BarModel Script

A DSL for "drawing" bar models

Author: Xuesong Wu (wuxuesong@gmail.com)

Abstraction

Bar modeling is an effective method for primary school students to visualize and understand word problems in Math. However, drawing a clean bar model with accurate size is not that easy. I designed a domain specific language, named bar model script, that allows primary school students and teachers to "draw" bar models by writing script. A tool that supports bar model script will help the students to focus on the problem itself rather than worrying about the sizes of individual boxes when they draw a bar model. A bar model script can not only define how a final bar model looks like but also specify how the bar model is drawn step by step.

In article, I explore the semantic model of bar models, the syntax of the bar model script, and discuss how a tool that supports bar model script can be implemented.

1. Introduction

Bar modeling is an effective method that can help students visualize and understand word problems in Math. Learning to draw bar models of word problems is a natural way to develop abstract thinking skills. Let's look at an example:

Mike has 3 less books than Alice. Wendy's books are 3 times as Alice. Alice and Mike's total books are 15 less than Wendy's. How many books does Wendy have?

It is not that easy for a primary 5 student to fully understand the problem without visualizing it.

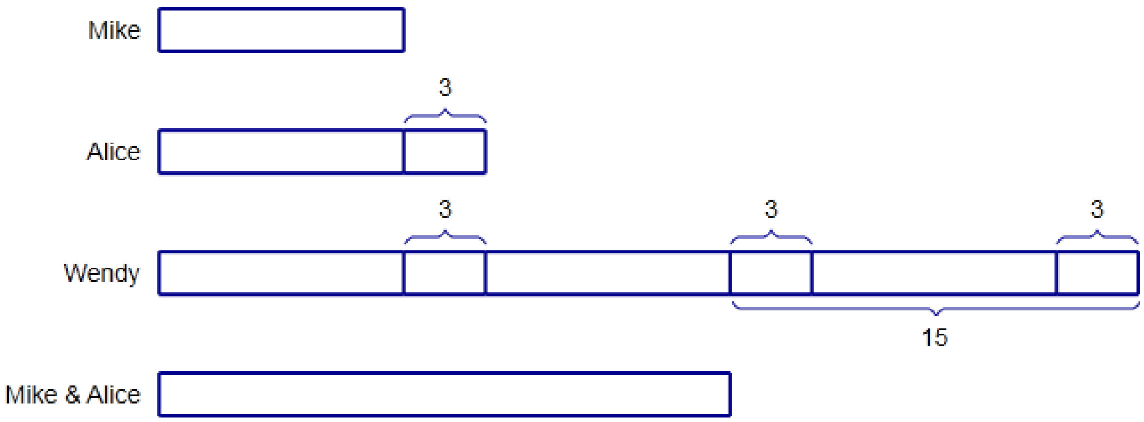


Figure 1: Book problem

Drawing a clean and clear bar model is challenging, teaching a student to draw bar models is even more so. A bar model should be drawn accurately to certain extent. An inaccurate bar model can be misleading. The problem is it is not easy to draw a bar model with accurate sizes of bars on paper, or even with software tool.

Another issue with bar models is that, in most cases, students can only see the final models. As a bar model becomes complex it itself can become difficult to understand for those who don't know how the bar model was drawn step by step.

The bar model Script, a domain specific language, is specially designed to tackle these issues. A software tool that supports the language would allow you to "draw" a bar model by writing script instead of drawing it directly. As you are adding more statements, the tool calculates the size of the bars and boxes and draws the model automatically. Once you've added enough constraints, which are derived from the word problems, the tool will draw a bar model with precise sizes.

The script not only defines the final model but also explains how the model is drawn, thus making it possible for students to examine the bar model by stepping forward and backwards to understand the meaning of each step.

2. The Semantic Model

A bar model consists of one or more bars. Each bar has a title and one or more boxes. The fill color, stroke color, and stroke style of a box can be specified. A box label can be assigned to one or more boxes and can be above or below the box(es). A side label can be assigned one or more bars. Figure 2 shows the elements of a bar model.

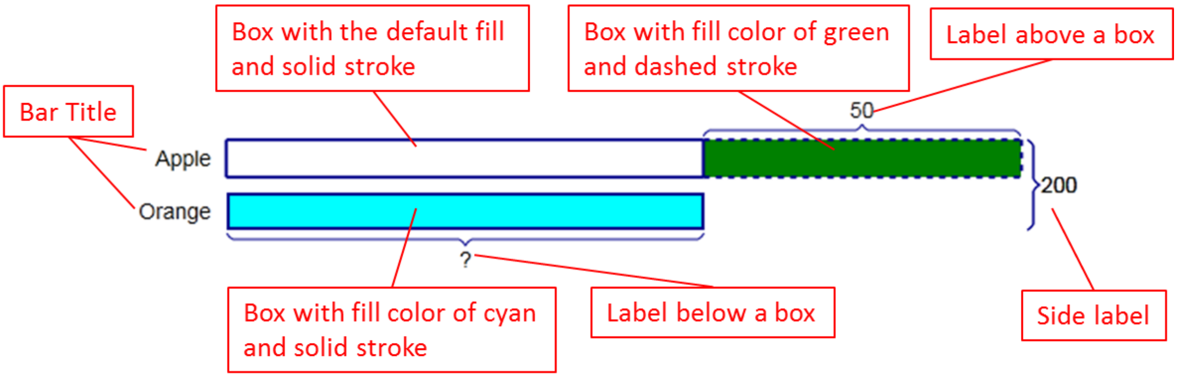

Box with the default fill and solid stroke
Box with fill color of green and dashed stroke
Label above a box
Bar Title
50
Apple
Orange
200
?
Box with fill color of cyan and solid stroke
Label below a box
Side label

Figure 2: The elements of a bar model

After a box is added to a bar, it can be cut into smaller boxes or shifted to another bar. The properties of a box can also be changed to make the bar model more visual.

One important task of the semantic model is to calculate the sizes of the boxes. One solution is to generate linear equations from the semantic model to form a matrix equation A*X = B, where A is an m x n matrix, B is a 1 x n vector, and X is a 1 x n vector of variables, which represents of sizes of the boxes. Once the matrix equation becomes solvable we can solve it to get the exact sizes of the boxes. When the equation is under-determined, we can "estimate" the sizes of boxes by using linear programming, for example.

The matrix equation is made solvable by adding constraints that are derived from the word problem. There are two kinds of constraints: ratio constraints and additive constraints.

A ratio constraint defines a ratio relationship between 2 or more boxes. For example, the ratio of the 3 boxes of the first bar is 5 : 3 : 1.

An additive constraint defines a relationship of 2 or more boxes that can be described with addition and/or subtraction. For examples, the first box of the first bar plus the first box of the second bar minus the second box of the third bar is 50.

Figure 3 shows the class diagram of the semantic model. A BarModel consists of Bars, Labels, and Constraints. A Bar has one or more Boxes. A SideLabel can be assigned to one or more Bars. A BoxLabel can be assigned to one or more Boxes. A RatioConstraint has two or more BoxRatio objects, each referring to a Box and having a value representing the ratio. An AdditiveConstraint has two or more BoxSign objects, each referring to a Box and having an integer indicating the sign of the box. The constant value of an AdditiveConstraint represents of the value of the arithmetic expression.

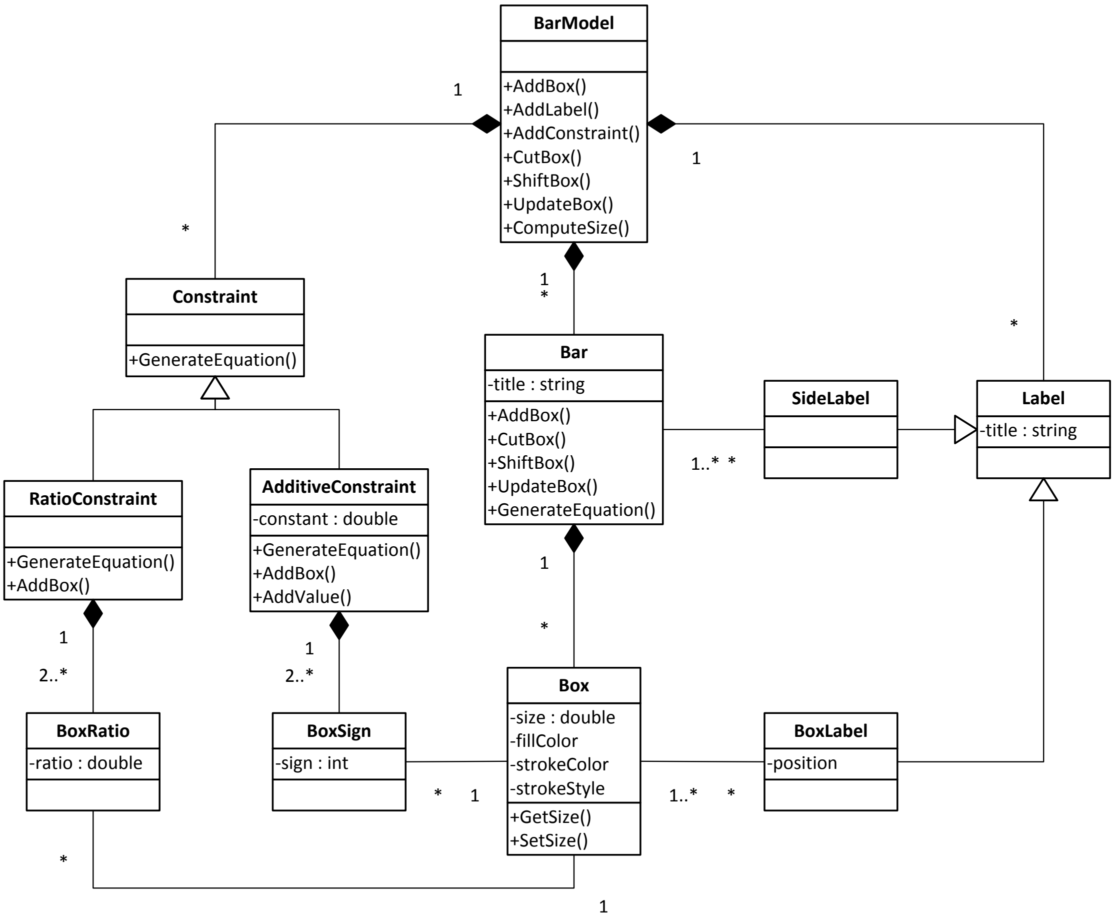


Figure 3: The semantic model

3. The Language

The bar model script is designed for primary school students and teachers, it must be as simple as possible, both its vocabulary and the syntax.

3.1 The Vocabulary

From the semantic model, we can extract

- 4 verbs: add, cut, shift, update;
- 3 entities: bar, box, label;
- 6 properties: title, size, position, fill color, stroke color, stroke style; and
- 7 prepositions: to, into, from, above, below, right, left

These words form the vocabulary of the Bar Model Script.

Another two important concepts are bar specifiers and box specifiers. A bar specifier references to a bar or a range of bars, and a box specifier references to a box or a range of boxes on a bar.

3.2 The Syntax

A bar model script is made up of statements and comments. Each statement or comment occupies a single text line. There are two kinds of statements in bar model script: action statements and constraint statements.

An action statement always starts with one of the 4 verbs followed by one of the 3 nouns. Only a box can be cut, shifted, and updated, so there are only 6 kinds of action statements:

1. Add bar
2. Add box
3. Add label
4. Cut box
5. Shift box
6. Update box

Figure 4 shows the elements of an add box statement, which states that "add a box, whose size is 200, fill color is cyan, and stroke style is dashed, to the first bar, and name it box1". Notice how the size, fill color, and stroke style properties of a box are specified.

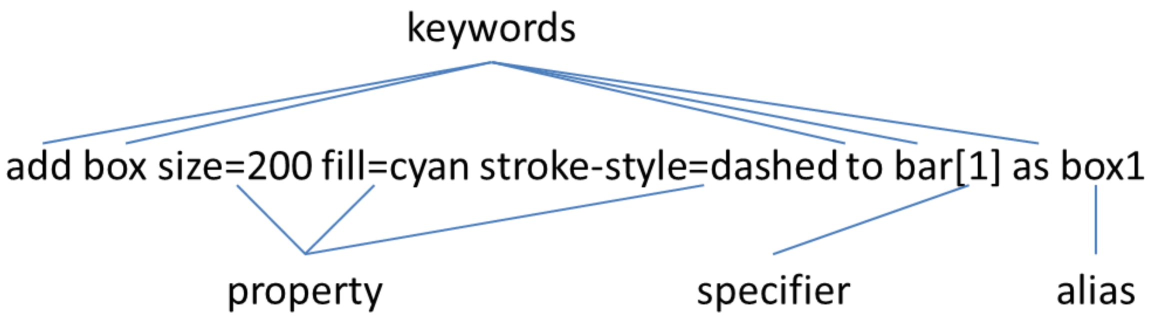

keywords
add box size=200 fill=cyan stroke-style=dashed to bar[1] as box1
property
specifier
alias

Figure 5: The elements of a statement

A constraint statement always starts with a box specifier. The following are a ratio statement and an additive statement.

Ratio statement:

```
Box[2,1] : box[3,1] : box[4,1] = 8 : 5 : 6
```

Additive statement:

```
Box[1,1] + box[2,1] + 20 = box[3,1] - 300
```

Another kind of statements is comment, which occupies a single line and starts with the symbol '#'.

So there are in total 9 kinds of statements: 6 kinds of action statements, 2 kinds of constraint statements, and comment.

3.3 Sample Script

Here is the script of the book problem (Figure 1 is the resulting bar model):

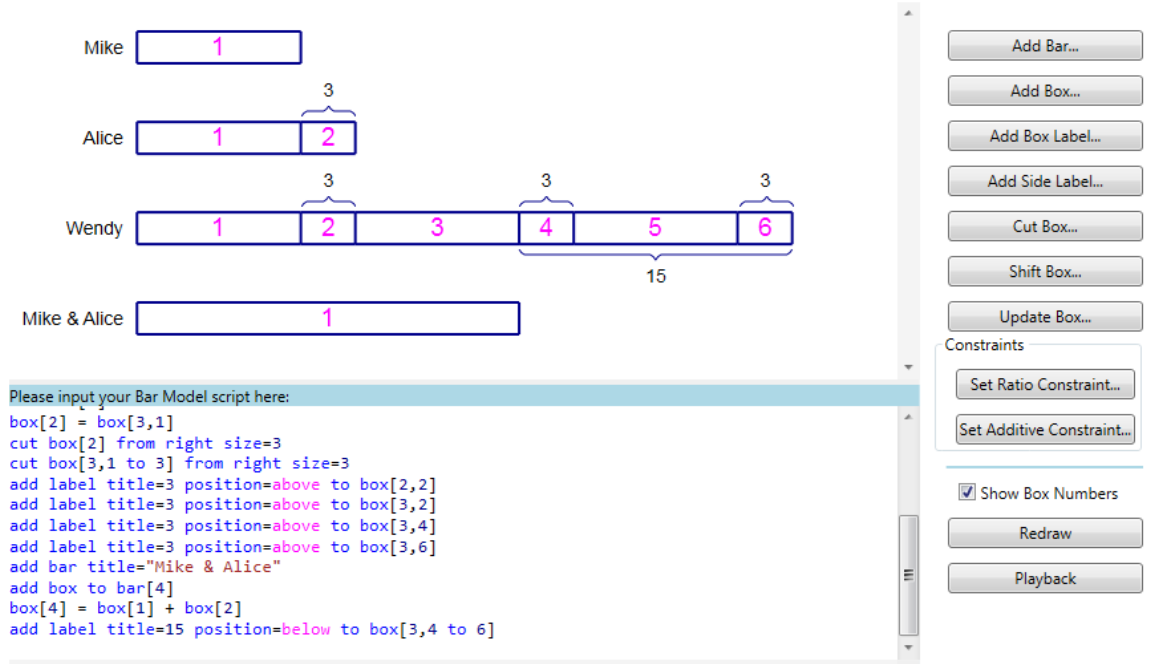
```
add bar title=Mike
add bar title=Alice
add bar title=Wendy
add box to bar[1]
add box to bar[2]
#Mike has 3 less books than Alice
box[1] + 3 = box[2]
add box to bar[3]
#Alice and Mike's total books are 15 less than Wendy's.
box[1] + box[2] = box[3] - 15
#Wendy's books are 3 times as Alice
cut box[3] into 3
box[2] = box[3,1]
cut box[2] from right size=3
cut box[3,1 to 3] from right size=3
add label title=3 position=above to box[2,2]
add label title=3 position=above to box[3,2]
add label title=3 position=above to box[3,4]
add label title=3 position=above to box[3,6]
add bar title="Mike & Alice"
add box to bar[4]
box[4] = box[1] + box[2]
add label title=15 position=below to box[3,4 to 6]
```

List 1: The script of the book problem

4. The Tool

The tool that supports bar model script must have a script editor for users to edit their script, and a view that displays the resulting bar model. It shall update the bar model as the user editing the script. Ideally it should provide dialog boxes or wizards that guide new users to add statements.

Figure 6 shows how such a tool might look like. After the user adds a statement and hits the Enter key, the tool parses the script and redraws the bar model to reflect the changes. The upper 9 buttons on the right are for new users. When a button is clicked, a dialog box or a wizard will pop up and guide the user to add the statement.



Please input your Bar Model script here:

```
box[2] = box[3,1]
cut box[2] from right size=3
cut box[3,1 to 3] from right size=3
add label title=3 position=above to box[2,2]
add label title=3 position=above to box[3,2]
add label title=3 position=above to box[3,4]
add label title=3 position=above to box[3,6]
add bar title="Mike & Alice"
add box to bar[4]
box[4] = box[1] + box[2]
add label title=15 position=below to box[3,4 to 6]
```

Figure 6: The UI of a tool that supports bar model script

Figure 7 shows, at a very high level, how a piece of bar model script eventually becomes a bar model displayed on a computer screen or a SVG file saved on a hard disk. The introduction of the semantic model and the graphic model helps decouple the parser, the code that manipulates the semantic model, and the code that layouts and displays a bar model.

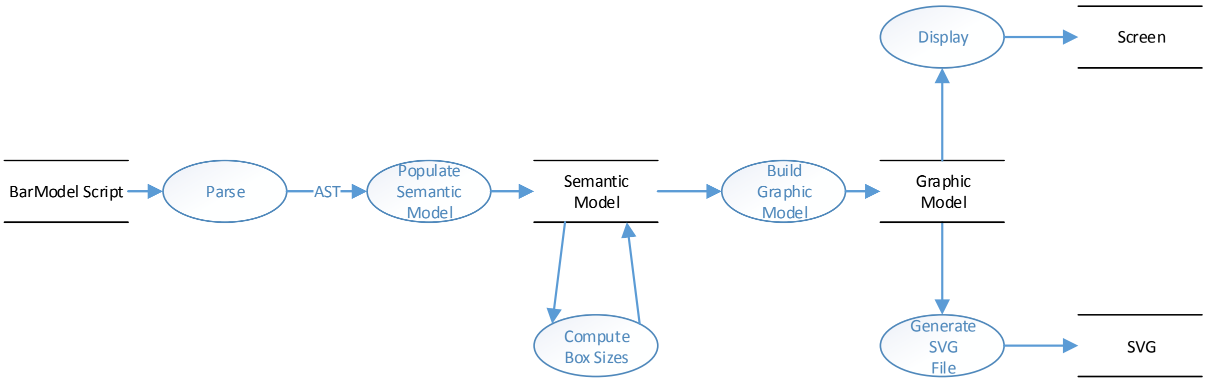


Figure 7: A proposed data flow diagram of a bar model tool

Since the syntax of the language is pretty simple, the parser can be implemented using either delimiter-directed translation [1] or syntax-directed translation [1]. The parser parses a piece of bar model script and generates an AST, which is used to populate the semantic model, which calculates the sizes of the boxes. A graphic model is then built from the semantic model. The graphic model lays out the bars, boxes, and labels. The graphic model should be ideally independent of any specific GUI systems, e.g. WPF, Java Swing, etc. Based on the graphic model, we can display or print the bar model, or save it into an SVG file or other formats.

5. Conclusion

My experiments show that developing a domain specific language that allows students or teachers to “draw” bar models by writing script is a viable approach. The language makes it easier to draw a bar model by removing the burden of worrying about the sizes of boxes when a student or a teacher draws a bar model manually. The bar model script allows a student to example a bar model by stepping forward and backward to deepen his/her understanding about the bar model, and therefore the word problem. A bar model written in bar model script can be saved as an SVG file, which can be easily accessible from a Web browser.

Although bar model script is a simple programming language, it might help inspire the students’ interest in computer programming.

6. References

[1] Martin Fowler with Rebecca Parsons. Domain-Specific Languages. Addison Wesley, 2010.

[2] Yeap Ban Har. Bar Modeling-A Problem-Solving Tool. Marshall Cavendish, 2010.