

Project description for CSc 22000

Yegor Bryukhov

June 15, 2003

In this project we'll implement a decision procedure Arith [1] that is capable of determining if certain inequality logically follows from the given set of inequalities. This procedure works only on a very restricted type of problems, it's far from being universal.

$$H_1, \dots, H_n \vdash C$$

Here H_i are our hypotheses (given inequalities), C is the target inequality which logical inference from the hypotheses we want to establish. Each inequality compares two polynomials with integer coefficients (and possibly many variables).

The procedure can be split in the following steps:

- (A) We'll use reasoning by contradiction. For this we have to add the inequality D opposite to C to the list of given inequalities and check that new set of inequalities imply false. It will mean that H_1, \dots, H_n, D are incompatible (contradictory).
- (B) Convert every inequality from H_1, \dots, H_n, D to the equivalent \geq -inequality. Let's name the set of new inequalities G_1, \dots, G_{n+1} .
- (C) Every G_i has a form $P_i \geq Q_i$, where P_i and Q_i are polynomials. Normalize every polynomial P_i and Q_i and move constant to the right side of inequality.
- (D) Now you have a set of inequalities of the form $A_i \geq B_i + c_i$ where c_i is an integer constant and A_i and B_i are polynomials without constant component (every additive component has at least one variable as a subterm). Some A_i and B_i may be identical, some may be not. Consider a graph with nodes A_i and B_i (identical occurrences of polynomials contribute only one node), two nodes N_i and N_j are connected by a directed edge from N_i to N_j with weight (label) c if our set of inequalities contains $N_i \geq N_j + c$.
- (E) In the constructed graph find the positive cycle (use one of the algorithms presented in the class). If successful, H_1, \dots, H_n, D are not compatible meaning that C logically follows from the given hypotheses H_1, \dots, H_n . If positive cycle was not found, we can't actually say anything about our original problem (that's why this procedure is called semi-decision procedure).

1 Polynomial normalization.

The input language for this step is the language of expressions, `expr.h` contains classes' signatures that your implementation has to satisfy (my test harness will use this classes to generate tests).

You can perform polynomial normalization in the following way:

1.0.1 Opening the parentheses

Expressions of the forms $(a + b) \cdot c$ or $a \cdot (b + c)$ should be converted to $ac + bc$ and $ab + ac$ respectively. Here you have to make sure that you've opened *all* parentheses.

1.0.2 Normalizing every monomial

Then we normalize every monomial (additive components of the polynomial), monomial is just a product of integer constants and variables. You have to multiply all constants and sort variables in the increasing order. Hint: for monomials you can use special representation (not the original representation via **BinOp**, **Const** and **Var**) - integer that represents the constant and ordered list/array of variables.

1.0.3 Contracting the similar monomials and ordering the result

Now as we have all monomials normalized, we have to sum up similar monomials (with the same lists of variables) and order the resulting monomials. You can actually do it in one step (sorting and contracting). *You are done - polynomial is normalized.*

References

- [1] T. Chan. An algorithm for checking PL/CV arithmetic inferences. In G. Goos and J. Hartmanis, editors, *An Introduction to the PL/CV Programming Logic*, volume 135 of *Lecture Notes in Computer Science*, appendix D, pages 227–264. Springer-Verlag, 1982.