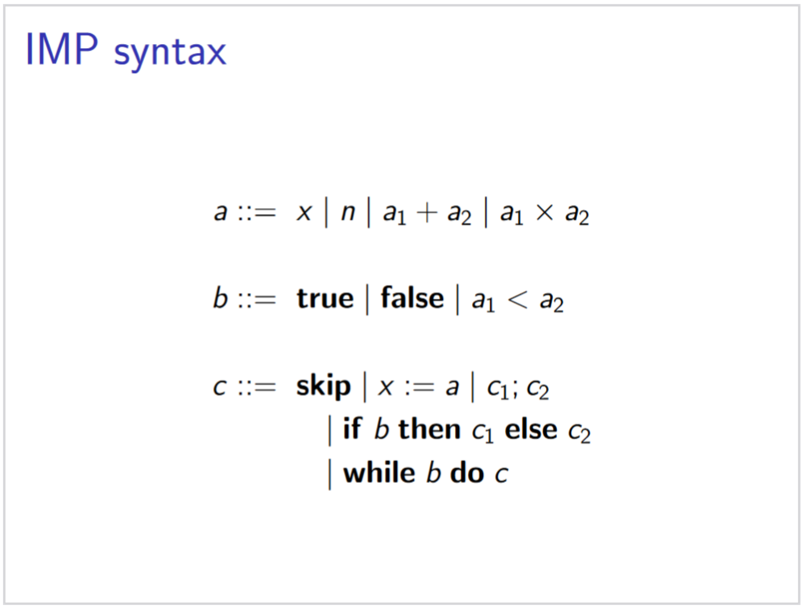
Simple Language Specification

Ari Primak

November 29, 2022

As I was writing the language spec I discovered that it wasn't as trivial a task as I had expected–It was, in fact, a significantly more trivial task than I had expected! I fast realized that I was reiterating the same primitive syntax required for every Turing-complete language (loops, conditional branches, basic memory management, etc.) and that I would get an identical practical result from using a language defined in the slides, or, really, anywhere else. I decided I will just use the Simple IMP language from the slides (also included in an image below). The facet of this simple language that is of particular interest to me is that it does not have the unary '-' operator and so I'll need to be clever when I include logical statements more complex than the bare minimum. It's clear that such behavior will need to be modeled with the if;then;else command, but I'm unsure as yet of the specifics and expect it to be the most complicated part of the project. I decided to compile to C as my target due to the anything-goes nature of the language meaning I'll have to spend more care on the output code to preserve security than I might need were I targeting a language that demands more responsibility from authors' of source code.



IMP syntax

$$a ::= x \mid n \mid a_1 + a_2 \mid a_1 \times a_2$$

$$b ::= \textbf{true} \mid \textbf{false} \mid a_1 < a_2$$

$$\begin{aligned} c ::= {} & \textbf{skip} \mid x := a \mid c_1; c_2 \\ & \mid \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\ & \mid \textbf{while } b \textbf{ do } c \end{aligned}$$

Figure 1: Simple IMP Language from Slides