Properties and Semantics of Cassowary

Athan Clark

July 13, 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 in standard form 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.

For general inequalities, we just combine the different forms with a sum type:

$$\begin{aligned} &\text{data } IneqStdForm\ \alpha \ = \\ &\quad\ \ EquStd\ (Equ\ \alpha) \\ &\quad \mid LteStd\ (Lte\ \alpha) \\ &\quad \mid GteStd\ (Gte\ \alpha) \end{aligned} \tag{INEQSTDFORM-DEF}$$

A tableau is then the pair of general constraints, and constraints in basic feasible form:

$$\text{type } \mathit{Tableau}\ \sigma \ = \ (\mathit{Map}\ \mathit{LinVarName}\ \sigma,\ \mathit{IntMap}\ \sigma) \tag{TABLEAU-DEF}$$

Where a tableau is polymorphic in the constraint type used, and split between user variables in basic-normal form, and slack variables in basic-normal form. This is discussed in section 2.1.

2 Simplex

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

2.1 Slack Variables

To generate slack variables, we take our list of arbitrary inequalities and turn them into equations, annotated with the extra slack variable:

$$\text{makeSlackVars} \ :: \ [\mathit{IneqStdForm}\ \alpha] \ \rightarrow \ \mathit{IntMap}\ (\mathit{Equ}\ \alpha)$$

Pivots for both primal and dual simplex will look similar - they take an objective function and a constraint set, then refactor the equations depending on the goal:

$$\begin{aligned} \mathit{pivotPrimal} \ &:: \ (\mathit{Equ}\ \alpha,\ \mathit{Tableau}\ (\mathit{Equ}\ \alpha)) \ \rightarrow \ (\mathit{Equ}\ \alpha,\ \mathit{Tableau}\ (\mathit{Equ}\ \alpha)) \\ \mathit{pivotDual} \ &:: \ (\mathit{Equ}\ \alpha,\ \mathit{Tableau}\ (\mathit{Equ}\ \alpha)) \ \rightarrow \ (\mathit{Equ}\ \alpha,\ \mathit{Tableau}\ (\mathit{Equ}\ \alpha)) \end{aligned}$$

2.2 Substitution

This is the operation we take when replacing an equation.

Lemma: When substituting an equation on itself, the empty equation should result.

$$\begin{aligned} &\forall f \ :: \ \mathit{IneqStdForm}\ \alpha. \\ &\quad \mathit{substitute}\ f\ f \ \equiv \ \emptyset \end{aligned} \tag{1}$$

2.3 Primal

In the primal simplex method, we first select a non-basic variable to become basic, and a constraint out of the set that satisfies the minimum Bland ratio for each pivot.

2.3.1 Next Variable

With primal simplex, first we need to select the next variable by selecting the **most negative** coefficient in the objective function:

$$\text{nextBasicPrimal} \ :: \ Equ \ \alpha \ \rightarrow \ Maybe \ LinVarName$$

Lemma: If all coefficients are positive, then the result is $Nothing$.

$$\forall f \ :: \ Equ \ \alpha \mid all \ (>= 0) \ \$ \ elems \ \$ \ coeffs \ f. \\ isNothing \ \$ \ nextBasicPrimal \ f \qquad \text{(NBP-POS-NULL)}$$

Lemma: If there is one or more negative coefficient, then the result is $Just$ the most minimum of the set.

$$\forall f \ :: \ Equ \ \alpha \mid any \ (< 0) \ \$ \ elems \ \$ \ coeffs \ f. \\ \text{let } x \ = \ fromJust \ (nextBasicPrimal \ f) \\ \text{in } x \ \equiv \ minimum \ (elems \ \$ \ coeffs \ f) \qquad \text{(NBP-NEG-MIN)}$$

Notes:

1. expect α to be comparable to 0
2. α should be lexicographically ordered

2.3.2 Bland Ratio

Bland's ratio is used to determine which constraints would have the most effect on the objective function by simply dividing the constraint's constant value by its coefficient (for some target variable). When using the ratio, you index for the minimum positive ratio.

$$blandRatioPrimal \ :: \ LinVarName \ \rightarrow \ Equ \ \alpha \ \rightarrow \ Maybe \ Rational$$

Lemma: If denominator is 0, then result is $Nothing$.

$$\forall n \ :: \ LinVarName, \ \forall f \ :: \ Equ \ \alpha \mid lookup \ n \ coeff \ f \ = \ 0. \\ isNothing \ (blandRatioPrimal \ f) \qquad \text{(BR-UNDEF-NULL)}$$

Lemma: If the ratio is negative, then the result is $Nothing$.

$$\forall n \ :: \ LinVarName, \ \forall f \ :: \ Equ \ \alpha \\ \mid const \ f \ / \ lookup \ n \ (coeff \ f). \\ isNothing \ (blandRatioPrimal \ f) \qquad \text{(BR-NEG-NULL)}$$

Notes:

1. expect α to be comparable to 0
2. expect α to be a denominator to *Rational* (from *const*)

2.3.3 Next Constraint / Slack Variable

The slack variables are initially unique and enumerated in all the constraints in a tableau, and constitute as basic-normal form. When we pivot with dual or primal simplex, we corrupt the *basic-ness* of this slack variable, and populate it in other equations (and likewise move a constraint from the $IntMap$ to the $Map\ LinVarName$).

We chose the next constraint from the $IntMap$ by finding the least postivie *Bland ratio* - basically the result of dividing the equations constant by the target variable's coefficient (note that in primal simplex, the next variable must be found before the next row).

$$nextRowPrimal\ ::\ LinVarName\ \rightarrow\ IntMap\ (Equ\ \alpha)\ \rightarrow\ Maybe\ Int$$

Lemma: If all coefficients are positive, then the result is $Nothing$.

$$\begin{aligned}&\forall f\ ::\ Equ\ \alpha\ |\ all\ (>= 0)\ \$\ elems\ \$\ coeffs\ f.\\ &\quad isNothing\ \$\ nextBasicPrimal\ f\end{aligned} \tag{NBP-POS-NULL}$$

Lemma: If there is one or more positive ratio, then the result is the minimum of the set.

$$\begin{aligned}&\forall n\ ::\ LinVarName,\ fs\ ::\ IntMap\ (Equ\ \alpha),\ \exists x\ ::\ Int\\ &\quad |\ blandRatioPrimal\ (fs\ !!\ x)\ >\ 0.\\ &\quad \text{let } x'\ =\ fromJust\ (nextRowPrimal\ n\ fs)\\ &\quad \text{in } blandRatioPrimal\ (fs\ !!\ x')\ \equiv\\ &\qquad minimum\ (blandRatioPrimal\ <\$>\ fs)\end{aligned} \tag{NRP-POS-MIN}$$

Notes:

1. We expect α to be comparable to 0
2. α should support (/)

2.4 Dual

3 Weights

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

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

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.

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

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

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

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:

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

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

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

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:

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

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.