Computing Quadratic Invariants with Min- and Max-Policy Iterations: A Practical Comparison

Pierre Roux\textsuperscript{1,2} and Pierre-Loïc Garoche\textsuperscript{1}

\textsuperscript{1} ONERA – The French Aerospace Lab, Toulouse, France
\textsuperscript{2} ISAE, Toulouse, France

Abstract. Policy iterations have been known in static analysis since a small decade. Despite the impressive results they provide – achieving a precise fixpoint without the need of widening/narrowing mechanisms of abstract interpretation – their use is not yet widespread. Furthermore, there are basically two dual approaches: min-policies and max-policies, but they have not yet been practically compared.

Multiple issues could explain their relative low adoption in the research communities: implementation of the theory is not obvious; initialization is rarely addressed; integration with other abstraction or fixpoint engine not mentioned; etc. This paper tries to present a Policy Iteration Primer, summarizing the approaches from the practical side, focusing on their implementation and use.

We implemented both of them for a specific setting: the computation of quadratic templates, which appear useful to analyze controllers such as found in civil aircrafts or UAVs.

Keywords: abstract interpretation, policy iteration, linear systems with guards, quadratic invariants, ellipsoids, semidefinite programming.

1 Introduction

Abstract interpretation is now commonly used as a framework to describe static analyses of programs. The collecting semantics, i.e., set of reachable states, has first to be characterized as a fixpoint computation; then abstract domains allow to perform in the abstract the fixpoint computation and obtain a sound over-approximation of the concrete fixpoint.

The most famous approach of this fixpoint over-approximation is based on a Kleene fixpoint computation using widening and narrowing mechanisms \cite{5}. The iteration process starts from an over-approximation, in the abstract domain, of the initial states, then it performs a sequence of computations using the abstract transfer function of the program. These iterations can be understood as local computations: each statement of the program is considered one by one until the global fixpoint is reached. Widening operators are then used while computing the iterates to ensure convergence. Narrowing helps to recover precision lost by widening steps: it is used once a postfixpoint is obtained to regain precision.
Another approach was more recently introduced in the static analysis community: policy \(^1\) iterations \([4,8,9]\). The idea is to exactly solve the fixpoint equation for a given abstract domain when specific conditions are satisfied using appropriate mathematical solvers. For example when both the abstract domain and the fixpoint equation use linear equations, then linear programming could be used to compute the exact solution without the need of widening and narrowing \([8,9]\). Similarly when the function and the abstract domain are at most quadratic, semi-definite programming (SDP) could be used \([11,12]\). In practice, the abstract domains should be rephrased as template domains, i.e., a finite a-priori-known set of functions that will be bounded thanks to the mathematical solvers.

This second approach is also very useful when abstract domains are not fitted with a lattice structure. For example ellipsoids, are not fitted with such: usually, there is no smallest (for inclusion order) ellipsoid containing two other given ellipsoids. But given a (fixed) set of quadratic templates, policy iterations could bound them. Policy iterations over quadratic templates is then a good approach to compute such invariants, that are not well suited for Kleene iterations.

We are interested in analyzing control command software, more specifically the ones found in UAVs or civil aircrafts. Most of them are based on well known principles of control theory: linear controllers. In general these controllers do not admit simple linear inductive invariants, but control theorists know for long \([3,16]\) that such systems are stable if and only if they admit a quadratic invariant. Therefore we are interested in computing these invariants on such linear systems.

Few static analysis work rely on quadratic invariants to bound linear systems \([1,2,6,7,11,18,19]\). In particular, ellipsoids of dimension two are used in the famous Astrée tool \([6,7]\).

About policy iterations, two different “schools” exist in the static analysis community. The “French school” \([1,4,8,12]\) offers to iterate on min-policies, starting from an over-approximation of a fixpoint and decreasing the bounds until the fixpoint is reached. The “German school” \([9,10,12]\) in contrary operates on max-policies, starting from bottom and increasing the bounds until a fixpoint is reached. While the first can be interrupted at any point leaving a sound over-approximation, the second approach requires to wait until the fixpoint is reached to provide its result.

Clearly those two approaches rely on comparable fundamentals, but no work actually compares them in practice. Furthermore their description is highly theoretical and not supported by actual implementation performing analyses on code. A few issues, that particularly matter when targeting a practical implementation, were also not actually addressed such as the initial state of the iterations, the use of unsound tools to perform numerical computations or the integration with other abstractions.

This paper tries to give a practical definition for both approaches and presents our experiments to compare them when inferring quadratic invariants for linear controllers. All the analyses have been implemented and all results are obtain without any other information than the code.

Section 2 details the state of the art, i.e., the definition of template domains, min- and max-policies. Section 3 provides some details on our implementation.

\(^1\) The word *strategy* is also used in the literature for *policy*, with equivalent meaning.