Curricular Automata and Their Applications

Yen-Teh Hsia
Department of Information and Computer Engineering
Chung Yuan Christian University, Chung Li, Taiwan, R.O.C.

Abstract
Computer-assisted learning can be either single-thread or multi-thread. When learning is single-thread, the learner works with one learning unit at a time; having passed or failed the "current" learning unit, the learner then moves on to the next learning unit. When learning is multi-thread, several learning threads may be "developing" simultaneously, and the learner switches among the current learning units of the learning threads. The learning-path approach is adequate for supporting the development of learning threads on a coarse level of granularity, such as planning the sequences of topics to study. But for learning that occurs on a fine level of granularity, the learning-path approach may not work so well. In this paper, we introduce the concept of curricular automata, and show how this formalism can be used to support both single-thread learning and multi-thread learning.

1. Introduction

One characterization of learning called single-thread learning is as follows. The learner selects some learning unit (a suitable chunk of learning activities) as the "current learning unit". When the learner passes or fails the current learning unit, he/she moves on to the next learning unit, which then becomes the current learning unit. The entire learning process (called a learning thread) comes to an end when either the learner has passed an associated goal unit or the underlying coaching mechanism decides that it can no longer help the learner pass a goal unit.

Based on the concept of single-thread learning, multi-thread learning is defined as follows. (1) There are one or more learning threads; some may have come to an end; some may be still ongoing; others just remain to be "activated". (2) One or more threads are automatically activated at the start. (3) A thread that is not activated at the start may later become activated when some other threads end. (4) Once a thread has been activated, it will stay activated. (5) Different (activated) threads may come to an end at different times. (6) The learner is concurrently engaged in all of the ongoing threads (by switching among the current units of the threads). (7) The entire learning process comes to an end when all activated learning threads end.

In a nutshell, a learning thread is just a linearly connected chain of learning units. Connections are in the form of arrows, labeled as either 'P' or 'F', and the arrows all point in the same direction (toward the end of the chain). Each learning thread has a learning goal to achieve. A learning thread ends successfully (alternatively, unsuccessfully) if and only if (a) the last element of the (final) learning thread is an associated goal unit, and (b) the learner has passed this last unit.

Past research on supporting the development of learning threads has mainly focused on finding suitable "learning paths" for the learner [1]. Very simply put, a learning path (called "formative path" in [1]) is a sequence of learning units. Knowing the extent of knowledge of the learner (including what learning units the learner has passed), the coaching mechanism suggests to the learner a "suitable" learning path that has a goal unit located at the end of the path. (The system may even suggest several learning paths for the learner to choose.) The idea is that the learner may be able to "move" successfully "along" the path until he/she has passed the goal unit located at the end of the path. If the learner fails a unit that is located somewhere in the middle of the suggested path, then supposedly the system would have three options. It either suggests a new path to the learner, or asks the learner to try again (to pass the learning unit in question), or just considers it the case that the learner has failed.

When it comes to learning that occurs on a fine level of granularity, the learning-path approach may not work very well. In this paper, we introduce the concept of curricular automata, and show how it may be used in an automated tutoring context. Section 2 motivates and defines curricular automata. Section 2 also gives an idea of how single-thread learning and multi-thread learning may be supported. Section 3 concludes.

2. Curricular automata

Suppose we want to help the learner solve a problem $Q_\lambda$. Solving $Q_\lambda$ is, in effect, the goal that we want the learner to achieve. However, we don't want to just give an answer to the learner. If the learner does not know how to solve $Q_\lambda$, we want to help the learner find its solution (see Figure 1). First, we just present $Q_\lambda$ to the learner, and see if he/she can solve the problem (let us call this learning unit $A$). If the learner can solve $Q_\lambda$, then the whole learning process is over (and the final learning thread contains just one passed learning unit which is a goal unit). If the learner cannot solve $Q_\lambda$, then we present to him/her several similar but simpler problems, perhaps with hints of some kind if necessary. Suppose that the learner passed this second learning unit
A3 F A1-F-B-P-A2-P-PASS is a learning thread (not counting the final PASS state). In other words, a learning thread is always a manifestation of the working of some underlying curricular automaton in our framework, with the learner’s performance (the assessed learning results) being translated into an input sequence of P’s and F’s. Consequently, the task of devising a tutoring plan (in the simplest, one-goal case) just amounts to the design of a curricular automaton - something that an instructor may be able to do by him/herself.

To have N (>1) learning threads develop simultaneously, all we need is just a list of the N underlying curricular automata. Some of these automata can even be the same automaton, because a curricular automaton is capable of supporting the development of different learning threads (albeit of the same “learning nature”). We also need to tell the coaching system which learning thread is to be supported by what curricular automaton. Details of our approach can be found in the full paper [3].

3. Some discussions

Curricular automata can be used together with a learning-path approach. The learning-path approach is used for planning the “right” curricular automata to use, while curricular automata themselves are responsible for handling the details of learning. For each learning goal, we design a curricular automaton for it. In doing so, we try to handle all reasonably conceivable cases, and we associate each state with a suitable learning unit.

The underlying coaching mechanism is basically an “automata interpreter”. First, it initializes the learner’s learning status as the initial states of the underlying curricular automata. Then, the system just keeps on updating the “current states” of the learner with respect to the automata in action. Whenever an automaton reaches its final state, the system uses the learning-path approach to decide about the next curricular automaton to use. The process continues until there are no more automata we can use in order to accomplish more goals.

Acknowledgement

An earlier part of this research was supported by a grant (89-H-FA07-1-4) from the Ministry of Education of the Republic of China.

References