Automating Proofs of Coarse Transaction Properties

Trek Palmer and Eliot Moss
Coarse-grained Transactions

- Coarse-grained transactions promise high concurrency among transactions
- Increases programmer's burden
  - Conflicts must be specified and correct
  - This is difficult
- Conflicts are actually over the abstraction
  - Checking needs a description of it
Abstractions are good

- Assume implementation handles low-level data conflicts *within* each method
- Inherent transaction conflicts are between methods of the *abstraction*
  - Abstraction simpler than implementation
  - Easier to reason about
• Need a simple specification language
  o Verify correctness of locking ahead of time
  o Describe abstraction and locking
  o Automatically prove correct or discover counterexamples (errors)
ACCLAM Language in a slide or two

- Describe Java-like code
- Verification problems translatable to SAT
- Must be feasible for SAT
  - No loops, recursion
  - Alternate constructs for expressiveness
    - Forall bulk update
    - Mapping data type
- Support (some) non-deterministic specs
  - choose statement
Differences from Java:
  - Relation/Mapping types
  - Simple Reductions
  - Invariants

Descriptions processed into SAT to verify:
  - Correctness of inverses
  - Correctness of locking protocols
  - Correctness of language components
    - Forall, non-deterministic choice
ACCLAM example, a Set

```java
model Set<T> {
    boolean in[T]; //a mapping
    int size = count(T idx ; in[T]);

    void add(T elem) { in[elem] = true; }
    void delete(T elem) { in[elem] = false; }
    int size() { return size; }

    boolean contains(T elem){
        return in[elem];
    }
}
```
Results

- **Set**
  - Time to prove conflicts correct: 1-70 ms
  - Time to prove inverses correct: 1-20 ms
  - Average conflict #clauses: 3541
  - Average conflict #vars: 1148

- **OrderedSet**
  - Time to prove conflicts correct: 1-181 ms
  - Time to prove inverses correct: 1-183 ms
  - Average conflict #clauses: 8557
  - Average conflict #vars: 2279
Any Questions?
void add(T obj)
{
    boolean present = in[obj];
    in[obj] = true;
}
onabort {
    in[obj] = present;
}
OrderedSet Example

```java
model OrderedSet<T> extends Set<T> {
    int rank[T]; //The rank/order
    T next[int]; T prev[int];
    void add(T elem) {
        in[elem] = true;
        forall(T z ; (next[rank[z]] ==
                        next[rank[elem]]) &&
                        (rank[z] < rank[elem])) {
            next[rank[z]] = elem;
        }
    }
}
```
Conflict Predicate Example

add(x) * add(y) { x == y }
add(x) * remove(y) { x == y }
add(x) * z = find(y) { x == y }
//state based predicates
add(x) * z = isEmpty() { sz == 0 }
add(x) * z = size() { !in[x] }