Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

Project CASA:
Help needed with C++ to Java translation

1. Proposition

Project CASA is about translating C++ code into functional Java counterpart. The bulk of the project is completed, however some parts are beyond the abilities of the team members.

Thus, we are seeking someone with knowledge of both C++ and Java programming to successfully finish the project.

Note: code contained here is only for reference, any potential co-worker would gain access to the project Git repository.

2. Sample problems

2.1. CoveringArray (1)

C++ CODE (CASA original):

```
void CoveringArray::setTrackingCoverage(bool trackingCoverage) {
  if (this->trackingCoverage) {
    this->trackingCoverage = trackingCoverage;
    return;
  }
  this->trackingCoverage = trackingCoverage;
  if (trackingCoverage) {
    unsigned strength = coverage.getStrength();
    unsigned limit = coverage.getOptions().getSize();
    Array<unsigned>firsts = coverage.getOptions().getFirstSymbols();
    Array<unsigned>counts = coverage.getOptions().getSymbolCounts();
    coverage.fill(0);
    coverageCount = 0;
    multipleCoverageCount = 0;
    if (substitutions->size()) {
      unsigned hint = 0;
      for (Array<unsigned>
             columns = combinadic.begin(strength),
             symbols(strength);
           columns[strength - 1] < limit;
           combinadic.next(columns), ++hint) {
       for (unsigned i = array.getSize();i--;) {
         for (unsigned j = strength;j--;) {
           symbols[j] = (*this)(i, columns[j]);
```

```
     }
     unsigned newCoverage =
       ++coverage.hintGet(hint, columns, firsts, counts, symbols);
     if (newCoverage == 1) {
       ++coverageCount;
     }
     if (newCoverage>1) {
       ++multipleCoverageCount;
     }
   }
  }
} else {
  // A special common case where we can bypass the () operator:
  unsigned hint = 0;
  for (Array<unsigned>
        columns = combinadic.begin(strength),
        symbols(strength);
      columns[strength - 1] < limit;
      combinadic.next(columns), ++hint) {
   for (unsigned i = array.getSize(); i--;) {
     for (unsigned j = strength; j--;) {
       symbols[j] = array[i][columns[j]];
     }
     unsigned newCoverage =
       ++coverage.hintGet(hint, columns, firsts, counts, symbols);
     if (newCoverage == 1) {
       ++coverageCount;
     }
     if (newCoverage>1) {
       ++multipleCoverageCount;
     }
   }
  }
```

JAVA CODE (CASA - TRANSLATION):

```
public void setTrackingCoverage(boolean trackingCoverage) {
    if (this.trackingCoverage) {
        this.trackingCoverage = trackingCoverage;
        return;
    }
    this.trackingCoverage = trackingCoverage;
    if (trackingCoverage) {
        Integer strength = coverage.getStrength();
        Integer limit = coverage.getOptions().getSize();

        Vector<Integer> firsts = coverage.getOptions().getFirstSymbols();
        Vector<Integer> counts = coverage.getOptions().getSymbolCounts();
        coverage.fill(0);
        coverageCount = 0;
        multipleCoverageCount = 0;

        if (substitutions.getImplementation().size() > 0) {
            Integer hint = 0;
            int [] columns = Combinadic.begin(strength);
```

Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

```
            int [] symbols = new int[strength];
            //TODO not sure if arrays are updated in for loop
            for (; columns[strength - 1] < limit; Combinadic.next(columns),
++hint) {
                for (int i = array.getSize(); i > 0; i--) {
                    for (int j = strength; j > 0; j--) {
                        symbols[j] = //TODO don't know what is being
dereferenced here
                    }
                    //TODO object casted to int
                    int tmp = (int) coverage.hintGet(hint,
arrToVect(columns), firsts, counts, arrToVect(symbols)).op_getContent();
                    int newCoverage = ++tmp;
                    if (newCoverage == 1) {
                        ++coverageCount;
                    }
                    if (newCoverage > 1) {
                        ++multipleCoverageCount;
                    }
                }
            }
        } else {
            // A special common case where we can bypass the () operator:
            Integer hint = 0;
            int[] columns = Combinadic.begin(strength);
            int[] symbols = new int[strength];
            for (; columns[strength - 1] < limit; Combinadic.next(columns),
++hint) {
                for (int i = array.getSize(); i > 0; i--) {
                    for (int j = strength; j > 0; j--) {
                        symbols[j] = //TODO don't know what is being
dereferenced here
                    }
                    //TODO object casted to int
                    int tmp = (int) coverage.hintGet(hint,
arrToVect(columns), firsts, counts, arrToVect(symbols)).op_getContent();
                    int newCoverage = ++tmp;
                    if (newCoverage == 1) {
                        ++coverageCount;
                    }
                    if (newCoverage > 1) {
                        ++multipleCoverageCount;
                    }
                }
            }
        }
    }
}
```

2.2. CoveringArray (2)

C++ CODE(CASA):

```
void CoveringArray::setTrackingNoncoverage(bool trackingNoncoverage) {
  if (this->trackingNoncoverage) {
    this->trackingNoncoverage = trackingNoncoverage;
    if (!trackingNoncoverage) {
      noncoverage->clear();
    }
    return;
  }
  this->trackingNoncoverage = trackingNoncoverage;
  if (trackingNoncoverage) {
    assert(trackingCoverage);
    assert(noncoverage->empty());
#ifndef NDEBUG
    unsigned impossible = 0;
#endif
    for (Coverage<unsigned>::iterator
         iterator = coverage.begin(),
         end = coverage.end();
       iterator != end;
       ++iterator) {
      if (!*iterator) {
      InputKnown known;
      Array<unsigned>combination = iterator.getCombination();
      for (unsigned i = combination.getSize(); i--;) {
        known.append(InputTerm(false, combination[i]));
      }
      if ((*solver)(known)) {
        noncoverage->insert(combination);
      }
#ifndef NDEBUG
      else {
        ++impossible;
      }
#endif
       }
    }
    assert(coverageCount + noncoverage->size() + impossible ==
         coverage.getSize());
  }
}
```

JAVA CODE (CASA - TRANSLATION):

```
public void setTrackingNoncoverage(boolean trackingNoncoverage) {
    if (this.trackingNoncoverage) {
        this.trackingNoncoverage = trackingNoncoverage;
        if (!trackingNoncoverage) {
            noncoverage.getImplementation().clear();
        }
        return;
    }
    this.trackingNoncoverage = trackingNoncoverage;
    if (trackingNoncoverage) {
        assert (trackingCoverage);
        assert (noncoverage.getImplementation().isEmpty());
        int impossible = 0;
        for () {
            //TODO iterator
        }
        assert ((coverageCount + noncoverage.getImplementation().size() +
impossible)
                == coverage.getSize());
    }
}
```

2.3. CoveringArray (3)

C++ CODE (CASA):

```
class CoveringArrayHeuristic : public Heuristic<CoveringArray,
CoverageCost> {
  CoverageCost estimate
    (const CoveringArray&state, const Goal<CoveringArray>&goal) const
{
    unsigned targetCoverage =
((CoverageGoal&)goal).getTargetCoverage();
    return CoverageCost
      (targetCoverage - state.getCoverageCount(),
       state.getMultipleCoverageCount());
  }
};
```

JAVA CODE (CASA - TRANSLATION):

```
//TODO ??? is this ok?
public class CoveringArrayHeuristic implements Heuristic {

    @Override
    public CoverageCost estimate(CoveringArray state, Goal<CoveringArray>
goal) { //TODO ???
        long targetCoverage = ((CoverageGoal) goal).getTargetCoverage();
//TODO ???
        return new CoverageCost(targetCoverage - state.getCoverageCount(),
state.getMultipleCoverageCount());
    }
}
```

2.4. CoveringArray (4)

C++ CODE(CASA):

```
set<CoveringArray>GraftSpace::getChildren
  (const CoveringArray&state, unsigned count) const {
  set<CoveringArray>result;
  unsigned rows = state.getRows();
  assert(options.getSize() == state.getOptions());
  assert(state.isTrackingNoncoverage());
  const set<Array<unsigned>, ArrayComparator<unsigned> >&noncoverage =
    state.getNoncoverage();
  if (noncoverage.empty()){
    return result;
  }
  unsigned attempts = 0;
  for (unsigned i = count; i; ++attempts) {
    unsigned row = rand() % rows;
    set<Array<unsigned>, ArrayComparator<unsigned> >::const_iterator
      combinationIterator = noncoverage.begin();
    unsigned combinationIndex = rand() % noncoverage.size();
    while (combinationIndex--) {
      ++combinationIterator;
    }
    Array<unsigned>combination = *combinationIterator;
    Array<unsigned>columns(strength);
    InputKnown known;
    if (attempts < MAXIMUM_SUBROW_CHANGE_FAILED_ATTEMPTS) {
      for (unsigned j = strength; j--;) {
     columns[j] = options.getOption(combination[j]);
      }
      for (unsigned j = options.getSize(), k = strength; j--;) {
```

```
     if (k && columns[k-1] == j) {
       unsigned symbol = combination[--k];
       known.append(InputTerm(false, symbol));
     } else {
       assert(!k || columns[k - 1] < j);
       unsigned symbol = state(row, j);
       known.append(InputTerm(false, symbol));
     }
      }
      if (solver(known)) {
     CoveringArray child = state;
     child(row, columns) = combination;
     result.insert(child);
     --i;
     attempts = 0;
      }
    } else {
      cout << "Considering a full row change" << endl;
      for (unsigned k = strength; k--;) {
     known.append(InputTerm(false, combination[k]));
      }
      CoveringArray child = state;
      child(row) = createRandomMatchingRow(known);
      result.insert(child);
      --i;
      attempts = 0;
    }
  }
  return result;
}
```

JAVA CODE (CASA - TRANSLATION):

```
@Override
public Set<CoveringArray> getChildren(CoveringArray state, int count) {
    Set<CoveringArray> result = new TreeSet<>();
    int rows = state.getRows();
    assert(options.getSize() == state.getOptions());
    assert(state.isTrackingNoncoverage());
    //TODO ???
    //const set<Array<unsigned>, ArrayComparator<unsigned> >&noncoverage =
    Set<Array<Integer>> noncoverage = state.getNoncoverage();
    if (noncoverage.isEmpty()) {
        return result;
    }
    int attempts = 0;
    Random rand = new Random();
    for (int i = count; i > 0; ++attempts) {
        int row = rand.nextInt() % rows;
```

Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

```
        int option = rand.nextInt() % options.getSize();
        //TODO
        int value = options.getOtherRandomSymbol(option, state(row, option));
        InputKnown known = new InputKnown();
        known.append(InputTerm(false, value));
        if (attempts < MAXIMUM_SINGLE_CHANGE_FAILED_ATTEMPTS) {
            for (int j = 0; j < option; ++j) {
                //TODO
                known.append(InputTerm(false, state(row, j)));
            }
            for (int j = option + 1; j < options.getSize(); ++j) {
                //TODO
                known.append(InputTerm(false, state(row, j)));
            }
            if (solver.solve(known)) {
                CoveringArray child = state;
                //TODO
                child(row, option) = value;
                result.add(child);
                --i;
                attempts = 0;
            }
        } else {
            System.out.println("Considering a full row change");
            CoveringArray child = state;
            //TODO
            child(row) = createRandomMatchingRow(known);
            result.add(child);
            --i;
            attempts = 0;
        }
    }
    return result;
}
```

2.5. CoveringArray (5)

C++ CODE (CASA):

```
unsigned CoveringArraySpace::computeTargetCoverage() const {
  unsigned result = 0;
  for (Array<unsigned>columns = combinadic.begin(strength);
       columns[strength - 1] < options.getSize();
       combinadic.next(columns)) {
    // Initialization:
    Array<InputTerm>combination(columns.getSize());
    for (unsigned i = columns.getSize(); i--;) {
      combination[i] = InputTerm(false,
options.getFirstSymbol(columns[i]));
    }
    // Body:
  loop:
    {
      InputKnown known(combination);
      if (solver(known)) {
     ++result;
      }
    }
    // Advance:
    for (unsigned i = columns.getSize(); i--;) {
      unsigned next = combination[i].getVariable() + 1;
      if (next <= options.getLastSymbol(columns[i])) {
     combination[i] = InputTerm(false, next);
     goto loop;
      }
      combination[i] = InputTerm(false,
options.getFirstSymbol(columns[i]));
    }
  }
  return result;
}
```

JAVA CODE (CASA - TRANSLATION):

```
public int computeTargetCoverage() {
    int result = 0;
    //TODO not sure
    int[] columns = Combinadic.begin(strength);
    for (;
         columns[strength - 1] < options.getSize();
         Combinadic.next(columns)) {
        // Initialization:
```

Authors: Yevgeniya Chekh, Jan Kohout, David Löffler, Kryštof Sýkora, Marek Szeles, Ho Minh Thanh, Miroslav Rudišin

```
        Array<InputTerm> combination = new Array<InputTerm>(columns.length);
        for (int i = columns.length; i > 0; i--) {
            combination.set(i, new InputTerm(false,
options.getFirstSymbol(columns[i]));
        }
        //TODO rewrite goto
        // Body:
        loop:
        {
            //TODO not sure
            InputKnown known = new InputKnown(combination);
            if (solver.solve(known)) {
                ++result;
            }
        }
        // Advance:
        for (int i = columns.length; i > 0; i--) {
            int next = combination.get(i).getVariable() + 1;
            if (next <= options.getLastSymbol(columns[i])) {
                combination.set(i, new InputTerm(false, next));
         goto loop; // TODO FAIL
            }
            combination.set(i, new InputTerm(false,
options.getFirstSymbol(columns[i])));
        }
    }
    return result;
}
```

3. About the project

Project CASA is part of a larger ongoing academic project, information on it can be found here: [Quality Assurance and testing methodology for IoT]

Since the main deliverable of the project is having working code at the end, we have opted for agile project management, which means less strict rotation of project roles with more iterations between shorter development phases.

The main use case is to recreate the original CASA program 1 to 1 in relation to structure. Some parts of the program, namely the “sat” and “minisat” packages might be online in Java already, which will save time.