Static Ownership Inference for Reasoning Against Concurrency Errors

Ana Milanova and Yin Liu

Department of Computer Science , Rensselaer Polytechnic Institute, Troy, NY

Motivation

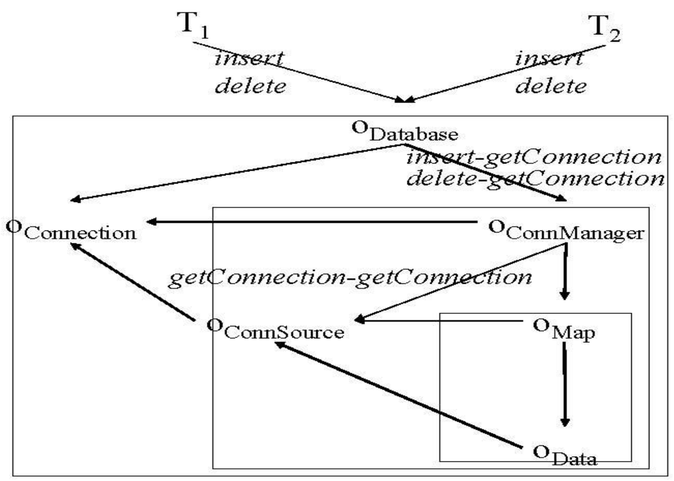
- Shared memory concurrency
  - Increasingly relevant
  - Difficult to reason about
- Data races
  - Detection
  - Understanding structure of sharing
- Current research primarily on dynamic techniques
  - Lockset, happens before, other approaches
  - Unsafe, time and space overhead, delay error reports
- Research on static techniques is underrepresented!
  - Algorithms for static detection of data races
  - Reasoning about structure of object sharing in OO programs

Our Approach

- A new view of OO programs
  - Emphasis on object structure and transfer of control
  - How threads access shared objects
  - Connection between data and control transfer
- Reasoning about concurrency with ownership
  - Owners as dominators: *no representation exposure*
  - Ownership is useful in reasoning about concurrency
- Overview of approach
  - Annotated object graph
  - Ownership inference
  - Static analysis for data race detection

Annotated Object Graph

- Runtime object graph $o \rightarrow o'$
  - Object access relationships during program execution
- Object graph has edge
  - A field *f* of *o* refers to *o'*
  - A local variable *r* in method *m* refers to *o'* in invocation *o.m*
- Edge annotation
  - Transfer of control between objects
  - Annotation m_1-m_2 on edge o_1-o_2 : $o_1.m_1$ calls $o_2.m_2$



Ownership Inference

- Implementation-level ownership
  - *Owners-as-dominators (FLAP without parameterization)*
  - *o owns o': o* is immediate dominator of *o'* in the object graph
- Approximate annotated object graph
  - Safe static approximation of structure (runtime object accesses)
  - Safe approximation of control transfer
- Ownership inference
  - Reason on approximate graph
  - *Dominancy boundary of o :* portion of object graph that is dominated by *o*

Reasoning Against Data Races

- Main intuition
  - To have data race on o_n : object race on the owner of o_n, o_k
  - If accesses to o_k appropriately synchronized: no race on o_n
- Data race detection
  - Start from a potential data race
  - Trace control transfer annotations backwards
    - Ownership hierarchy
    - Take into account synchronization on owners

The Structure Of Object Sharing

- Thread owned objects
  - Objects owned by their creating threads
- Central shared objects
  - Objects directly accessed by two or more threads
  - Deep dominance boundaries: deep data race
- Distributed shared objects
  - Objects created in one thread, passed to boundary of another object in another thread. Difficult to reason about
- Our work: use static analysis to study the structure of sharing in real-world applications

Conclusion And Future Work

- Construction of annotated object graph
  - Novel representation of objects and object accesses
  - Structural information
  - Control transfer information connected with data
- Static analysis algorithms for data race detection
  - Ownership inference on annotated object graph
  - May lead to easier detection of data races
- Study the structure of sharing in real-world applications

Related Work

- Race detection dynamic and hybrid approaches
  - *Sen [PLDI08], Park and Sen [FSE08], many other*
- Static race detection
  - *Naik et al. [PLDI06]*: uses precise points-to and other analyses
- Ownership in reasoning about concurrency
  - *Von Praun and Gross [OOPSLA01]*: thread ownership, dynamic object race detection
  - *Boyapati et al. [OOPSLA02]*: ownership type system