Properties of Cassowary

Athan Clark

July 12, 2015

Abstract

This paper details the entailed logical properties that a linear constraint solver, annotated with weights and error variables, should have. We hope to use this document as a paper implementation of the automated test suites accompanied with the Haskell and PureScript versions of Cassowary.

1 Equations

The generic form for a linear inequality consists of a unique set of variables summed together, and a constant value:

$$\text{newtype } LinVarMap\ \alpha \ = \ LinVarMap\ (Map\ LinVarName\ \alpha) \tag{LINVARMAP-DEF}$$

$$\begin{aligned}&\text{data } IneqExpr\ \alpha \ = \ IneqExpr\\ &\quad \{\ coeffs\ ::\ (LinVarMap\ \alpha)\\ &\quad ,\ const\ ::\ Rational\ \}\end{aligned} \tag{INEQEXPR-DEF}$$

We segregate the different forms of inequality expressions with newtypes:

$$\text{newtype } Equ\ \alpha \ = \ Equ\ (IneqExpr\ \alpha) \tag{EQU-DEF}$$

$$\text{newtype } Lte\ \alpha \ = \ Lte\ (IneqExpr\ \alpha) \tag{LTE-DEF}$$

$$\text{newtype } Gte\ \alpha \ = \ Gte\ (IneqExpr\ \alpha) \tag{GTE-DEF}$$

Note that the equations are polymorphic in their coefficient type - this will be important when we introduce weights in section 3.

2 Simplex

There are several properties for dual and primal simplex method and Bland's ratio.

$$\alpha = \sqrt{\beta} \tag{1}$$

2.1 Subsection Heading Here

Write your subsection text here.

3 Weights

Weights are implemented as a (non-empty) list of coefficients:

$$\text{newtype } Weight\ \alpha \quad = \quad [\alpha] \tag{WEIGHT-DEF}$$

When an equation using *Rational* values as coefficents gets augmented with a weight (usually with a natural number - $augment\ eq1\ 5$), the coefficients are pushed to that index in an empty stream of 0s; the example just mentioned would stream five 0s before containing the original coefficient.

3.1 Arithmetic

3.1.1 Addition

Instances:

$$(.+.) \ :: \ Weight\ Rational \ \rightarrow \ Weight\ Rational \ \rightarrow \ Weight\ Rational \tag{ADD-SYM}$$

Addition in ADD-SYM is implemented with $unionWith$ - leaving the larger of the two lists intact.

$$(.+.) \quad = \quad \text{unionWith } (+)$$

Lemma: The length of the resulting list, when using addition, is the maximum length of the two lists added.

$$\text{length } (xs\ .+.\ ys) \equiv \max\ (length\ xs)\ (length\ ys)$$

3.1.2 Subtraction

Instances:

$$(.-.) \ :: \ Weight\ Rational \ \rightarrow \ Weight\ Rational \ \rightarrow \ Weight\ Rational \tag{SUB-SYM}$$

$$(.-.) \ :: \ Rational \ \rightarrow \ Weight\ Rational \ \rightarrow \ Rational \tag{SUB-FORGET-1}$$

$$(.-.) \ :: \ Weight\ Rational \ \rightarrow \ Rational \ \rightarrow \ Rational \tag{SUB-FORGET-2}$$

For the first instance SUB-SYM, we use $unionWith$ again:

$$(.-.) \;=\; unionWith\;(-)$$

For SUB-FORGET-1 and SUB-FORGET-2, we sum the list (and forget weight data) before subtracting:

$$\begin{aligned} x\;.-.\;ys &= x \;-\; sum\;ys \\ xs\;.-.\;y &= sum\;xs \;-\; y \end{aligned}$$

3.1.3 Multiplication

Instances:

$$(.*.) :: Weight\;Rational \to Rational \to Weight\;Rational \tag{MUL-DIST-1}$$

$$(.*.) :: Rational \to Weight\;Rational \to Weight\;Rational \tag{MUL-DIST-2}$$

$$(.*.) :: Weight\;Rational \to Weight\;Rational \to Weight\;Rational \tag{MUL-FORGET}$$

MUL-DIST-1 and MUL-DIST-2 naturally distributes the *Rational* multiplied value to every element in the $Weight$ list. In the MUL-FORGET instance, one of the arguments must be forgotten, and is therefore ambiguous for it's behaviour. We leave the implementation for this instance ambiguous, only necessary for implementing substitution.

$$\begin{aligned} xs\;.*.\;y &= (*\;y) \;<\$>\; xs \\ x\;.*.\;ys &= (x\;*) \;<\$>\; ys \end{aligned}$$

3.1.4 Division

Instances:

$$(./.) :: Rational \to Weight\;Rational \to Rational \tag{DIV-FORGET}$$

We will need to divide a coefficient (*Rational*) by a coefficient (possibly a $WeightRational$) in $blandRatioPrimal$, where we have a forgetful instance - divide the constant by the sum of the error coefficients:

$$x\;./.\;ys \;=\; x\;/\;sum\;ys$$

4 Conclusion

Write your conclusion here.