Science of Computer Programming 120 (2016) 1–24

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Efficiently intertwining widening and narrowing ☆,☆☆

Gianluca Amato [a], Francesca Scozzari [a], Helmut Seidl [b,*], Kalmer Apinis [c], Vesal Vojdani [c]

[a] Università di Chieti-Pescara, Italy
[b] Technische Universität München, Germany
[c] University of Tartu, Estonia

ARTICLE INFO

Article history:
Received 25 February 2015
Received in revised form 1 December 2015
Accepted 1 December 2015
Available online 7 January 2016

Keywords:
Static program analysis
Fixpoint iteration
Constraint solving
Widening and narrowing
Termination

ABSTRACT

Accelerated fixpoint iteration by means of widening and narrowing is the method of choice for solving systems of equations over domains with infinite ascending chains. The strict separation into an ascending widening and a descending narrowing phase, however, may unnecessarily give up precision that cannot be recovered later. It is also unsuitable for equation systems with infinitely many unknowns — where local solving must be used.
As a remedy, we present a novel operator $\boxtriangledown$ that combines a given widening operator ∇ with a given narrowing operator Δ. We present adapted versions of round-robin and worklist iteration as well as local and side-effecting solving algorithms for the combined operator $\boxtriangledown$. We prove that the resulting solvers always return sound results and are guaranteed to terminate for monotonic systems whenever only finitely many unknowns are encountered.

1. Introduction

From an algorithmic point of view, static analysis typically boils down to solving systems of equations over a suitable domain of values. The unknowns of the system correspond to the invariants to be computed, e.g., for each program point or for each program point in a given calling context or instance of a class. For abstract interpretation, complete lattices were proposed as domains of abstract values [15]. In practice, partial orders can be applied which are not necessarily complete lattices as long as they support an effective binary upper bound operation. This is the case, e.g., for polyhedra [20], zonotopes [23] or parallelotopes [1]. Still, variants of Kleene iteration can be used to compute solutions. Right from the beginnings of abstract interpretation, it became clear that many interesting invariants can only be expressed by domains with *infinite* strictly ascending chains. In the presence of possibly infinite strictly ascending chains, naive Kleene iteration is no longer guaranteed to terminate. For that reason, Cousot and Cousot proposed a *widening* iteration to obtain a valid invariant or, technically speaking, a *post* solution which subsequently may be improved by means of a *narrowing* iteration [14,18]. The widening phase can be considered as a Kleene iteration that is accelerated by means of a widening operator

☆ This article extends and generalizes results presented by [4] by integrating key ideas from [2].

☆☆ This work was partially supported by the ARTEMIS JU under grant agreement n° 269335 (MBAT) and from the German Science Foundation (DFG) project OpIAT SE 551/13-2. This work was funded by institutional research grant IUT2-1 from the Estonian Research Council.

* Corresponding author.
E-mail addresses: gamato@unich.it (G. Amato), fscozzari@unich.it (F. Scozzari), seidl@in.tum.de (H. Seidl), kalmer.apinis@ut.ee (K. Apinis), vesal.vojdani@ut.ee (V. Vojdani).

http://dx.doi.org/10.1016/j.scico.2015.12.005
0167-6423/

to enforce that only finitely many increases of values occur for every unknown. While enforcing termination, it may result in a crude over-approximation of the invariants of the program. In order to compensate for that, the subsequent narrowing iteration tries to improve a given post solution by means of a downward fixpoint iteration, which again may be accelerated, in this case by means of a narrowing operator.

Trying to recover precision once it has been thrown away, though, may not always possible (see, e.g., [31] for a recent discussion). Some attempts try to improve precision by reducing the number of points where widening is applied [12,8], while others rely on refined widening or narrowing operators (see, e.g., [43,10]). Recently, several authors have focused on methods to guide or stratify the exploration of the state space [26,25,28,39,33], including techniques for automatic transformation of irregular loops [29,42] or by repeating the widening/narrowing phases starting from a different initial state [31]. An interesting novel idea is to add a third phase where fixpoint iteration is started from scratch, but the best known previously computed upper bound for each unknown is exploited to improve intermediate values [13]. In that third phase, yet another operator, namely a *dual* narrowing is applied to enforce termination.

Our approach here at least partly encompasses those of [12] and [8], while it is complementary to the other techniques and can, potentially, be combined with these. Our idea is to avoid postponing narrowing to a second phase after a post solution has been computed, in which all losses of information have already occurred and been propagated. Instead, we attempt to systematically improve the current information immediately by downward iterations. This means that increasing and decreasing iterations are applied in an *interleaved* manner. A similar idea is already used by syntax-directed fixpoint iteration engines as, e.g., in the static analyzers Astrée [5,19] and Jandom [2]. The Astrée analyzer follows the syntax of the program and performs a fixpoint iteration at every detected loop, consisting of a widening iteration followed by a narrowing iteration. Nesting of loops, therefore, also results in nested iterations. In order to enforce termination, *ad hoc* techniques such as restrictions to the number of updates are applied. Here, we explore iteration strategies in a generic setting, where no *a priori* knowledge of the application is available, and provide sufficient conditions for when particular fixpoint algorithms are guaranteed to terminate.

As we concentrate on the algorithmic side and application-independent generic solvers, we use the original notions of narrowing [14,18], rather than more elaborate definitions [17,13] which refer to the *concrete* semantics of the system to be analyzed. The classic formulation of narrowing requires right-hand sides of equations to be *monotonic* so that the second iteration phase is guaranteed to be *descending* and thus improving. Accordingly, the narrowing operator is guaranteed to return meaningful results only when applied in *decreasing* sequences of values. The assumption of monotonicity of right-hand sides, even disregarding the occurrences of widening and narrowing operators, may not always be met. Monotonicity can no longer be guaranteed, e.g., when compiling context-sensitive inter-procedural analyses into systems of equations [21,3].

Example 1. For some domain $\mathbb{D}$, consider the system of equations

$$h(x) = g(f(x)) \quad (x \in \mathbb{D})$$

over the unknowns $f(d), g(d), h(d)$, $d \in \mathbb{D}$. Similar systems of equations are used by a context-sensitive inter-procedural analysis when separate unknowns are introduced for every possible (abstract) calling context of a procedure. In the given example system, the value of $f(x)$ determines the unknown $g(x')$ whose value contributes to the value of $h(x)$. Now consider a domain $\mathbb{D}$ with two distinct elements $a \sqsubset b$ and consider the assignments ρ_1, ρ_2 with

$$\begin{array}{lll} \rho_1[f(a)] = a & \rho_1[g(a)] = b & \rho_1[g(b)] = a \\ \rho_2[f(a)] = b & \rho_2[g(a)] = b & \rho_2[g(b)] = a \end{array}$$

which otherwise agree for every unknown. Then $\rho_1 \sqsubseteq \rho_2$, but the right-hand side of $h(a)$, when evaluated over ρ_1 results in b, while an evaluation over ρ_2 results in a. Accordingly, the given right-hand side cannot be monotonic. □

For inter-procedural analysis with infinite domains, the resulting equation systems may also be *infinite*. These can be handled by *local* solvers. Local solvers query the value of an *interesting* unknown and explore the space of unknowns only as much as required for answering the query. For this type of algorithm, neither the set of unknowns to be evaluated nor their respective dependences are known beforehand. Accordingly, the values of fresh unknowns that have not yet been encountered may be queried in the narrowing phase. As a consequence, the rigid two-phase iteration strategy of one widening phase followed by one narrowing phase can no longer be maintained.

In order to cope with these obstacles, we introduce an operator $\boxtriangledown$ which is a combination of a given widening ∇ with a given narrowing operator Δ and show that this new operator can be plugged into any solver of equation systems, be they monotonic or non-monotonic. The $\boxtriangledown$ operator behaves like narrowing as long as the iteration is descending and like widening otherwise. As a result, solvers are obtained that return reasonably precise post solutions in one go, given that they terminate.

Termination, then, is indeed an issue. We present two example systems of monotonic equations where standard fixpoint algorithms, such as round robin or worklist iteration, fail to terminate when enhanced with the new operator. As a remedy, we develop a variant of round robin as well as a variant of worklist iteration which in absence of widening and narrowing