Using Concept Maps for Notional Machine Selection in CS1

Ethan Richards
Colorado School of Mines
Golden, Colorado, U.S.A.
erichards@mines.edu

Sonia Spindt
Cherry Creek Innovation Campus
Centennial, Colorado, U.S.A.
sspindt@cherrycreekschools.org

Gabriel Fierro
Colorado School of Mines
Golden, Colorado, U.S.A.
gtfierro@mines.edu

ABSTRACT

A notional machine is an abstract representation of a program's execution. These alternate representations serve to highlight otherwise hidden or subtle behavior in program execution, and eliminate unnecessary or irrelevant details that can obstruct understanding. Prior work has established that, by construction, there is no "universal" notional machine that is appropriate for the entirety of a CS1 curriculum. How, then, does an educator choose an appropriate notional machine for a given learning sequence?

We present initial work on helping educators choose an effective sequence of notional machines for a CS1 course. Our key insight is to leverage a concept map capturing hard and soft dependencies between computing concepts. By characterizing concepts and notional machines together, we show that it is possible to design feasible sequences of notional machines that cover a desired learning sequence.

ACM Reference Format:
Ethan Richards, Sonia Spindt, and Gabriel Fierro. 2024. Using Concept Maps for Notional Machine Selection in CS1. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635535

1 INTRODUCTION

Notional machines (NMs) are a tool for explaining computing concepts in different or abstracted terms. These alternate representations serve to highlight otherwise hidden or subtle behavior in program execution, and eliminate unnecessary or irrelevant details that can obstruct understanding. Prior work has established that, by construction, there is no "universal" notional machine that is appropriate for the entirety of a CS1 curriculum [2, 4]. Thus, the natural question arises: how does an educator choose an appropriate notional machine for a given learning sequence?

Our key insight is to root this decision making process in a CS1 concept map that captures hard and soft dependencies between computing concepts. We represent the map as a directed graph so a learning sequence can be described as a traversal of that graph. To inform the choice of notional machines for a given learning sequence, we look to recent work in characterizing notional machines by their representation and concepts covered [1]. Through annotating the notional machines with the same concepts as captured in the map, we show that it is possible to design and evaluate different sequences of notional machines that complement the desired learning sequence.

In this poster abstract, we show our initial work in using a hypothetical CS1 concept map to choose a sequence of notional machines given a learning sequence over the concept map.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635535

2 BACKGROUND AND RELATED WORK

A notional machine is an abstract representation of a program's execution [4]. Fincher [1] proposes two forms in which this representation can manifest: a representation-based interactive activity or an analogy-based representation. Such representations are beneficial to a learner to understand topics and form their own mental models of program flow and state.

[2] asserts that here is no single "universal" notional machine; that is, there is no single abstraction sufficient to cover all CS1 concepts. This means that a full learning sequence for a CS1 course would require multiple notional machines, whether analogy-based, representation-based, or otherwise. However, little work has been done in mapping sequences of notional machines to curricula.

Concept maps are an established idea in pedagogy [3]. Such maps can be used to better inform the development and design of curriculum by portraying it in a sequence, and then adapting material based on the constraints of the educator.

However, such sequences are not guaranteed to be linear, and have no such description of dependence between concepts. As a result, there is a need for helping educators choose an effective sequence of notional machines. We propose that if we express concept maps and notional machines together using a directed graph, we can leverage the graph's structure to inform the selection of sequences of notional machines in a CS1 curriculum.

3 CONCEPT MAP FOR CS1

Educators have inherent constraints such as time, domain knowledge, student interest, or mandated curriculum (e.g., AP). Concept maps can help to make obvious the distinctions within a directed set of concepts and how to best approach designing a learning sequence given those constraints.

Figure 1 shows a simplified concept map with multiple possible learning sequences for teaching functions. Each node represents a single programming concept, such as variables. A programming concept is a generalized concept such that it can relate to other concepts, and can have child sub-concepts of its own. For simplicity and clarity of presentation, we illustrate only a subset of our larger concept map that we have developed. Notional machines in the figure are drawn from [1].

3.1 Relationships

In our formulation, concept maps have two types of relationships: hard prerequisites and soft prerequisites. **Hard prerequisites** capture which concepts *must* be covered before another concept can be taught. For example, functions are a natural hard prerequisite of teaching recursion.

Soft prerequisites capture optional, but helpful, dependencies between concepts. For example, while we contend that an understanding of data types in programming languages is essential knowledge for understanding variables as containers of state (hard prerequisite), a conceptualization of expressions is not strictly necessary. This does limit the kinds of programs that can be expressed by the learner (i.e. only those with static assignments to variables). We capture this concept topology in Figure 1.

3.2 Learning Sequences

The intent of the concept map is to capture all of the *hard* and *soft* dependencies between programming concepts. The process of designing a learning sequence can thus be framed as choosing a topological sort of this graph such that (a) hard prerequisites must be covered before their dependent concepts, and (b) soft prerequisites *may* be covered before their dependent concepts. The concept map offers a principled way of choosing which concepts to teach, and in what order. Further, we will show how the concept map can inform the choice of notional machines (§4).

We first illustrate how our (intentionally) simple concept map permits multiple learning sequences for starting from data types and progressing to teaching functions. The first sequence (`a-c-e`) minimizes the number of concepts covered before functions are taught by ignoring any soft prerequisites. Another choice (`a-b-c-e`) includes expressions in the learning sequence to expand the set of programs that learners can write; here, expressions are taught before variables because of the soft prerequisite relationship.

Neither of these learning sequences requires teaching control flow. However, if the educator decides to include recursion as a learning goal, then the concept map will lead them to include control flow as a necessary component of the learning sequence. The topology of the map allows them to cover control flow either before or after functions, as long as it is covered before teaching recursion. This also changes the potential sequence of notional machines.

Ultimately, there is no one correct way of mapping notional machines to concepts in any given curriculum, but we intend to give a set of options to educators that are useful enough to produce a learning sequence.

4 CHOOSING A NOTIONAL MACHINE SEQUENCE

Notional machines can be characterized in terms of program concepts using a concept map. This characterization can be applied with a graph-theoretic approach in order to choose a sequence of notional machines to use in a curriculum.

Using the concept map, the educator can determine which notional machines can be used to teach or illustrate a given concept. With the added context of dependencies between concepts, the educator can also determine how a particular *sequence* of notional machines can best support the learner experience.

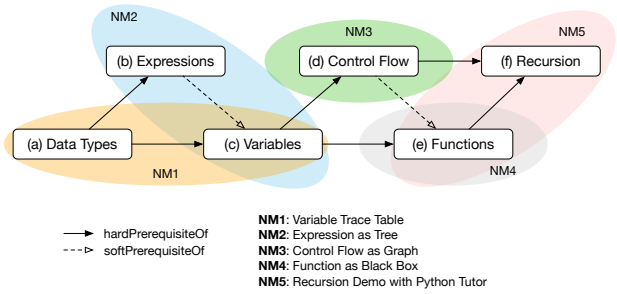

Figure 1: A simple concept map allows multiple possible learning sequences for teaching Functions: `a-c-e`, `a-b-c-e` and `a-c-d-e`. Notional Machines (NM*X*) are drawn from [1]

Consider the sequence `a-c-d-e` from Figure 1. This sequence can be covered with notional machines NM1, NM3 and NM4. NM2 would not necessarily be appropriate to use for teaching variables because its design also incorporates expressions, which are not in the learning sequence. The use of the concept map and notional machine characterization supports this choice. If an educator wished to extend the above sequence to also include recursion (`a-c-d-e-f`), then they must make a decision whether to (a) use NM4 to teach functions and then NM5 to teach recursion, or (b) use NM5 to teach both functions and recursion.

One caveat within a sequence is that transitioning between notional machines may be confusing for learners. For that reason, an educator might want to minimize the number of notional machines used to cover concepts in a sequence. Expressing a learning sequence with concepts and notional machines together is an effective tool to solve this issue; educators can reason about when the transition between one notional machine to another may or may not be appropriate and timely for students.

Our proposed approach enables informed co-design of learning sequences and notional machines. However, it is ultimately the responsibility of the educator to decide which sequences of notional machines are beneficial to implement in their learning sequence.

5 CONCLUSION

Graph theory can be utilized with concept maps to better inform our decisions about the sequence of notional machines in CS1 curricula. Future work will approach the problem of determining "good" sequences of concepts and notional machines algorithmically.

REFERENCES

[1] Sally Fincher, Johan Jeuring, Craig S Miller, Peter Donaldson, Benedict Du Boulay, Matthias Hauswirth, Arto Hellas, Felienne Hermans, Colleen Lewis, Andreas Mühling, et al. 2020. Notional machines in computing education: The education of attention. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*. 21–50.

[2] Sally A Fincher and Anthony V Robins. 2019. *The Cambridge handbook of computing education research*. Cambridge University Press.

[3] Vinicius dos SANTOS, Érica F de SOUZA, Katia R Felizardo, and Nandamudi L Vijaykumar. 2017. Analyzing the use of concept maps in computer science: A systematic mapping study. *Informatics in Education* 16, 2 (2017), 257–288.

[4] J Sorva. 2013. Notional Machines and Introductory Programming Education. *ACM Transactions on Computing Education (TOCE)* 13, 2 (2013), 8–1.