Thanks, again, for sending the reviews of our partial draft of How Computers Work. Ruben and I have been mulling this over and would like to propose an unusual course of action.

Reviewer 1 points out that the book, as it stands, targets a limited audience. There aren't many existing courses that could use How Computers Work as their main text. The reviewer correctly observes that we have tried to place logic as a fundamental tool of software engineering in the same way that physics is a fundamental tool of electrical engineering.

We don't think we failed in that mission, as the reviewer does, but it is true that our claim to success is based on the view functional programming is the proper domain for most software development projects. As reviewer 1 points out, the adoption of functional programming by the bulk of software developers is not going to happen. However, we believe that a book that illustrates how logic could be the physics of software development, even if it's limited to the functional programming domain, contains a body of knowledge that all software engineers could benefit from studying.

The problem is in finding courses where such a book could be adopted. Almost all computer science programs require a discrete math course. Few require a separate logic course. Therefore, reviewer 1's suggestion that we expand our book to a full-fledged discrete math text is good advice, and we are prepared to do that. It will take a while, though. The most popular texts are long. Rosen (McGraw Hill) is 800 pages. Grimaldi (Pearson) is 700 pages. Hein (Jones and Bartlett) is 1,000 pages.

So, it's a big project. Our approach would be to place each topic in the context of one or more important applications in software development, or at least within the field of computer science. Rosen and Grimaldi pay lip service to this notion, at best. Hein does a better job of it, but falls short of our goals. These ideas cannot be thrown over the fence in a minimalist context and remain useful to students of software engineering and computer science.

Popular discrete math texts cover three to five dozen separate concepts in 500 to 1,000 pages. That's 10 to 20 pages per concept. Placing them in context approximately doubles the space requirements, so we'd be on the high end of that figure. So, our 800-page book would cover 30 to 40 concepts, which should make it adoptable in many CS programs. We think we could have a draft meeting our goals ready for review in the August, 2013 timeframe.

However, we don't want to give up on our original concept: a short book (about 200pp) of interest to people studying CS or SE using logic as a basis for understanding how computers work. We think such a book would play a role similar to the one we imagine Friedman's "little" series provides in course work. Namely, the role of a supplementary text that students study to help them understand what they are learning in a course that has another book as it's primary text. In addition, we think our short, How Computers Work, book would appeal to professional software developers and other intellectually curious people.

So, we would like to publish two books: the short one (How Computers Work) and the long one (Discrete Math for Software Engineers). We would be happy to have How Computers Work be online only, if MIT Press finds that advantageous. The same goes for the long book, but an 800 page book seems unwieldy for online use.

We hope you will give this proposal some consideration and let us know what you think.