognizes that providing algorithm visualization tools alone does *not* increase student performance:

> "*With few exceptions, we found that studies in which students merely viewed visualizations did not demonstrate significant learning advantages over students who used conventional learning materials*".

Instead, increased student performance can be attributed to *interactive* elements of the visualization tool, such as constructing own input sets, programming tasks, answering questions and building own visualizations [24]. Hundhausen et al. also find that tools designed for active learning and based on *cognitive constructivism* have more significant results in increasing student performance. Naps et al. argue that "*visualization technology, no matter how well it is designed, is of little educational value unless it engages learners in an active learning activity* [24]". They present an engagement taxonomy of possible involvement: (1) no viewing, (2) viewing, (3) responding, (4) changing, (5) constructing and (6) presenting. Subsequently, this taxonomy was extended by Myller et al. [23] to include more fine-grained levels.

We observe that formal methods are easy to adapt to interactive visualization for multiple forms of involvement in the engagement taxonomy (2), (3) and (4).

[1] The most prominent artifact from these early approaches is the short film Sorting Out Sorting [4]. For a historic overview, we refer to Baecker [5].

There is a major difference between the algorithms taught in typical algorithm and data structure courses, which form the basis for the above studies, and formal methods: Many formal methods are non-deterministic (without a strategy for rule selection) and as such enable even more interaction than algorithm visualization. However, we stress that the reason for increased student performance is that the visualization is used to provide and manipulate a certain mental model that is taught in the lecture. As such, application-oriented tools are not suitable for this task—there is a gap between how a method is explained in its pure form and how it is implemented. Even systems with a focus on user interactions, such as the KeY tool [1], work on an extension of the pure method and for novices, it is not easy to distinguish the parts which stem from the pure method and which stem from application needs.

Learning with Serious Games. A similar approach to interaction in learning are (educational) serious games. There is no agreement on the exact definition of a serious game[2], and for the rest of this paper, we commit to using the one put forward by Wouters et al. [32]: interactive systems based on a set of agreed rules and constraints, directed toward a goal, which provide feedback to the player to enable monitoring the progress towards the goal.

Serious games are *not* necessarily about gamification: Serious games are game-like programs for education, while gamification is about game-like mechanics in non-game contexts [21], e.g., awards for participation in online discussions. Nonetheless, serious games must be embedded in context: a meta-study of Wouters et al. [32] concludes that while serious games can lead to better learning and retention, the students do *not* feel more motivated by them.

We discuss the exact connection with serious games in Sect. 3.1. For now, it suffices to remark that we can interpret the set of syntactic rules (of the formal method) as the goal-directed rules of a serious game. The goal is, for example, to close a proof. For a formal method to become a serious game, we have to add visualization, interaction, and feedback.

Related Work in Formal Methods. Cerone et al. [7] discuss specific challenges for teaching formal methods based on a recent workshop. While most of their discussion focuses on the role of formal methods in a curriculum for software engineers, some of the authors also mention the role of games in their teaching. However, they use games as case studies to introduce formal methods, e.g., to have an intuitive set of rules that needs to be modeled *using* the formal method. Tools developed on this idea, like the `FormalZ` tool of Prasetya et al. [27], are successfully applying gamification to the course, but do not see the formal methods as the serious game itself.

Another raised point is that the tools are not suitable due to confusing error messages and interface. This coincides with the epistemic fidelity theory: these tools do not provide a faithful account of the mental representation of the *teacher*

[2] Defining a game is notoriously challenging. For a recent survey from the view of electronic games, we refer to Arjoranta [2].

Table 1. Selected tools for proof calculi and criteria for serious games.

Tool	Rules	Interactive	Visual	Feedback
Sequent calculus trainer [10,11]	Sequent calculus	✓	✓	✓
Panda [15]	Natural deduction	✓	✓	?
Easyprove [22]	Pen-and-paper math	✓	×	✓
CalcCheck [20]	Pen-and-paper math	Partial	×	✓
WinKE [8,12]	Tableaux calculus	✓	✓	?
`KalkulierbaR`	Multiple calculi	✓	✓	✓

for theoretical concepts, as they are based on the mental representation used for their *application by experts*. Farell and Wu discuss in a recent experience report [13] also the problems of relating tools and theory due to the disconnect of concepts as taught in the course and concepts as used in the tool. We see their experiences as representative for several studies reported in the aforementioned white paper of Cerone et al. [7] as well as other surveys [31,34], which also identify a lack of tool support and visualization as challenges for formal methods.

There are numerous visual interfaces for single proof calculi (e.g., [8,10,15, 22]). Additionally, we present `KalkulierbaR` in Sect. 3, a tool that covers multiple calculi and is explicitly designed following the didactical theories behind interactive visualization and serious games. Not all of these tools support all features of a serious game, see Table 1. For example, CalcCheck and Easyprove were intentionally designed for text-based math proofs and not for visual proofs. The Sequent Calculus Trainer is an interactive visualization of the sequent calculus, following a didactical motivation [11]. Since it also provides interactive feedback, it fits our perception of a serious game—although not explicitly designed as a such.

As we will discuss in the next section, interactive provers can be seen as serious games, but not necessarily as *interactive* visualizations. This is, for example, the case for Isabelle and Coq, which have a *textual* interface. Textual interfaces are suitable for different teaching approaches. For example, the CalcCheck tool [20] aims to give an interface that is as near as possible to the notation (and language) in the used textbook. Similar ideas are used by Pierce et al. [26] in a series of books that are executable Coq scripts. The exercises in these books can be seen as serious games (without interactive visualization), but this connection is not made explicit.

Note that we focus here on the aspect of teaching concepts – powerful tools are still necessary in teaching if formal methods are taught in an applied program. Ölveczky [25] takes a different approach to this and argues that Maude is a tool that can bridge the gap between theory and application in one tool, as its formalism, a rewriting logic, is similar to functional programming. As such, Maude relies on prior knowledge of the student with another similar formalism, which is not possible for no-programming based formal methods. It is an example of a formal method where the gap between concepts and tool is small, in this work we focus on teaching methods for formal methods with a bigger gap.

3 Proof Calculi as Serious Games

Based on the principles discussed in the previous section, we suggest teaching methods to assist lectures, exercises and self-study with interactive visualization. We present `KalkulierbaR`, an implementation of this concept for proof calculi.

3.1 Formal Methods Are Serious Games

Our key observation is that formal methods are serious games, if an interactive visualization is provided. We remind that a serious game is an interactive system with a set of rules and constraints directed towards a certain goal, that provides feedback to monitor the progress towards the goal.

For formal methods, there is a set of agreed rules and constraints (operating on syntax) and a clear goal (closing the proof or reaching/avoiding some state). For example, for proof calculi, the goal of the game is to close the proof using a fixed set of proof rules. For, say, model checking of liveness properties in automata, the goal is to find a path to some location. Compared to other games, they are near to puzzle or tile-matching games.[3] Most notably, formal methods are single-player games and do not have an opponent. While they have a clear winning condition (reaching the goal) they do not necessarily have a clear losing condition. Such a condition is not necessary for a system to be a game.

As we see, formal methods merely lack interaction and feedback. This must be provided by an interactive visualization of the formal method. This means that even without gamification efforts, formal methods with interactive visualizations are serious games. As we have seen in the previous section, serious games and interactive visualization do not automatically increase student engagement or retention. Instead, they must be integrated into the course in a way that they are similar to the mental representation of taught concepts. We give a concept to do so in the rest of this section.

3.2 Teaching Methods

Our concept is to use an interactive visualization tool in a formal method course as a consistent help for the student. At the core, the tool allows the student to work with a formal method as it is taught in the lecture and textbooks. In particular, we aim to use the same syntax and visualizations for rules as already given in the lecture. Before we introduce our implementation, we describe where and how the tool is used in the course, with proof calculi as a guiding example.

We assume a tool that (1) visualizes the current state of a formal method (e.g., a proof tree or program configuration) (2) permits selection of parts of the state (e.g., a proof node), (3) displays a list of possible rules to apply and (4) provides detailed feedback if a rule is selected that cannot be applied.

[3] For instance, Sudoku, Tetris or Candy Crush. The similarity to such puzzles goes beyond the definition: Candy Crush and other three-matching games have been shown to be NP-complete despite their simple rule sets [16].

While optional, we also assume that the tool (5) permits reverting steps for backtracking.

Lectures. The tool can be used to increase the interactivity of a lecture by executing the taught formal method in the plenum. For example, a teacher performs a proof but asks the students how to proceed in each step. A student can answer with a rule and a proof node. In an online live session, the answers can be typed in a chat. As a variant, for lectures with 20 students and more, anonymous polling software could be used to find the next step by majority vote.

Advantages of using an interactive proof tool in the lecture:

Flexibility. Contrary to slides, a teacher can react to unforeseen answers of the students and explore alternative strategies suggested by the students. Contrary to (digital) blackboards, we do not need to reserve space beforehand: Scaling and arranging a proof (tree) is done automatically by the software. Similarly, it is less time-consuming to perform proof steps, further facilitated by reverting or undoing steps. Student answers also provide valuable feedback for the teacher to assess the learning progress.

Student Engagement. The tool fosters category 3 in the engagement taxonomy (responding), as it can be used to ask the students questions (e.g., "what will be the result of this step"). Tool support also helps using category 4 (changing), where the students can influence the next steps of the formal method.[4]

Low Threshold for Participation. The method using an interactive proof tool lowers the participation threshold in multiple ways: Since students also have access to the tool, they can protect themselves before answering by performing the steps beforehand. Furthermore, students (otherwise anxious to participate) are activated, since they only need to follow the rules of a calculus. The answers are often a few letters only, which could be typed (depending on the lecture format) in an anonymous chat.

Reproducibility. Students can reproduce proofs using the same tool.

Self-study. After an aspect of a formal method is introduced in the lecture, students not only have the possibility to reproduce the examples, but also to explore different strategies and examples on the students' own time and terms. In our proof calculus setting, a student selects a node in the proof and an appropriate rule. The proof is drawn by the software and feedback is given, whenever the student tries to apply a rule in a wrong way. To achieve the final goal, a specific order of rule applications may be necessary: By using *backtracking* in the form of undoing proof steps, the student may fix this order.

In particular, the student can (a) stay within the conceptualization introduced in the lecture, (b) get *step-wise feedback* on erroneous input (e.g., trying to apply a rule that does not match the situation) in terms of the very same conceptualization and automatically. It is neither necessary to learn new syntax

[4] In terms of AV, changing is mostly used to provide new inputs to the algorithms. Formal methods have more flexibility in possible input.

or different visualizations as used by tools aimed for applications, nor necessary to connect error messages from application tools to the basic concepts of the formal method. Ehle et al. [10] have observed that syntactically wrong rule instantiations are a major source of errors.

Exercises. Exercises come with a variety of different tasks and are an essential part in learning formal methods. In exercise/lab sessions, the student is given multiple tasks to apply the content of the lecture. In the first parts of the course, this is done on a conceptual level. Some tasks can be solved interactively in a calculus as described above. For example, when we give students a formula that needs to be proven using a calculus. The advantages of interactive visualization in this context are the same as in the lecture setting described above: Both pen-and-paper solutions and interactive visualization solve the same problem, but by using the software, the students benefit from step-wise and early feedback.

Interactive visualization is mainly suited for exercises on reasoning *within* a formal method (formal proofs in a calculus, algorithmic proof procedures, etc.). Exercises on meta-level properties *of* a formal method do not benefit directly (like soundness of a calculus).

Labs. There is a difference between using and extending a tool. Students benefit from both. On one hand, *using* allows one to perform steps in the formal method with step-wise feedback, as described above. On the other hand, *extending* engages students even further: Category 5 of the engagement taxonomy is to construct new visualizations. We suggest the following labs:

Strategies. We assume a formal method to be non-deterministic. However, it likely has a reasonable strategy for rule selection. Such strategies can be implemented in the interactive visualization tool. Additionally, it may be used to visualize auxiliary structures used for the strategies (such as "set of support" in resolution).

Extensions. An interactive visualization tool could be a basis for the implementation of additional rules or alternative rule sets. Extensions include the calculus itself, the syntax, or the graphical view.

Exams. Interactive visualization tools have limited applications in exam situations. Students are generally expected to perform pen-and-paper proofs by themselves and need to demonstrate that they understood the system *without* a program to guide them. However, students certainly benefit from the interactive visualization tool when preparing for the exam by self-study, as described above.

There are exam situations when pen-and-paper are difficult to organize, e.g., in (open-book) online written exams or online oral exams. In these situations, students may use the interactive proof tool. It should be carefully monitored that (1) such exams are still being fair to all students and (2) students' performance is independent and not bound to a particular proof assistance tool. Furthermore, we should respect students' privacy and the in-homogeneity of environment and infrastructure on the students side.

Table 2. Calculi implemented in `KalkulierbaR`, listed with supported variants and logics. PL denotes propositional logic, FO denotes first-order logic.

Calculus	Variants	Logics	Model generation
(Clausal) tableaux	Regular, (strongly) connected	PL, FO	×
Non-clausal tableaux		FO, Modal	×
Resolution	Hyper resolution	PL, FO	×
Sequent calculus		PL, FO	×
DPLL		PL	✓

3.3 `KalkulierbaR`

`KalkulierbaR` is an interactive formal proof tool designed to support teaching proof calculi following the concept described above. We refrain to introduce the calculi in detail and instead illustrate the usage of `KalkulierbaR` by example. Table 2 shows the implemented calculi and their variants. `KalkulierbaR` is implemented as a web application, consisting of a frontend (written in JavaScript) for the interface and a backend (written in Kotlin) for the state.

`KalkulierbaR` uses a responsive design and can be used with a touchscreen (on a smartphone) as well as with a pointer device (desktop or laptop). Thus, we rely only on standard web technology and no installation is necessary, reducing the threshold for the student to use the tool. The backend can be installed locally and includes a build system that downloads all dependencies.

Overview. Initially, the students select a calculus with a suitable logic and set the parameters, for example, the weakly connected tableaux variant. Additionally, there are some additional settings, such as backtracking: the ability to undo steps. Then a formula is entered (a clause-set or a sequent, depending on the chosen logic and calculus). At this point, one can set optional parameters. Some parameters control details of the calculus, like restricting tableaux to weakly connected tableaux, while others influence the overall workflow, such as backtracking. In the serious game view, this corresponds to adjusting the rule set.

The input is then parsed and sent to the backend server, where logic and calculi are implemented. If parsing fails, error messages are provided. Before the proof is started, the backend server might perform some normalization steps.

Once the calculus is selected and a formula is entered, one can start playing by applying rules of the respective calculus. When a proof is shown for the first time, a tutorial appears. The tutorial can be reopened using the `help` button on the screen. Usually, deduction steps in the calculi consist of selecting one or two formulas and a rule to apply. The rules are either in the lower right or on the left. Sometimes, a rule might request additional parameters in a pop-up window.

Once a proof is finished we can use the `check` button to verify our result. Properties of the proof are displayed in the message box.

Example. We give an example to demonstrate `KalkulierbaR`. Our example is in spirit of Smullyan [30], Chapter XIV, and gives students a more motivating

exercise then an abstract formula. It also demonstrates the full range of the exercise, from situation, over logical modeling to the use of calculi for solving.

> "Once, a parent went grocery shopping with their child. Most notably, they bought yogurt and a chocolate egg. The parent says: 'My dearest child, you have done so well today. I would like to reward you, only you have to say the truth in whatever statement. If your statement is true, then I will reward you with the yogurt, but if the statement is false, I will not give it to you.' Now it so happened that the child wanted to have the egg and not the yogurt! The child is clever and makes the following statement: 'You will give me neither the egg nor the yogurt"'.

This statement forces the parent to give the egg to his child, as only this outcome makes the demanded statement by the child false without breaking the promise of the parent. We could check that this produces the desired outcome by examining all possible combinations manually. But by applying formal methods, we can do this more systematically.

First, we need to formalize the statement in logic. Our formalization is given in Table 3 and uses two propositional variables e and y.

Table 3. Formalization of Smullyan's puzzle.

Proposition	Meaning
y	"The parent gives the yogurt to the child"
e	"The parent gives the chocolate egg to the child"
$\neg(y \vee e)$	Statement by the child: "You will not give me neither the yogurt nor the egg"
$\neg(y \vee e) \rightarrow y$	First proposition by the parent: "If the statement is true, then I will give you the yogurt"
$\neg\neg(y \vee e) \rightarrow \neg y$	Second proposition by the parent: "If the statement is false, then I will not give it to you"

There are two parts in the verification of the puzzle's solutions. We show that the desired conclusion follows, assuming the parent keeps to true to their word. Furthermore, we have to show consistency of the premises, i.e., we check that it is possible to actually keep the parents' word. We solve the first part using a proof calculus and the second part by checking a satisfiable model.

We show that the parent gives the yogurt to the child, whenever both propositions by the parent are true. To do so, we use the sequent calculus, where premises are on the left and the conclusion on the right of the turnstile $\vdash$:

$$\neg(y \vee e) \rightarrow y,\ \neg\neg(y \vee e) \rightarrow \neg y \ \vdash\ e \tag{1}$$

We may enter the sequent in the respective ASCII-notation, as shown on the left in Fig. 1. Once we start the proof, the input sequent is displayed in the

usual notation at the bottom. Now we can try applying rules like "impLeft" on $\neg\neg(y \vee e) \rightarrow \neg y$. The proof tree after this step is shown at the top of Fig. 2. Whenever the student tries to apply a rule on a formula where it is not possible, the exact cause is displayed. There are multiple ways to finish the sequent proof, one of them is shown at the bottom of Fig. 2.

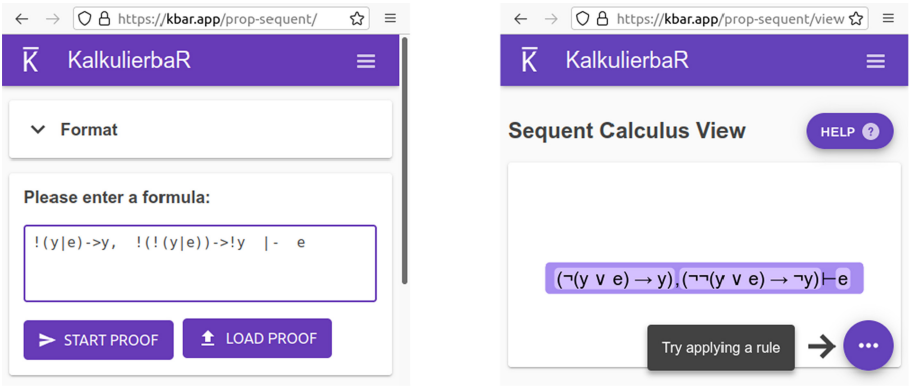

Fig. 1. Starting a sequent proof with `KalkulierbaR`.

Model Generation. For DPLL, `KalkulierbaR` supports checking whether a model satisfies the original formula. For the above example, a resulting model is given by $\neg y$ and e, i.e., the parent gives the egg but not the yogurt. Due to the course contents (and not due to theoretical reasons), model generations had been restricted to DPLL.

Variants and Layout. As discussed, `KalkulierbaR` allows the user to select a variant of the used calculus. For example, the user may use *regular* tableaux, where no literal may occur twice on a branch. If such a restriction is violated, the user is informed which of the variants is not followed correctly and where. Once the proof is closed, the user is also informed which variants could have been activated. For example, the left side of Fig. 3 shows a closed proof with the corresponding message. Beyond feedback on erroneous rule applications, `KalkulierbaR` also has a button that explains all currently available rules and shows animations to illustrate the rule with an example.

Tableaux is the main focus in the course and multiple more advanced calculi are implemented to show variants of tableaux beyond the clausal-based system for standard first-order logic and propositional logic: `KalkulierbaR` provides classical non-clausal tableaux and signed modal tableaux [33]. Just to give an idea of other calculi: On the right of Fig. 3 is a small modal logic proof for the basic modal axiom K.

The layout of DPLL, sequent calculus and tableaux is fixed. For resolution, as no specific layout is used in the lecture, we let the user switch between two

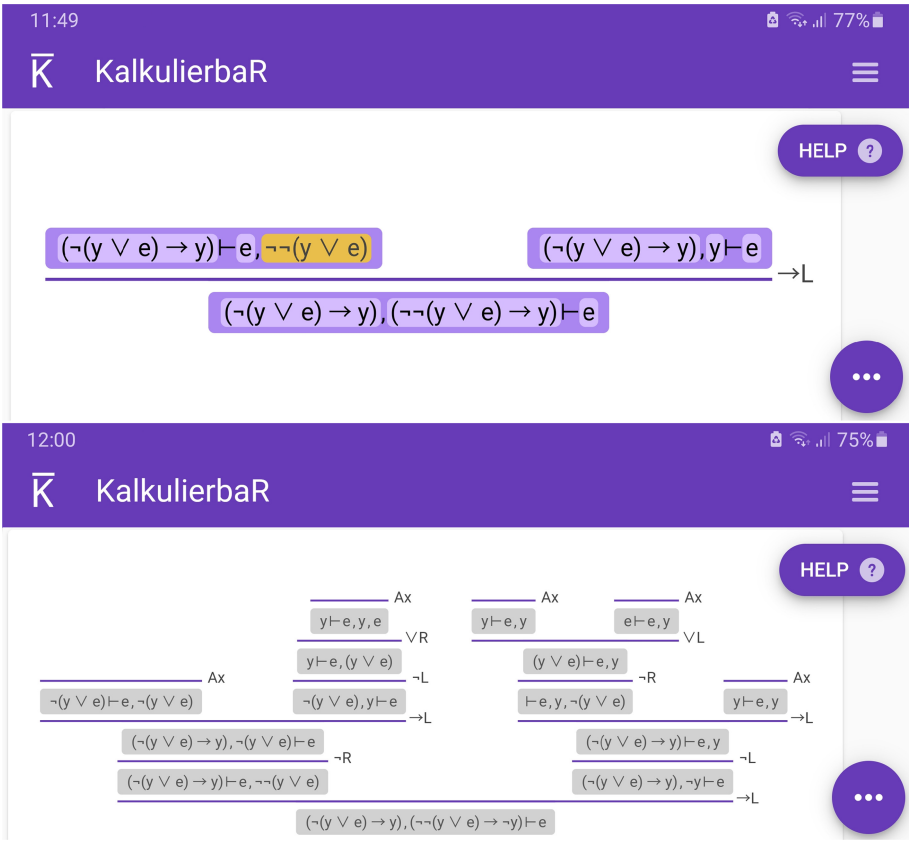

Fig. 2. A sequent proof with `KalkulierbaR` on a mobile phone display.

possible layouts. One where the clauses are arranged in a circle and one where the clauses are arranged in a grid. In the circle, whenever a clause is selected, the clauses are rearranged such that possible resolution partners are near the selected clause.

Backend. The backend permits to hide calculi and variants from the user through an admin interface. This is used to synchronize the course and the calculi offered by the webtool. The backend also automatically translates user input into conjunctive normal form, if the user selects a clausal calculus. Alternatively, the user may enter a set of clauses directly.

Additionally, high-score tables can be activated for certain calculi to compare properties with proofs from other users. After the proof is checked, the high score table appears and the user can enter a name to save store the result. We stress that this is *not* part of the serious game concept. It is a *gamification* approach that is orthogonal to the formal-method-as-serious-game view.

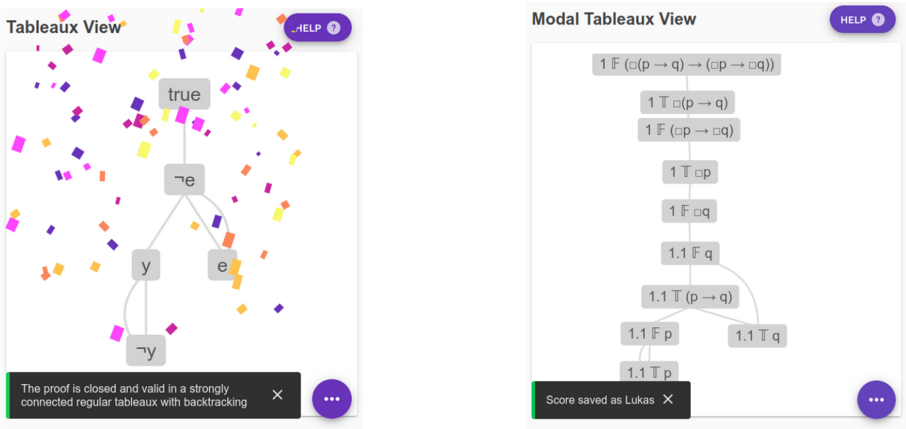

Fig. 3. A clausal tableau proof and a modal tableau proof.

4 Discussion

Application. The original development of `KalkulierbaR` started in winter 2019, based on experiences with the course on "Automated Theorem Proving" at TU Darmstadt. When the course was repeated in summer 2020, we had the opportunity to use `KalkulierbaR` in a course with 15 regularly attending students in the lecture.

As the COVID-19 pandemic forced us to change to a virtual setting, our tool was used differently as originally planned. Similar to other courses we observed that the short-term, unplanned switch to a live streaming format of courses negatively affected student interaction and engagement [17], making it difficult to compare its effects to previous iterations. There was no evaluation on the participants of the lecture in connection with the use of `KalkulierbaR`. Such an evaluation could give detailed feedback on how the usage of `KalkulierbaR` affected the learning process. We plan to make evaluations when the lecture is held again.

Exercises. `KalkulierbaR` was mainly used for exercises, where the exercise sheets could be solved using `KalkulierbaR` instead of pen-and-paper as in the previous years. The solution discussion took place in dedicated sessions. Following the structure of previous iterations of this course, the students were not required to submit their solutions. 4 out of 7 exercises were suited `KalkulierbaR` and the solutions were partially presented using `KalkulierbaR`.

Lecture. The course had 7 lectures on the concepts of tableaux, resolution and DPLL. For the respective calculi, `KalkulierbaR` was used as described in Sect. 3.2. As mentioned, the course suffered from the widely observed negative effect of the pandemic negatively on student engagement. Nevertheless, students actively participated using the text message function, both when

asked and on self-initiative by asking questions for comprehension, when `KalkulierbaR` was used.

Labs. We offered one additional exercise sheet to modify `KalkulierbaR`, thus realizing the *construct* category of engagement. The task of this lab was to modify the rule used for first-order hyper resolution, which had a bug.

Further Forms of Involvement. The previous section discussed our concept for a formal method course focused on an expert teaching a certain method. We could extend it to further forms of involvement from the engagement taxonomy: presenting, which is defined as "*presenting a visualization to an audience for feedback and discussion. The visualizations to be presented may or may not have been created by the learners themselves* [24]". This way, interactive visualization can also be used in a seminar setting where students read up, implement and present variants of formal methods.

`KalkulierbaR` itself was implemented in two mandatory student lab (*"Bachelor Lab"*) which simulates an industrial development environment using agile practices. The project lead, in our case the authors, has an already designed application concept, that needs to be implemented and give the students user studies in regular meetings, which they implement and get approved by the project lead. We used `KalkulierbaR` in two consecutive such labs, where the code from the first lab was extended in the second one. Most of the students had not taken the ATP course before. These labs were educational tasks in themselves, realizing the "constructing" aspect in the engagement taxonomy. The design of `KalkulierbaR` was not part of the students' work.

`KalkulierbaR` has a modular structure that separates the logical operations in the backend from their visual representation in the frontend. As such, a possible lab would be to implement the proof strategies to make the system automatic, such that the visualization can be used to examine the intermediate state without modification. Modifying the visualization allows one also to explore the internal state of the strategy. For example, implementing resolution with set-of-support (SOS), requires adding the SOS clauses to the visual interface.

Furthermore, `KalkulierbaR` implements several calculi and can be extended to support more. It permits a contrasting approach to teach proof multiple calculi in a uniform interface. Single-calculus tools [8,10,15,22] cannot be used so.

On Generalization. The presented connection of formal methods with interactive visualizations as serious games is independent of the chosen implementation for proof calculi and we conjecture that the concept in Sect. 3.2 can be used for any formal method. While there is an overhead to develop a tool specifically for teaching one formal method, we deem it acceptable for the following reasons: As the aim is to provide a serious game form of the method *as taught*, such tools are more reusable by other lecturers than tools used for the application. It is not necessary to have a teaching companion for an advanced tool and keep the teaching tool up-to-date with the application-oriented tool.

That the teaching tool requires less maintenance is an important practical point: Lack of maintenance is of the reasons for the growing disconnect between

KeY and its teaching companion[5], KeY-Hoare, which is based on KeY 1.6, while the current release version of KeY is 2.8 and includes usability improvements that cannot be used in KeY-Hoare. The development of the tool can be integrated with student projects to further increase engagement in an applied manner.

The serious games teaching companion described in this work can be complemented by application-oriented tools if the course structure includes such tools. In our course, we had some case studies using theorem provers as application-oriented tools. There is no redundancy—the teaching companion may be used to introduce the concepts and describe them in their pure form, while application-oriented tools can focus in later parts of the course on bigger case studies.

5 Conclusion

This work establishes a firm connection between formal methods on one side and interactive visualization and serious games on the other side: The formal method itself is a serious game, where the rules of the formal method are the rules of the game. To control the rules and get suitable feedback, the user needs an interactive visualization tool fitted for the formal method.

Using this connection, and further theory from educational sciences, such as the engagement taxonomy, we present a teaching concept tailored to the challenges of formal methods, in particular, the notoriously novice-unfriendly tools. The main goal is to increase student engagement in the theoretical parts of a course, by providing a specific teaching tool that helps to learn the concepts before applying them in the application-oriented tool.

Future Work. We plan to use `KalkulierbaR` in a more mainstream course to be able to perform a quantitative study on its effects on student engagement. To our best knowledge, there are no recent studies on interactive visualization after the advent of smartphones and their mass use by students. It is worth investigating whether this has an effect on how students react to interactive visualization.

We plan to integrate additional output formats, such as LaTeX, and integration into a wider tool for online teaching that builds on `KalkulierbaR` for grading exercises and interactions with students through quizzes and chats, two tools that were also shown to increase engagement in an online setting. Finally, we consider using an additional module to input proof rules, e.g., MUltlog [3], to use `KalkulierbaR` in a setting with a more volatile treatment of proof calculi.

Acknowledgments. The authors thank Daniel Drodt, Julius Henk, Mirko Hirsch, Tim Kilb, and Nils Rollshausen, as well as Nils Elze, Lars Hoffmann, Enrico Martin, Henrik Metternich, and Ashim Siwakoti, who helped to implement `KalkulierbaR` during two student labs. The lecture part of the described course was hold by Reiner Hähnle and Richard Bubel. This work was partially supported by the Research Council of Norway via *SIRIUS* (237898) and *PeTWIN* (294600).

[5] KeY-Java is used to teach Java verification directly, not the general setting.

References

1. Ahrendt, W., Beckert, B., Bubel, R., Hähnle, R., Schmitt, P.H., Ulbrich, M. (eds.): Deductive Software Verification - The KeY Book - From Theory to Practice. LNCS, vol. 10001. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-49812-6
2. Arjoranta, J.: How to define games and why we need to. Comput. Games J. **8**(3), 109–120 (2019). https://doi.org/10.1007/s40869-019-00080-6
3. Baaz, M., Fermüller, C.G., Salzer, G., Zach, R.: MUltlog 1.0: towards an expert system for many-valued logics. In: McRobbie, M.A., Slaney, J.K. (eds.) CADE 1996. LNCS, vol. 1104, pp. 226–230. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61511-3_84
4. Baecker, R. Sorting out sorting. Educational film (1980)
5. Baecker, R.: Sorting out sorting: a case study of software visualization for teaching computer science. In: Software Visualization: Programming as a Multimedia Experience, pp. 369–382 (1998)
6. Cerone, A., Roggenbach, M. (eds.) Formal Methods - Fun for Everybody, Revised Selected Papers. CCIS, vol. 1301. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-71374-4
7. Cerone, A., et al.: Rooting formal methods within higher education curricula for computer science and software engineering - a white paper. White paper (2020). https://arxiv.org/abs/2010.05708
8. D'Agostino, M., Mondadori, M., Endriss, U., Gabbay, D., Pitt, J.: WinKE: a pedagogic tool for teaching logic and reasoning. In: Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V.J. (eds.) ITS 1998. LNCS, vol. 1452, p. 605. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-68716-5_69
9. Dongol, B., Petre, L., Smith, G. (eds.): Formal Methods Teaching. LNCS, vol. 11758. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-32441-4
10. Ehle, A., Hundeshagen, N., Lange, M.: The sequent calculus trainer - helping students to correctly construct proofs. In: 4th International Conference on Tools for Teaching Logic TTL. abs/1507.03666 (2015)
11. Ehle, A., Hundeshagen, N., Lange, M.: The sequent calculus trainer with automated reasoning - helping students to find proofs. In: Quaresma, P., Neuper, W. (eds.) 6th International Workshop on Theorem Proving Components for Educational Software. EPTCS, vol. 267pp. 19–37 (2017). https://doi.org/10.4204/EPTCS.267.2
12. Endriss, U.: An interactive theorem proving assistant. In: Murray, N.V. (ed.) TABLEAUX 1999. LNCS (LNAI), vol. 1617, pp. 308–313. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48754-9_26
13. Farrell, M., Wu, H.: When the student becomes the teacher. In: Cerone, A., Roggenbach, M. (eds.) FMFun 2019. CCIS, vol. 1301, pp. 208–217. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-71374-4_11
14. García-Matos, M., Väänänen, J.: Abstract model theory as a framework for universal logic. In: Beziau, J.-Y. (ed.) Logica Universalis, pp. 19–33. Birkhäuser, Basel (2007)
15. Gasquet, O., Schwarzentruber, F., Strecker, M.: Panda: a proof assistant in natural deduction for all. A Gentzen style proof assistant for undergraduate students. In: Blackburn, P., van Ditmarsch, H., Manzano, M., Soler-Toscano, F. (eds.) TICTTL 2011. LNCS (LNAI), vol. 6680, pp. 85–92. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21350-2_11

16. Gualà, L., Leucci, S., Natale, E.: Bejeweled, candy crush and other match-three games are (NP-)hard. In: IEEE CIG, pp. 1–8. IEEE (2014). https://doi.org/10.1109/CIG.2014.6932866
17. Hjelsvold, R., Nykvist, S. S., Lorås, M., Bahmani, A., Krokan, A.: Educators' experiences online: how COVID-19 encouraged pedagogical change in CS education. In: Norwegian Conference on Didactics in IT education (2020)
18. Hundhausen, C.D.: Toward effective algorithm visualization artifacts: designing for participation and negotiation in an undergraduate algorithms course. In: Karat, C., Lund, A. M. (eds.) CHI, pp. 54–55. ACM (1998). https://doi.org/10.1145/286498.286526
19. Hundhausen, C.D., Douglas, S.A., Stasko, J.T.: A meta-study of algorithm visualization effectiveness. J. Vis. Lang. Comput. **13**(3), 259–290 (2002). https://doi.org/10.1006/jvlc.2002.0237
20. Kahl, W.: Calculational relation-algebraic proofs in the teaching tool CalcCheck. J. Log. Algebraic Methods Program. **117**, 100581 (2020). https://doi.org/10.1016/j.jlamp.2020.100581
21. Landers, R.N.: Developing a theory of gamified learning: linking serious games and gamification of learning. Simul. Gaming **45**(6), 752–768 (2014). https://doi.org/10.1177/1046878114563660
22. Materzok, M.: Easyprove: a tool for teaching precise reasoning. In: 4th International Conference on Tools for Teaching Logic TTL. abs/1507.03675 (2015)
23. Myller, N., Bednarik, R., Sutinen, E., Ben-Ari, M. Extending the engagement taxonomy: software visualization and collaborative learning. ACM Trans. Comput. Educ. **9**(1), 7:1–7:27 (2009). https://doi.org/10.1145/1513593.1513600
24. Naps, T.L., et al.: Exploring the role of visualization and engagement in computer science education. ACM SIGCSE Bull. **35**(2), 131–152 (2003). https://doi.org/10.1145/782941.782998
25. Ölveczky, P.C.: Teaching formal methods for fun using maude. In: Cerone, A., Roggenbach, M. (eds.) FMFun 2019. CCIS, vol. 1301, pp. 58–91. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-71374-4_3
26. Pierce, B.C., et al.: Software foundations (2021). https://softwarefoundations.cis.upenn.edu/lf-current/index.html
27. Prasetya, W., et al.: Having fun in learning formal specifications. In: ICSE-SEET, pp. 192–196 (2019). https://doi.org/10.1109/ICSE-SEET.2019.00028
28. Richardson, F.C., Suinn, R.M.: The mathematics anxiety rating scale: psychometric data. J. Couns. Psychol. **19**(6), 551–554 (1972). https://doi.org/10.1037/h0033456
29. Roggenbach, M., Cerone, A., Schlingloff, H., Schneider, G., Shaikh, S.A.: Formal Methods for Software Engineering. Springer (2022, to appear). https://link.springer.com/book/9783030387990
30. Smullyan, R.: The riddle of Scheherazade and other amazing puzzles, ancient & modern. New York (1997)
31. Spichkova, M., Zamansky, A.: Teaching of formal methods for software engineering. In: Maciaszek, L.A., Filipe, J. (eds.) ENASE, pp. 370–376. SciTePress (2016). https://doi.org/10.5220/0005928503700376
32. Wouters, P., van Nimwegen, C., van Oostendorp, H., van der Spek, E.D.: A meta-analysis of the cognitive and motivational effects of serious games. J. Educ. Psychol. **105**(2), 249–265 (2013). https://doi.org/10.1037/a0031311
33. Zach, R.: Boxes and diamonds: an open introduction to modal logic (2019)
34. Zhumagambetov, R.: Teaching formal methods in academia: a systematic literature review. In: Cerone, A., Roggenbach, M. (eds.) FMFun 2019. CCIS, vol. 1301, pp. 218–226. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-71374-4_12