Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 267 (2010) 73–87

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Static Analysis by Abstract Interpretation: A Mathematical Programming Approach

Eric Goubault[a,2,1], Stéphane Le Roux[b,3], Jeremy Leconte[c,4], Leo Liberti[b,5] and Fabrizio Marinelli[d,6]

[a] *CEA Saclay, France*

[b] *LIX, École Polytechnique, 91128 Palaiseau, France*

[c] *Dep. Info., ENS, 45 rue d'Ulm, 75005 Paris, France*

[d] *DIIGA, Univ. Politecnica delle Marche, Ancona, Italy*

Abstract

Static analysis of a computer program by abstract interpretation helps prove behavioural properties of the program. Programs are defined by means of a forward collecting semantics function relating the values of the program variables during the execution of the program. The least fixed point of the semantics function is a program invariants providing useful information about the program's behaviour. Mathematical Programming is a formal language for describing and solving optimization problems expressed in very general terms. This paper establishes a link between the two disciplines by providing a mathematical program that models the problem of finding the least fixed point of a semantics function. Although we limit the discussion to integer affine arithmetic semantics in the interval domain, the flexibility and power of mathematical programming tools have the potential for enriching static analysis considerably.

Keywords: Guaranteed smallest code invariant, constraints, bilinear MINLP, policy iteration, branch-and-bound.

1 Introduction

Static Analysis by Abstract Interpretation (SAAI) was introduced by Cousot and Cousot in [9] and [10], and further developed, *e.g.*, in [11]. It is widely used in static

[1] We thank David Monniaux and Nicolas Halbwachs for many enlightening discussions and precious suggestions. This work was partially supported by grants: Île-de-France research council (post-doctoral fellowship), System@tic consortium ("EDONA" project), ANR 07-JCJC-0151 "Ars", ANR 08-SEGI-023 "Asopt", Digiteo Emergence "Paso".

[2] Email: eric.goubault@cea.fr

[3] Email: leroux@lix.polytechnique.fr

[4] Email: jeremy.leconte1@ens.fr

[5] Email: liberti@lix.polytechnique.fr

[6] Email: marinelli@diiga.univpm.it

doi:10.1016/j.entcs.2010.09.007

analysis of imperative programs to approximate the behaviour of a program, for instance in terms of its variable environments. Given a program, one builds a forward collecting semantics function expressing statically how the environments at a given control point depend dynamically on other control points. This function has a least fixed point (lfp), which is the "best" information that the function may give about the program. Usual methods to compute the lfp range from increasing sequences of under-approximations (relying on Kleene fixed point theorem), decreasing sequences of over-approximations (relying on Tarski fixed point theorem), or both methods combined (relying on widening). The Policy Iteration (PI) method was introduced on the interval domain in [7], further developed in [1] and extended to other (relational) domains in [12,2]. PI computes the lfp when the semantics function is non-expansive in the sup norm, and a fixed point otherwise. Another PI method on intervals was described in [14] and later generalized to relational domains in [15].

Computing the lfp of the semantics function is quite naturally an optimization problem. Mathematical Programming (MP) is a declarative language that describes the solution of very general optimization problems [26]. An MP consists of a set of parameters (encoding the problem input prior to the solution process), a set of decision variables $x \in \mathbb{R}^n$ (encoding the problem output after the solution process), an objective function $f : \mathbb{R}^n \to \mathbb{R}$, a set of equality and/or inequality constraints $g(x) \leq 0$ with $g : \mathbb{R}^n \to \mathbb{R}^m$, a set of variable bounds $x^L \leq x \leq x^U$ and a set of integrality constraints $\forall j \in Z \ x_j \in \mathbb{Z}$ [19]. MPs are categorised according to the nature of the solution as: Linear Programs (LPs), Nonlinear Programs (NLPs), Mixed-Integer Linear Programs (MILPs), Mixed-Integer Nonlinear Programs (MINLPs), each category having dedicated solution algorithms.

We study the following decision problem.

> Static Analysis by Abstract Interpretation Problem (SAAIP). Given a program written in the language $\mathbb{P}$ (defined in Sect. 2) does its semantics function (defined in Sect. 3.1) have a finite lfp?

SAAIP is actually a problem schema, because it can be parametrized by the type of abstraction used to overapproximate the concrete program semantics. This paper aims to establish a strong link between SAAI and MP by formalizing the search for the lfp by means of a MP formulation. When the semantics function only includes integer convex arithmetic, the MP turns out to be a MINLP with convex objective and constraints, which can always be solved to optimality in worst-case exponential time [4]. For semantics functions including continuous and/or nonconvex arithmetic, the resulting MINLP can be solved to ε-approximation using the spatial Branch-and-Bound (sBB) algorithm [3]. The MP standard toolbox also includes several practically efficient heuristic methods [5,21] which find non-optimal but feasible solutions: in the present setting, these correspond to fixed points without guarantee of minimality, which may provide useful information about the program. The flexibility of MP can hardly be underestimated: variable relations, for example, simply give rise to additional constraints which can just be adjoined to the current MP formulation.

We set the framework by exemplifying the use of MP in SAAI limited to a