NuFrameIT: Serious Games via Computation and Reasoning

Stefanie Heim, Michael Kohlhase[0000-0002-9859-6337], Silas Kuder, and Florian Rabe[0000-0003-3040-3655]

Computer Science, FAU Erlangen-Nürnberg

Abstract. In the FrameIT framework for serious games, we have previously experimented with ideas for co-maintaining virtual worlds (such as those maintained by game engines like Unity) with logical worlds (such as those maintained by knowledge representation and reasoning systems). By treating the virtual world as a physical model of the logical world that tracks the player's knowledge, it achieves a coupling that is stable even if the player can interactively extend this knowledge with measurements taken in the virtual world and with deductions about them.
In this paper we present a major reconceptualization, NuFrameIT, aimed to alleviate limitations that have been discovered by previous prototypes. It uses a new representation language for the logical worlds that combines both axiomatic and efficiently executable knowledge descriptions, and allows for a more dynamic and fine-grained maintenance of the logical world. And it implements a novel technique to display interactive mathematical formulas within the game.

Keywords: serious games, FrameIT, UniFormal, virtual world, mathematical knowledge

1 Introduction

The FrameIT framework for serious games [CICM20; UFO] explores how to use mathematical knowledge management (MKM) techniques for developing serious – i.e. educational – math games. Such games combine four aspects:
A1 a virtual world which is explored in the game
A2 the player/learner's knowledge about of the virtual world
A3 the mathematical background of the topic to be conveyed by the game
A4 the didactic ability to convey the topic effectively.
(**A1**) is usually (and usefully) implemented in a game engine like Unity [UNTY] or Unreal [UNR]. Those employ complete concrete representations including e.g. positions, velocities, and bounding boxes, well-suited for simulating an immersive learning environment for the user.

Most serious games implement the remaining three aspects via custom code in the programming language underlying the game engine. This has the drawback that the mathematical knowledge (e.g. an analytic treatment of trajectories),

player knowledge and the way both can interact, all have to be specified in a programming language designed for fast numeric computation, not abstract reasoning.

Therefore, the FrameIT system separates concerns by using a game engine as **frontend** for the maintenance and display of (**A1**), and a logic-based knowledge management system as **backend** to explicitly formalize (**A2**) and (**A3**) as a set of logical theories. The latter focuses on a knowledge-oriented high-level perspective that is well-suited for logical problem-solving. (Good treatment of (**A4**) is still a desideratum of research, but techniques from [Ber+23; ALeA] should apply.)

The concrete implementation UFrameIT used the Unity engine and the MMT system [MMT], which provides knowledge management services including elaboration of theories, type checking and simplification. The theory graph paradigm underlying MMT has been an excellent fit for the FrameIT framework: the player knowledge (**A2**) can be represented as a dynamic theory that grows as the player's knowledge about the worlds expands, called the **situation theory**. It can be seen as an abstraction of the virtual world, or a mathematical model of those parts of the world that are currently relevant. The background knowledge (**A3**) is represented as a modular knowledge graph in MMT.

In addition to the usual ways to explore a virtual world, FrameIT adds two game mechanics to allow for mathematical exploration by the player: **Gadgets** allow actions like defining points, specifying lines, and measuring distances in the virtual world. These generate new **facts**, player knowledge that is represented by visuals in game, and by an extension of the situation theory in the backend. **Scrolls**[1] allow the use of small theories in the background knowledge that represent the abstract mathematical theorems that the game wants to teach. To apply a scroll, the player instantiates the free variables of the scroll with objects in the situation theory. Then the situation theory is extended by taking the pushout of the scroll along this instantiation in the category of MMT theories and morphisms [CICM20].

We have been able to build several prototypical serious games based on this framework, validating the overall approach. But these prototypes also unveiled serious limitations in the initial design:

L1 The text display facilities in game engines are ill-suited for the mix of prose and formulas, ubiquitous in the communication of mathematical ideas.

L2 MMT's bias towards formal logic allowed computation only via symbolic rewriting and pushout. This was the more problematic, the more concrete a piece of knowledge was, suboptimal for a framework that uses a virtual world specifically to ground knowledge as much as possible.

(**L1**) fundamentally limited the complexity of topics that were worth attempting, and (**L2**) led to the use of workarounds that made the resulting system too complex to scale up the game development.

Contribution We present a major redesign of multiple parts of the system, which we call *NuFrameIT*. We employ the WebView plugin [Ess24] developed for the

[1] The name is intended to invoke the image of a piece of magical parchment.

dynamic and interactive display of mathematical text, and have replaced the MMT backend with the new UniFormal language [CICM25] that was partly inspired by the complexity of the FrameIT design. UniFormal combines both logical/mathematical knowledge representation features using theories with object-oriented mathematical computation features akin to the Axiom computer algebra system [JS92]. Contrary to MMT, UniFormal's implementation is systematically lean and can be compiled into a small, easily packagable binary or JavaScript file. This enabled us to replace the makeshift web service-based connection between Unity and MMT with a tightly coupled integration of Unity and UniFormal. By calling directly into UniFormal's data structures and operational semantics, the representation of the situation theory becomes much simpler and scroll application much smoother.

In the sequel, we explain the key innovations both from the frontend and the backend perspective.

2 Frontend Perspective

We use the well-known word problem of determining the height of a tree via the intercept theorem as a running – didactically motivated – example. Figure 1 shows a diagram of the idea: a person can determine the height h of a given (vertical) tree by sighting the top and bottom of the tree over a (vertical) stick of known length h_a. If we know the distance d between person and tree and the arm length d_a, we can compute the height as $h = \frac{d}{d_a} \cdot h_a$.

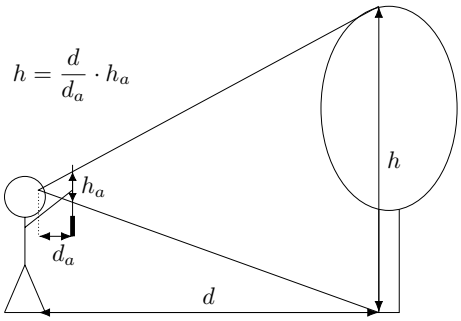

Fig. 1. Tree Height via Intercept Theorem

Figure 2 shows the corresponding construction in the virtual world halfway towards a solution. On the right side is the Unity rendering of the virtual world, the toolbar at the bottom contains the gadgets that can be used for, e.g. defining points and measuring distances. The player has previously employed these to mark the points B through F, and to measure the distances $[DF]$, $[DC]$ and $[EF]$.

A scroll consists of (at least) two different representations of the same knowledge, and a formal description of how they are connected. The representation of a scroll for the backend is a formal theory, see Section 3. The representation of a scroll for the frontend is a small narrative document that corresponds to text in a mathematics textbook, and may also include other media like the triangle on the bottom left. On the left of Figure 2 is such a document that describes how to apply the intercept theorem in a concrete situation. It is, however, not a static image, but an HTML webpage, using MathML for the formulas, SVG for the triangle, and JavaScript for interactivity. The interactivity is used to support the player in applying the theorem in the virtual world problem. In our example,

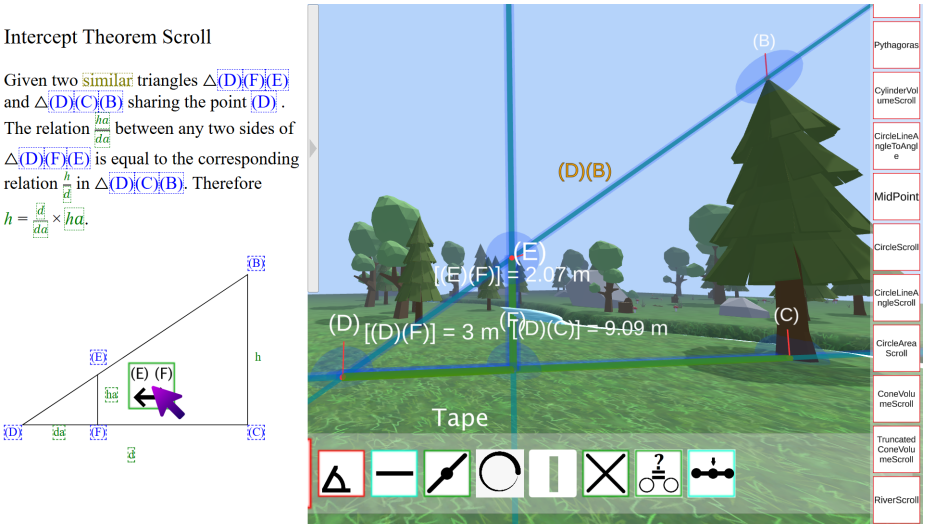

Fig. 2. A Tree Height Problem being solved in FrameWorld

this requires first specifying the relevant points, and measuring the distances d, d_a, and h_a. Every scroll has free variables that must be instantiated, which can be done via drag'n'drop. Figure 2 shows the player in the process of dragging the line segment $[(E)(F)]$ (represented by the square icon below the cursor on the left) from the virtual world onto the variable h_a in the scroll. Once all variables are instantiated, the scroll can be applied.

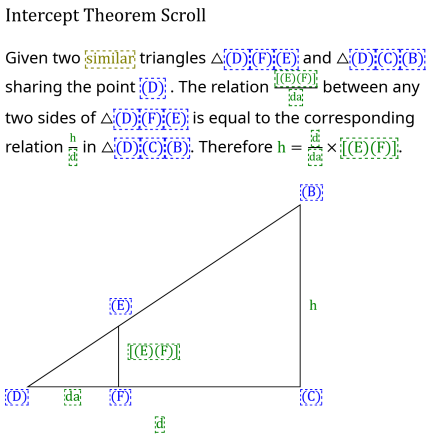

Fig. 3. The scroll after dropping $[(E)(F)]$

Figure 3 shows the resulting display of the scroll after thusly instantiating h_a with the measured length of the line segment $[(E)(F)]$: every occurrence of h_a has been substituted. This interactive display of formulas was very difficult to achieve, but critical to overcome (**L1**). Previously, Unity could only approximate the display of more visually complex mathematical formulas like fraction by ASCII art. We now use the WebView plugin that Björn Esswein developed for the display of HTML5 (including MathML) in Unity worlds. WebView is essentially a headless Chrome browser that streams images into Unity, and events (like dropping a fact) back out via the Chrome Developer tools Protocol; see [Ess24] for details on the plugin itself,

and why existing technologies did not suffice. This solves the math front-end problems for NuFrameIT in a way that decouples the presentation of the scroll (which is a didactic document that is part (**A4**)) from that of virtual world (**A1**) in the game engine. This has the added benefit that the HTML descriptions and JS interaction scripts could likely be reused with minor – if any – changes in any other game engine that allows for a similar plugin, or at least supports webpage display.

3 Backend Perspective

The listing below shows the formalization of the document for the intercept scroll from Figure 2. It declares 5 points and the lengths of 4 sides of the resulting triangles. The type point is part of the background knowledge and opaque. Even though the coordinates are known to Unity, they could only be exposed to the situation theory through an explicit measurement. But no such measurement is ever needed in our example, which only requires relative distances.[2] This of course means, the dist function cannot be defined using the usual distance metric, but has to remain abstract as well.

Therefore, the lengths must be declared as separate abstract fields in the scroll. This decouples the representation of the floating point value of e.g. d, which will be needed to perform computations, from its geometric meaning. Said geometric meaning is captured by the axioms. E.g. the axiom d_prop captures that d is the length of the line segment DC. NuFrameIT uses the naming convention of appending _prop for the defining properties, such that the line segment d that is visible in the user interface can be connected to its geometry axiom. NuFrameIT also prefixes all field names in a scroll with an underscore to prevent name clashes. This has been omitted here for readability.

```
theory InterceptTheoremScroll
  // variables
  B, C, D, E, F : point
  similarity: ⊢ similar(Triangle(D,C,B), Triangle(D,F,E))
  d, h, da, ha: float
  d_prop:  ⊢ dist(D,C) == d
  h_prop:  ⊢ dist(C,B) == h
  da_prop: ⊢ dist(D,F) == da
  ha_prop: ⊢ dist(F,E) == ha
  // payload
  intercept: ⊢ h/d == ha/da
```

To instantiate the scroll, 3 of the 4 lengths must be instantiated with concrete values, and the proof obligations induced by the corresponding geometry axioms (as well similarity) must be discharged by instantiating them with an

[2] In fact, we do not even provide a gadget that would allow measuring absolute coordinates, inspired by our physical reality's adamant refusal to provide one either.

appropriate fact, i.e. proof term. These are named properties that were generated when using a gadget, or applying a previous scroll. For example, the action in Figure 2 is performed in a situation theory that already contains the axiom EF_prop: $\vdash$ dist(E,F) == 2.07 that was added when the player used the tape gadget on E and F. Because dist is abstract, the expression dist (E,F) can never be evaluated, and a term for EF_prop cannot be created by a numerical coincidence, but only through Unity explicitly exposing of the distance between the two points it calculated in the virtual world. The action in Figure 2 of dragging that fact icon onto ha extends the situation theory with the instantiation ha_prop = EF_prop. Additionally, unification of their types induces the instantiation ha = 2.07. Importantly, it is not possible for the player to instantiate ha with an arbitrary number or measurement: The interactive instantiation-building only succeeds if the types unify, and that requires that the geometry axioms in the scroll match the geometric meaning of the measurements performed to obtain the facts they are to be instantiated with.

There is no distinction between variables that must be instantiated and variables that can be computed – the pushout for scroll application is always defined. It can be constructed simply by adding the uninstantiated variables to the situation theory, and solving the payload theorem for their declarations. But for didactic reasons, it is practical to attach meta-information about the intended instantiations, like Figure 3 stating which 3 of the 4 lengths should be instantiated with concrete values to compute a large vertical distance.

The theory resulting from scroll application is initially needlessly complex: Firstly, it contains the instantiations for the scroll variables, and their right-hand side is usually just a number or an identifier. Which can be substituted everywhere and then dropped from the situation theory. Secondly, the concrete values of some declarations can be derived using the payload theorem, such as h in our case. For this purpose, UniFormal provides a *theory simplifier*. We use it to rewrite the situation theory into an isomorphic one after every scroll application. It performs three kinds of operations:

1. Expand identifiers with simple definiens into their definiens, and
2. remove identifiers whose definitions have been expanded by (1).
3. Solve for the values of identifiers that are determined by the axioms (such as h in our example), and add these as definiens, then
4. remove axioms that have become redundant because of (3).

In our running example, the scroll application adds uninstantiated declarations for h and intercept. Step (1) expands all declarations for variable instantiations by expanding them into their definitions, such as ha and ha_prop Step (2) removes these declarations entirely. Step (3) solves intercept for h and then adds the resulting value as the definiens of h. At this point, the axiom intercept is redundant because all 4 lengths have numeric definitions and the formula is simply true. It is thusly removed in Step (4). At the end of theory simplification, all names declared in the scroll have disappeared from the situation theory, and the definiens of h has been added. Not only does this keep the situation theory simple, it also prevents name clashes when performing repeated applications

of the same scroll with different instantiations. (Name clashes from a single application are prevented by the underscore convention mentioned above.)

4 Conclusion

We have reported on a major re-conceptualization of the FrameIT approach to serious games, fixing the weak points that have been discovered by previous prototypes: We have completely replaced the backend with one based on the new UniFormal language, which is a much better fit for FrameIT. In the front-end we have added a plugin to the game engine that provides responsive rendering of state of the art mathematical notation, and we have build interaction components that allow the player to interactively build the coupling between logical and virtual world.

The new components work well together and provide a much more natural and principled serious game development framework than the original ad-hoc prototype version of FrameIT.

References

[ALeA] *ALeA: Adaptive Learning Assistant*. URL: http://alea.education (visited on 07/16/2024).

[Ber+23] M. Berges, J. Betzendahl, A. Chugh, M. Kohlhase, D. Lohr, and D. Müller. "Learning Support Systems based on Mathematical Knowledge Managment". In: *Intelligent Computer Mathematics (CICM) 2023*. Ed. by C. Dubois and M. Kerber. Vol. 14101. LNAI. Springer, 2023. DOI: 978-3-031-42753-4. URL: https://url.mathhub.info/CICM23ALEA.

[CICM20] M. Kohlhase et al. "FrameIT: Detangling Knowledge Management from Game Design in Serious Games". In: *Intelligent Computer Mathematics (CICM) 2020*. Ed. by C. Benzmüller and B. Miller. Vol. 12236. LNAI. Springer, 2020, pp. 173–189. DOI: 10.1007/978-3-030-53518-6_11. URL: https://kwarc.info/kohlhase/papers/cicm20-frameit.pdf.

[CICM25] F. Rabe. "Global, Regional, and Local Contexts". In: *Intelligent Computer Mathematics (CICM) 2025*. Ed. by V. de Paiva and P. Koepke. Vol. 16136. LNAI. Springer, 2025. URL: https://github.com/UniFormal/UPL/blob/main/doc/levels.pdf.

[Ess24] B. Esswein. "Interactive MathML Formulas in UFrameIT". B.Sc. Thesis. FAU Erlangen-Nürnberg, Dec. 2024. URL: https://gl.kwarc.info/supervision/BSc-archive/blob/master/2024/EsseinBjoern.pdf.

[JS92] R. Jenks and R. Sutor. *AXIOM: The Scientific Computation System*. Springer, 1992.

[MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: https://uniformal.github.io/ (visited on 01/15/2019).

[UFO] *The FrameIT Project*. Project Website. URL: https://uframeit.org/ (visited on 03/19/2026).

[UNR] *The most powerful real-time 3D creation tool – Unreal Engine*. URL: https://www.unrealengine.com (visited on 03/19/2026).

[UNTY] *Unity Engine: 2D & 3D Development Platform | Unity*. URL: https://unity.com/products/unity-engine (visited on 03/19/2026).