An Efficient Algorithm for Computing MHP Information for Concurrent Java Programs

Gleb Naumovich, George S. Avrunin, and Lori A. Clarke

Laboratory for Advanced Software Engineering Research
Department of Computer Science
University of Massachusetts at Amherst
Amherst, MA 01003-6410
{naumovic, avrunin, clarke}@cs.umass.edu

Abstract. Information about which statements in a concurrent program may happen in parallel (MHP) has a number of important applications. It can be used in program optimization, debugging, program understanding tools, improving the accuracy of data flow approaches, and detecting synchronization anomalies, such as data races. In this paper we propose a data flow algorithm for computing a conservative estimate of the MHP information for Java programs that has a worst-case time bound that is cubic in the size of the program. We present a preliminary experimental comparison between our algorithm and a reachability analysis algorithm that determines the "ideal" static MHP information for concurrent Java programs. This initial experiment indicates that our data flow algorithm precisely computed the ideal MHP information in the vast majority of cases we examined. In the two out of 29 cases where the MHP algorithm turned out to be less than ideally precise, the number of spurious pairs was small compared to the total number of ideal MHP pairs.

1 Introduction

Information about which statements in a concurrent program may happen in parallel (MHP) has a number of important applications. It can be used for detecting synchronization anomalies, such as data races [5], for improving the accuracy of various data flow analysis and verification approaches (e.g. [14][19]), for improving program understanding tools, such as debuggers, and for detecting program optimizations. For example, in optimization, if it is known that two threads of control will never attempt to enter a critical region of code at the same time, any unnecessary locking operations can be removed.

In general, the problem of precisely computing all pairs of statements that may execute in parallel is undecidable. If we assume that all control paths in all threads of control are executable, then the problem is NP-complete [21]. In this paper, we call the

* This research was partially supported by the Defense Advanced Research Projects Agency and the Air Force Research Laboratory/IFTD under agreement F30602-97-2-0032, and by the National Science Foundation under Grant CCR-9708184. The views, findings, and conclusions presented here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, the Air Force Research Laboratory/IFTD, the National Science Foundation, or the U.S. Government.

© Springer-Verlag Berlin Heidelberg 1999
solution with this assumption the ideal MHP information for a program. In practice, a trade-off must be made where, instead of the ideal information, a conservative estimate of all MHP pairs is computed. In this context a conservative estimate contains all the pairs that can actually execute in parallel but may also contain spurious pairs. The precision of such approaches can be measured by comparing the set of pairs computed by an approach with the ideal set, if the latter is known.

In this paper we propose a data flow algorithm for computing a conservative estimate of MHP information for Java programs that has a worst-case time bound that is cubic in the size of the program. In the rest of this paper we refer to this algorithm as the MHP algorithm. To evaluate the practical precision of our algorithm, we have carried out a preliminary experimental comparison between our algorithm and a reachability-based algorithm that determines the ideal MHP information for concurrent Java programs. Of course, since this reachability algorithm can only be realistically applied to small programs, our experiment was restricted to programs with a small number of statements. This initial experiment indicates that our algorithm precisely computed the ideal MHP information in the vast majority of cases we examined. In the two out of 29 cases where the MHP algorithm turned out to be less than ideally precise, the number of spurious pairs was small compared to the total number of ideal MHP pairs.

Several approaches for computing MHP information for programs using various synchronization mechanisms have been suggested. Callahan and Subhlok [4] proposed a data flow algorithm that computes, for each statement in a concurrent program with post-wait synchronization, the set of statements that must be executed before this statement can be executed (B4 analysis). Duesterwald and Soffa [6] applied this approach to the Ada rendezvous model and extended B4 analysis to be interprocedural. Masticola and Ryder [15] proposed an iterative approach that computes a conservative estimate of the set of pairs of communication statements that can never happen in parallel in a concurrent Ada program. (The complement of this set is a conservative approximation of the set of pairs that may occur in parallel.) In that work, it is assumed initially that any statement from a given process can happen in parallel with any statement in any other process. This pessimistic estimate is then improved by a series of refinements that are applied iteratively until a fixed point is reached. This approach yields more precise information than the approaches of Callahan and Subhlok and of Duesterwald and Soffa. Masticola and Ryder show that in the worst case the complexity of their approach is $O(S^5)$, where $S$ is the number of statements in a program.

Recently, Naumovich and Avrunin [17] proposed a data flow algorithm for computing MHP information for programs with a rendezvous model of concurrency. Although the worst-case complexity of this algorithm is $O(S^6)$, their experimental results suggest that the practical complexity of this algorithm is cubic or less in the number of program statements. Furthermore, the precision of this algorithm was very high for the examples they examined. For a set of 132 concurrent Ada programs, the MHP algorithm failed to find the ideal MHP information in only 5 cases. For a large majority of the examples, the MHP algorithm was more precise than Masticola and Ryder’s approach.

The MHP algorithm described in this paper is similar in spirit to the algorithm proposed for the rendezvous model but has a number of significant differences prompted by the difference between the rendezvous-based synchronization in Ada and the shared variable-based synchronization in Java. First, the program model for Java is quite differ-

1 The size of the program model is $O(S^2)$ in the worst case, and the worst-case complexity of the algorithm is cubic in the size of the program model. It appears that in practice the size of the program model is linear in the number of program statements $S$ [17].