Snowflake: Supporting Programming and Proofs

Anh Vu, Oluwatobi Alabi, and Peter-Michael Osera (Summer 2022 MAP)

Grinnell College

Abstract

Theoretical (proof-based) mathematics courses, while important to developing well-structured programs and efficient algorithms, often suffer from the lack of constant feedback or assistance from instructors during the proof-writing process. While proof assistants—computer systems that help users develop proofs[1]—such as Coq and Carnap[2] have been built to help resolve this issue, they do not allow for integration into the pedagogies of most classes without a steep learning curve for students, especially those in introductory courses. Our solution is *Snowflake*, a proof assistant tool that assists computer science (CS) students in understanding, validating, and authoring well-structured proofs while supporting the pedagogy of the professor. Snowflake supports a variety of proof reasoning techniques and currently supports two discrete mathematics classes with distinct pedagogies. Snowflake is implemented with TypeScript which enables its integration with an SMT solver (Alt-Ergo) and an web-based, interactive UI for students and instructors.

Background

Mathematical skills are undoubtedly essential in writing well-structured programs and designing efficient algorithms. However, theoretical (proof-based) mathematics courses are generally perceived by students as challenging and pointless[3].

The problem stems from not only the sheer difficulty of the field, but also from the involved pedagogical approaches. While introductory CS courses often take place in feedback-rich environments, theoretical mathematics ones do not: instructors first explain proof concepts expecting students to fully absorb them, students then follow the model with pencil and paper, receiving limited external feedback much later when the work is graded[4]. Such methodology seems inadequate for students to fully absorb and become fluent in the concept and skills of mathematical proofs.

Snowflake uses both logic and code reasoning to support students' understanding of the proof writing and validation process[4]. Snowflake, formally Orca, was originally built with a logic programming-like core for carrying out deductive reasoning and an embedded domain-specific language in Haskell for authoring logics, along with a web-based block-based front-end for authoring proofs[4]. Our work expands Snowflake by integrating inductive reasoning into its core engine, incorporating it with an SMT solver (Alt-Ergo), migrating the codebase from Haskell to Typescript to allow easier integration with a web interface, and creating a more accessible interface and interactive experience.

Results

Snowflake is now built with an SMT solver, a web interface, and supports (First-order logic) logical reasoning, inductive reasoning, substitution, and other discrete mathematic proof methods.

Within the pedagogy of Grinnell's CSC208 (Discrete Structures) class, Snowflake supports students by validating the proofs they write into the system by taking the proof question and generating its own proof using inductive reasoning and Alt-Ergo. It first generates the proof in a tree-like structure using theorems supported by first-order logic and natural deduction. Snowflake then runs the students' proof through its system, validating the theorems that students used at each step in the proof. From this, Snowflake informs students about the correctness of their proof, and where they made mistakes. This helps students learn why their proofs are incorrect and helps them to get better at writing proofs and understanding the full logic behind the proof.

Snowflake also supports the pedagogy of the Haverford CS1 class, which uses a subset of Python that allows for logical expressions to conduct proofs about systems. Snowflake supports students here as a proof-writing tool. At each step, students choose a theorem to be applied to the current state of the proof. Snowflake then checks the validity of the step (by applying the proof by substitution method based on the theorems the student chooses and validates the substitution using Alt-Ergo), giving error messages to prompt students to try again if the step is incorrect. Here, Snowflake becomes a tool that students can use to hone their proof-authoring skills and more profoundly understand the systems in question.

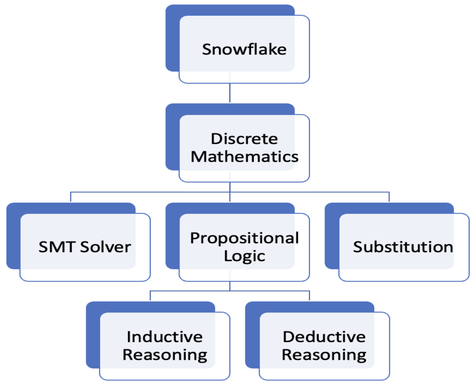

Figure 1: Snowflake's support for discrete mathematics proof methods

Figure 2 shows an example of a proof from the Haverford CS1 pedagogy. Here, we are trying to verify the correctness of the *power* function. The first box on the left shows the initial state of the proof in question. To the right are step names that can be applied to one proof state to produce the next.

We can see that while doing complicated proofs such as this one, students may find it difficult to identify which part of the proof state a rule is being applied to, and what assumptions allow them to use that rule. Also, students may not be able to immediately identify their mistakes and unknowingly proceed with writing the proof. In the figure below, we show how the same proof can be written using Snowflake, and how the proof assistant can help address the problems mentioned before and others, allowing students to author and understand their own proofs in a concise and thorough manner.

Snowflake provides support for multiple discrete mathematics proof methods, including substitution, propositional logic (inductive and deductive reasoning), and testing satisfiability using an SMT solver.

```
(   power(x, y)
if  y > 0 else
    ERROR
)                                                        name-to-body
    (   x
    if  y == 1 else
(       x * power(x, y-1)
    )
if  y > 0 else ...                                       name-to-spec-simpler
    (   x
    if  y == 1 else
        x * (  x ** (y-1)
(           if  y-1>0 and y>1 and y>y-1 else
                ERROR
            )
    )
if  y > 0 else ...                                       algebra (consider-tests)
    (   x
    if  y == 1 else
        x * (  x ** (y-1)
(           if  True else
                ERROR
            )
    )
if  y > 0 else ...                                       if-True
    (   x
    if  y == 1 else
(       x * ( x ** (y-1) )
    )
if  y > 0 else ...                                       algebra
    (   x
    if  y == 1 else
(       x ** y )
    )
if  y > 0 else ...                                       algebra
    (   x ** y
    if  y == 1 else
(       x ** y )
    )
if  y > 0 else ...                                       if-irrelevant
(   ( x**y )
if  y > 0 else
    ERROR
)                                                        Q.E.D.
```

Figure 2: Example of a proof from the Haverford CS1 pedagogy

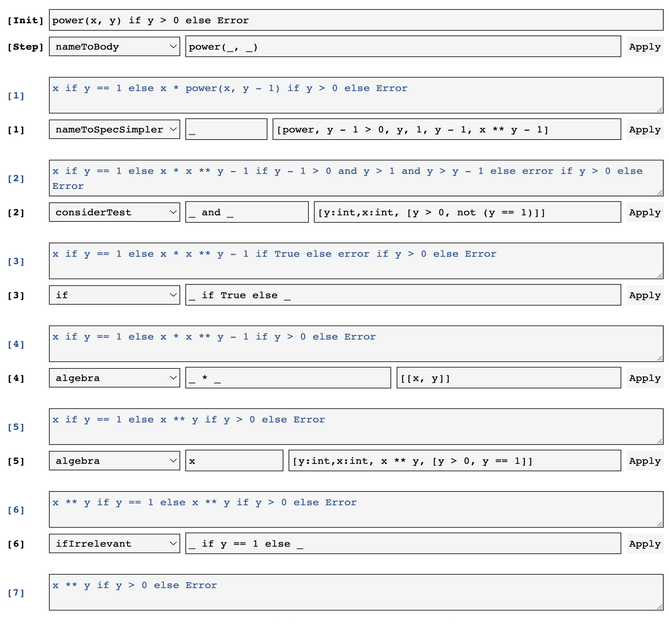

Figure 3: Proof in figure 2 authored using Snowflake

Figure 3 shows the example proof in Figure 2 being written and executed using Snowflake:

1. At the first two steps, Snowflake runs *nameToBody* which substitutes the *power* function call with its body. Then, the rule *nameToSpecSimpler* looks for the recursive call and substitutes it with a conditional that helps prove the recursive function with abstract variables.

2. At step 2, Snowflake runs the rule *considerTest* on the proof which invokes the SMT solver. To do this SMT solver requires premises that the student inputs as the argument to that rule, Snowflake ensures that those arguments (y > 0, not (y == 1)) are premises provided in the body of the proof.

3. At steps 4 and 5, Snowflake uses algebra simplifies the expressions x * x ** (y – 1) into x ** y when y > 0, and x into x ** y when y == 1.

4. At step 6, there are now two identical terms x ** y, Snowflake uses the *ifIrrelevant* rule to make the expression more concise.

5. At step 7, we can see that x ** y is equivalent to power(x, y) in the initial expression. Therefore, the proof is correct.

Future Directions and Conclusion

We plan to utilize the tool for both Grinnell's CSC208 class and Haverford's CS1 class in an upcoming semester to gauge its effectiveness in helping students understand, author, and validate well-formed and appropriate proofs. We intend on gathering feedback from both the students and instructors of said classes and use that feedback to make informed decisions about revisions to Snowflake's design, including the addition of extra features and remedies to its shortcomings.

In addition to revising Snowflake based on feedback from students and instructors about its gauged effectiveness as a proof assistant, we have also laid out plans to adapt and improve some of its features. First, we plan to continue improving Snowflake's user interface to provide students with a user experience that is intuitive and easy to learn and use. Some potential features include the ability to revisit the history of past proofs and steps and the functionality of importing and exporting proofs to and from downloadable files. We also plan to make Snowflake more accessible to other instructors' pedagogies by generalizing the tool instead of constraining it to our two current pedagogies.

References

1. Geuvers, H. "Proof Assistants: History, Ideas and Future." Sadhana 34, no. 1 (February 2009): 3–25. https://doi.org/10.1007/s12046-009-0001-5.
2. Leach-Krouse, Graham. "Carnap: An Open Framework for Formal Reasoning in the Browser." Electronic Proceedings in Theoretical Computer Science 267 (March 2, 2018): 70–88. https://doi.org/10.4204/EPTCS.267.5.
3. Sigurdson, Nikki, and Andrew Petersen. "A Survey-Based Exploration of Computer Science Student Perspectives on Mathematics." In Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 1032–38. Minneapolis MN USA: ACM, 2019. https://doi.org/10.1145/3287324.3287416.
4. Wonnacott, David G., and Peter-Michael Osera. "A Bridge Anchored on Both Sides: Formal Deduction in Introductory CS, and Code Proofs in Discrete Math," 2019. https://doi.org/10.48550/ARXIV.1907.04134.

Acknowledgements

We would like to thank Professor Dave Wonnacott at Haverford College for his guidance and support in developing Snowflake framework that supports his pedagogy for the Haverford CS1 class, and Professor John Dougherty at Haverford College and Dr. Bruce Char at Drexel University for their professional advice during the development of Snowflake. We would also like to acknowledge all students who previously worked on Snowflake as their valuable contributions are crucial to making Snowflake the tool it is today.