

A Model for Representing Ruby Circuits

Carl Johan Block

Datavetenskap, Chalmers Tekniska Högskola
Göteborg, Sweden

Abstract

This paper presents a method for drawing Ruby circuit descriptions. The method contains a sophisticated algorithm that calculates the placing of circuits and the connections between them, and sometimes transforms circuits from one shape to another. For some instances of Ruby circuits, ambiguities in the visualisation appear and this can make it very tricky to know how a circuit is supposed to be drawn. The algorithm makes a good choice when to transform and when not. The pictures are constructed in a compositional way, which reduces re-calculation and simplifies the building of a system that implements the algorithm. This approach gives layouts that are easy to read and the system can support different layout policies. The model is used in a project to develop a software system that draws Ruby circuit descriptions and is a part of the Glasgow Ruby Compiler.

1 Introduction

Ruby is a hardware description language in which, by defining a relation between a domain and a range, a description of a circuit that realises the relation can be designed [3, 4, 5]. This design also suggests the actual layout of the circuit on the implementation technology. In realistic Ruby designs, it can be very difficult to get an idea of what the circuit is like, because of the complexity of the descriptions. This is why it is useful to have a module in the Ruby compiler that draws the circuits. The layout description of the circuit and the way of thinking about them that is advocated by Jones and Sheeran [3, 7] also supports the drawing of circuits.

Since Ruby not only gives the expression that corresponds to a circuit, but also the actual layout and placing of the different sub-circuits, much of the visualisation is already captured. The problem that has to be solved is to get a good algorithm that is able both to draw basic circuits and to draw composite circuits without re-drawing sub-circuits. The algorithm also must support some transformations of circuits when ambiguities appear.

2 Composing Ruby circuits

In Ruby, circuits can be combined in different ways to form composite circuits. Ruby can describe many circuits: basic gates such as AND, OR and INVERTER-gates; serial and parallel composition; map; below, beside, row and column; converse; wiring-combinators.

A Ruby circuit defines a relation from its domain to its range. The circuit can be described as a tile that has a number of connections, which can appear on any of the four sides of the tile. The connections carry signals which can be directed either in to the tile or out from it. A two-sided circuit has connections on the left and the right sides of the tile, representing the domain and the range of the relation, respectively. If the domain and the range both are pairs, the relation can be seen as a four-sided tile. The domain appears on the west and the north sides, and the range on the south and the east sides of the tile, see figure 1.

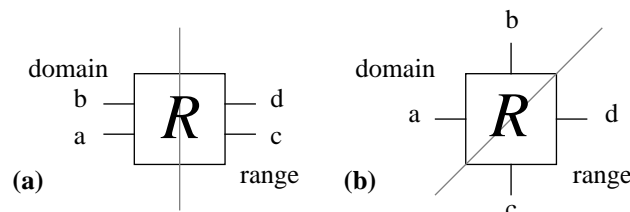


Figure 1: Domain and range of (a) the two-sided and (b) the four-sided Ruby circuit, $R: (a, b) \sim (c, d)$. The west and the north sides of the four-sided tile form a pair in the domain and the south and the east a pair in the range.

Parallel and serial composition, written $[R, S]$ and $R ; S$ respectively, are used to combine circuits in a two-sided way, while below and beside are used to combine in a four-sided. Map gives a repeated parallel composition of a circuit. Row and column generalise beside and below in the same way as map generalises parallel composition. Converse, writ-