The query-based debugger model and implementation can be extended in a number of ways. First, the query model could be extended to include projection on “columns” of the result corresponding to the domain variables. The query model could allow computations that involve the result objects. For example, it would be useful to write queries that calculate the average length of certain lists. Although it is possible to do this now by iterating over the tuples in the query result, integrating such functionality into the query model could make such computations easier to express and potentially more efficient. In this way, the system would no longer need to construct and then consume the output tuples.

A second avenue for the future work is to extend the dynamic queries to handle collections and system classes. System classes can be instrumented using a different load-time adaptation method (section 4.3.6.2). Handling of collections presents a more complicated challenge. First, collections are usually based on arrays, and the current implementation does not handle arrays. Second, collection interfaces are exported through accessor methods. For such methods the automatic change set determination may present problems. Third, collection support should be coupled with individual instance identification in queries. Only some collections may belong to the debugged program while most would be a part of the library code. Without filtering out these extraneous collections, the query results may be confusing and the debugger may become inefficient.

The rest of this chapter discusses two open problems: efficient automatic change set generation and safe query reevaluation. Section 6.2.2 also outlines the ongoing research on distributed query-based debugging.

6.1 AUTOMATIC CHANGE SETS

Though the current debugger implementation automatically determines change sets of queries without method invocations (section 4.3.3), the general problem of automatic change set determination is not easy to solve. The first problematic area lies with method invocations that make it harder to determine which fields and objects affect the query result. The second problem
is reference chains that introduce additional overhead in chain splitting. This section discusses possible solutions for both problems.

6.1.1 Automatic Change Sets for Method Invocations

Automatically determining a change set of the query when the query invokes methods is a difficult problem. To determine a change set, the system needs to perform a static or dynamic analysis of method invocations and field accesses. Such analyses would find the methods invoked during the query evaluation and objects referenced by these methods. These objects belong to the change set. The conservative starting point of the analysis are all objects transitively reachable through the fields of the domain class, and through all static fields of classes in the system. The analysis attempts to reduce this set.

The result of a static analysis that determines the methods involved in a query evaluation is equivalent to a static program slice [192] at the end of the query expression evaluation. To determine methods invoked, the system needs to use a type inference algorithm [6] because, in dynamically dispatched languages, a target method depends on the receiver type. The debugger already knows types of domain objects and their fields, allowing the system to determine the first methods invoked. If an ideal type inference engine were available, the system would know exactly which methods are called. Unfortunately, even the best type-inference methods require large amounts of memory, are relatively slow, and may not be precise enough to reduce the change set into an efficient size for monitoring purposes.

The second part of the static analysis would extract objects and fields accessed from the executed methods and so would determine which objects to monitor. The relationship between these objects and concrete instances of the domain class would need further identification as discussed in section 6.1.2.

The system could also find change sets by using dynamic analysis. In this case, the debugger would determine a change set at the time when a query is evaluated. Unlike the static analysis, no type inference algorithm would be needed, since the system would wait to detect method invocations until runtime. However, the system would incur a runtime overhead with each query reevaluation.

The dynamic analysis determines the query change set by marking all objects accessed during the evaluation. The marking can be done using a number of techniques proposed in the research on data breakpoints and on