Writing Questions for an Intelligent Book Using External AI

William Billingsley, Kasim Rehman, and Peter Robinson
University of Cambridge
wbillingsley@cantab.net

Abstract

Intelligent Books are Web-based textbooks that can adapt and improve their content and guide students through graphical example exercises that resemble the diagrams and notations a student might use on paper. The exercises use formal AI systems to analyse students' work, and different AI systems are used for different questions. This brings the issue of how a person can write questions if they are not an expert in the AI system used. We describe our experiences developing an authoring tool for electronics questions that use a specialised circuit AI with its own extensive circuit language. The tool works on the principle of exposing an appropriate visual model of the AI, while factoring out the language detail and the architecture of the book itself, and allowing the question writer to decide which parts of the AI model to expose to the student (as the desired mental model for the student).

1. Introduction

Intelligent Books [1, 2] are online textbooks that: automatically improve their content; allow students to add new material and annotate existing material with their comments; use artificial intelligence to help students work through graphical example exercises; and relate advice from the AI during an exercise to content within the book.

Writing and managing content for intelligent tutors and AI-based learning environments is difficult in general, and various approaches have been taken to address this concern [3]. These include providing a generalised development environment [4], allowing programming by demonstration [5], and providing representational views of the data to be used in an inquiry tutor [6]. For Intelligent Books, a particular issue is that questions are centred on domain-specific AIs and diagram notations, and different questions can use very different styles of AI and diagram. For example, the exercise we discuss in this paper uses a constraint propagation system [7] to analyse students' work determining values for currents, voltages, and components on a circuit diagram; whereas an exercise we developed for mathematics uses the Isabelle/HOL proof assistant [8] to help analyse student-written proofs. While question writers can be expected to understand the general architecture behind questions, it is unrealistic to expect them to know low-level details about each AI model language, or implementation details of how those model elements are mapped to the different diagram languages that are used. For Intelligent Books to be viable, then, we must be able to produce authoring tools that reduce the need for low-level knowledge, without restricting the expressiveness of the underlying languages (which would limit the range of questions that can be asked).

In this paper we describe an authoring tool we developed for building circuit specification questions, and the suitability of our approach.

2. The Question and AI

The student is given the diagram of an electrical circuit and a set of requirements that it must meet. He or she must then set currents, voltages, and component values on the diagram in order to fully specify the circuit. All answers that obey the rules of electronics and meet the requirements are accepted as correct.

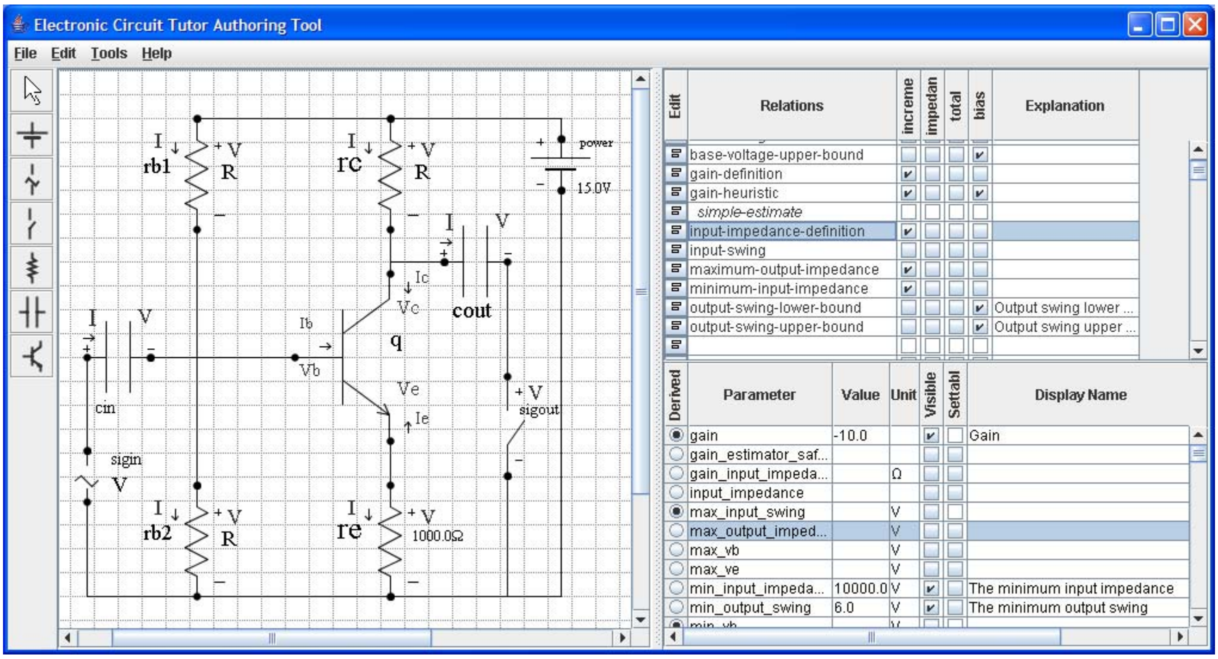
When a value is set on the diagram, the teaching script passes it into the constraint propagation AI, which makes deductions based on a *relation* (or *constraint*) model of the circuit. For example, in a circuit node where three wires meet, Kirchhoff’s Current Law imposes the relation that the three currents entering the node must sum to zero. If two currents are set, the AI will deduce the third; if all three are set, it will signal a contradiction if they do not sum to zero. Deduced values are propagated into other relations to make further deductions. The teaching script marks the deduced values and contradictions on the diagram. The student can ask to see how any of these came about, in which case the teaching script animates the AI’s sequence of deductions on the diagram. The relation model contains many more variables than most students’ mental models of circuits, making unedited explanations from the AI long and cumbersome, so the teaching script uses the diagram as a context against which to prune the AI’s explanation before it is animated. Effectively, the diagram is treated as a desired mental model that the student should have of the circuit, and the discourse between the student and AI is centred on that model. The student can also ask the teaching script for strategic advice.

3. Authoring Tool

The authoring tool generates the three low-level parts needed to specify a question: the Scheme-based relation model used by the AI, the XML-based circuit diagram for the client, and the XPath- and Scheme-based mappings between them for the teaching script. The desired mental model that we present visually to the user is that defining a question involves designing the relation model of a partly specified circuit, and then deciding which variables and components within that part to expose to the student (as the desired model for the student).

In the AI’s relation model, each circuit element (“*part*”) has *terminals*, *parameters*, and *relations*. A terminal has a *current* and a *potential*. Relations may involve the terminal currents, the terminal potentials, and the parameters. For example, a resistor has two terminals; the currents in the terminals are related by Kirchhoff's Current Law, and the difference in the terminal potentials is related to the currents and the resistance parameter by Ohm's law. Parts can also have different *models*, with different relations in each model. Transistors often have a bias model describing their steady-state behaviour and an incremental model describing their response to transient signals. The relation model is also hierarchical: a transistor amplifier is composed of parts, but it is also a part that can be used in larger circuits. So, it has its own parameters and relations. For instance its *gain* relates the signal output to the signal input in the incremental model. The gain can be set directly, or circuit values inside the amplifier can cause it to be deduced.

The tool’s main interface, shown in Figure 1, allows the user to define a part by constructing it from component parts available as icons on the left. The relations and parameters for the part being defined (the amplifier) are created and maintained in the two panes on the right. The hierarchical nature of the relation model is exposed using a *namespace browser*, shown in Figure 2. At the top of the tree are the part’s global parameters (marked **p**) and its models (marked **M)**. Within the models are component parts, such as the nodes b and c, and the capacitor cin. Within each component part are the component’s own parameters, models, and terminals. The dot-separated path at the foot of the namespace browser describes how to refer to a particular parameter when defining relations. (A



Edit	Relations	increme	impedan	total	bias	Explanation
	base-voltage-upper-bound	□	□	□	☑	
	gain-definition	☑	□	□	□	
	gain-heuristic	☑	□	□	☑	
	simple-estimate	□	□	□	□	
	input-impedance-definition	☑	□	□	□	
	input-swing	□	□	□	□	
	maximum-output-impedance	☑	□	□	□	
	minimum-input-impedance	☑	□	□	□	
	output-swing-lower-bound	□	□	□	☑	Output swing lower ...
	output-swing-upper-bound	□	□	□	☑	Output swing upper ...
		□	□	□	□	

Derived	Parameter	Value	Unit	Visible	Settabl	Display Name
●	gain	-10.0		☑	□	Gain
○	gain_estimator_saf...			□	□	
○	gain_input_impeda...		Ω	□	□	
○	input_impedance			□	□	
●	max_input_swing		V	□	□	
○	max_output_imped...		V	□	□	
○	max_vb		V	□	□	
○	max_ve		V	□	□	
○	min_input_impeda...	10000.0	V	☑	□	The minimum input impedance
○	min_output_swing	6.0	V	☑	□	The minimum output swing
●	min_vb		V	□	□	

Figure 1: The authoring tool presents a unified visual model of the relational model and diagram.

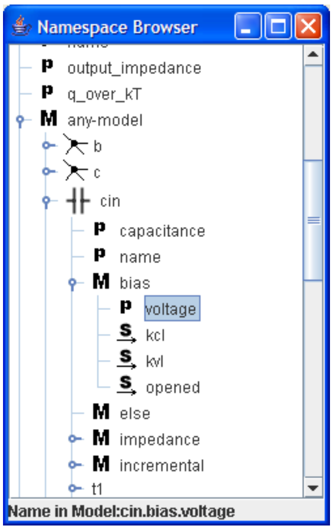


Figure 2: A namespace browser exposes the hierarchical nature of the circuit.

relation editor performs syntax checking and assistance in defining relations.)

The diagram of the circuit to show the student is taken from the layout of the diagram in Figure 1. This is then refined by altering the visibility of components and parameters to the student, and whether they are read-only, in the right hand panes of Figure 1 and through context menus on the diagram. English language descriptions of the relations can also be entered. This allows the writer to create the desired model for the student from the AI relation model.

5. Conclusions

We have been able to generate different circuit questions using the tool, and the process of creating the question appears effortless compared to specifying the question in the low-level languages. There is still a reasonable amount of understanding required to write to write a question, but that understanding is about composing circuits from parts and describing the relationships between circuit properties under different models – the inescapable core without which it is hard to talk about the question as a concept.

Since the interaction with the AI is the core of the question, we focus on providing an approachable visual model of the AI's modelling language, and then allow the question-writer to specify a desired mental model for the student based on that AI model. The fact that it has been possible to do this for a non-trivial question with non-trivial AI, without losing much of the flexibility of the underlying language, gives us confidence that this is a suitable approach to authoring tools.

6. Acknowledgments

This paper describes joint work with Hal Abelson, Gerald Sussman and Chris Hanson from the Massachusetts Institute of Technology.

7. References

[1] Billingsley, W., and P. Robinson, "Towards an intelligent online textbook for discrete mathematics", in *Proceedings of the 2005 International Conference on Active Media Technology, Takamatsu, Japan*. pp 291 – 296. 2005.

[2] Billingsley, W., P. Robinson, P., M. Ashdown, and C. Hanson, "Intelligent tutoring and supervised problem solving in the browser", in *Proceedings of the IADIS International Conference WWW/Internet 2004, Madrid, Spain*. pp 806 - 811. 2004.

[3] Murray, T., "An overview of intelligent tutoring system authoring tools: updated analysis of the state of the art." Chapter 17 in T. Murray, S. Blessing, and S. Ainsworth (Eds.) *Authoring Tools for Advanced Technology Learning Environments*, Kluwer, 2003.

[4] Munro, A, M.C. Johnson, Q.A. Pizzini, D.S. Surmon, D.M. Towne, and J.L. Wogulis, "Authoring simulation-centered tutors with RIDES", *Intl. Journal of Artificial Intelligence in Education*, 8, pp284-316, 1997.

[5] Koedinger, K.R., V. Aleven, N. Hefferman, B. McLaren, and M. Hockenberry. "Opening the door to non-programmers: authoring intelligent tutor behaviour by demonstration" *Proceedings of the Seventh International Conference on Intelligent Tutoring Systems*, 2004.

[6] Murray, T., B. Woolf, and D. Marshall, "Lessons learned from authoring for inquiry learning: a tale of authoring tool evolution", *Intelligent Tutoring Systems*, Springer, pp197 – 206, 2004

[7] Stallman, R.M., and Sussman, G.J., "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis", *Artificial Intelligence*, 9(2): pp 135 – 196, 1977.

[8] Nipkow, T., L.C. Paulson, and M. Wenzel, *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, Springer, 2002.