JSR-331

Constraint Programming API

Early Draft Review

Jacob Feldman
Cork Constraint Computation Centre
University College Cork
Cork, Ireland
j.feldman@4c.ucc.ie

Narendra Jussien
Computer Science Department
Ecole des Mines de Nantes
Nantes, France
narendra.jussien@emn.fr

June 15, 2010
JSR-331 – Java Specification Request

- Java Constraint Programming API under the roof of the Java Community Process (JCP)  
  [www.jcp.org](http://www.jcp.org)
- We present key concepts and design decisions related to the proposed standard for representation and resolution of constraint satisfaction and optimization problems
- JSR-331 Early Draft is now available for public review at [www.cpstandards.org](http://www.cpstandards.org)
- Everybody comments are welcome
CP Standardization Perspective

Standard is Oriented to Application Developers while allowing CP Vendors to provide implementations.

Business World

Top-Down View

CP Interface

Bottom-Up View

CP World
Key Objectives

- Make CP more accessible for business software developers
- Allow a Java business application developer to easily switch between different solver implementations without any changes in the application code
- Assist CP vendors in creating practical and efficient JSR-331 implementations

Start with Java and surrounding languages (Scala, Groovy, Clojure, …)
JSR-331 Architecture

javax.constraints

javax.constraints.impl

javax.constraints

javax.constraints.impl
JSR-331 interfaces “javax.constraints”

- **Only 6** major CP concepts:
  - Problem
    - ConstrainedVariable
    - Constraint
  - Solver
    - SearchStrategy
    - Solution
Simple Example (Problem Definition)

```java
public static void main(String[] args) {
    //==== PROBLEM DEFINITION ==============================
    Problem p = new Problem("Test");
    //======== Define variables
    Var x = p.var("X",1,10);
    Var y = p.var("Y",1,10);
    Var z = p.var("Z",1,10);
    Var r = p.var("R",1,10);
    Var[] vars = { x, y, z, r };
    //======== Define and post constraints
    try {
        // X < Y
        x.lt(y).post();
        // X + Y = Z
        x.add(y).eq(z).post();
        // or p.linear(new Var[]{x,y}, Oper.EQ, z).post();
        p.allDifferent(vars).post();
        // 3x + 4y -7z + 2r > 0
        int[] coef1 = { 3, 4, -7, 2 };
        p.linear(coef1,vars, Oper.GT, 0).post();
        // x + y + z + r >= 15
        p.linear(vars, Oper.GE, 15).post();
        // 2x - 4y + 5z - r > x*y
        int[] coef2 = { 2, -4, 5, -1 };
        p.linear(coef2,vars, Oper.GT, x.mul(y)).post();
    } catch (Exception e) {
        p.log("Error posting constraints: " + e);
        System.exit(-1);
    }
}
```
Simple Example (Problem Resolution)

// === PROBLEM RESOLUTION ==================

p.log("=== Find One Solution:");
Solver solver = p.getSolver();
Solution solution = solver.findSolution();
if (solution != null)
    solution.log();
else
    p.log("No Solution");
solver.logStats();

=== Find One Solution:
*** Execution Profile ***
Number of Choice Points: 5
Number of Failures: 3
Occupied memory: 683448
Execution time: 31 msec
Problem Definition

- **Problem** – a factory and a placeholder for all other objects
- **ConstrainedVariable** – a base class for:
  - Var
  - VarBool
  - VarReal
  - VarSet
  - Common associated concepts:
    - name, impl, business object, #of constraints,…

- **Constraint**
Var – constrained integer variables

- Getters: getMin(), getMax(), getValue(),..
- But no setters, instead:
  - var.le(25).post();
- Creation:
  
  ```java
  Var x = problem.var("X", 1, 10);
  int[] domain = new int[] {1,2,4,7,9};
  Var var = problem.var("A", domain);
  Var[] vars = problem.varArray("A", 0, 10, 100);
  ```

- Domain type

  ```java
  public enum DomainType {
      DOMAIN_SMALL,
      DOMAIN_MIN_MAX,
      DOMAIN_SPARSE,
      DOMAIN_OTHER
  }
  ```
Constraint: common methods

- Constraint \texttt{and}(Constraint c)
- Constraint \texttt{or}(Constraint c)
- Constraint \texttt{negation}()
- Constraint \texttt{implies}(Constraint c)
- VarBool as \texttt{bool}
- void \texttt{post}()
- void \texttt{post}(ConsistencyLevel)

// green bin can contain plastic, wood, copper
\begin{verbatim}
c1 = pb.linear(type, Oper.EQ, green);
c2 = pb.linear(counts[glass], Oper.EQ, 0);
c3 = pb.linear(counts[steel], Oper.EQ, 0);
c1.implies( c2.and(c3) ).post();
\end{verbatim}
try {
    x.lt(y).post();
    p.allDifferent(vars).post();
    int[] coef1 = { 3, 4, -7, 2 };
    p.linear(coef1,vars, Oper.GT, 0).post();
}

} catch (Exception e) {
    p.log("Error posting constraint: " + e);
}
Constraints

- Currently Included:
  - All Basic
  - Linear
  - AllDifferent
  - Element
  - Cardinality
  - GlobalCardinality

- Under Consideration:
  - “regular”, “diffn”, “cumulative”, …
User-Defined Constraints

- Combinations of predefined constraints
- New sub-classes of Constraint
- Controversial Concepts (postponed):
  - Propagators
  - PropagationEvent
Problem Resolution

- **Solver** – a factory and a placeholder for search related objects

- **Search Methods**
  - Find One Solution
  - Find Optimal Solution
  - Iterate through solutions within user-defined limits

- **SearchStrategy**
  - The default strategy and implementation strategies
  - Variable and Value Selectors

- **Solution**
Search Strategies

- For now only one search strategy is required:

  ```java
  SearchStrategy strategy = solver.getSearchStrategy();
  strategy.setVarSelectorType(VarSelectorType.MIN_DOMAIN_MIN_VALUE);
  ```

- **Solver-specific strategies are listed** (but not required):
  - RestartSearchStrategy(RestartFunction)
  - BoundedBacktrackingSearchStrategy(Steps)
  - LimitedDiscrepancySearchStrategy(Disc)
  - CreditBasedSearchStrategy(InitialCredit,CreditFunction,Steps)
  - DepthBoundedSearchStrategy(Level,Steps)
  - Other

  ```java
  SearchStrategy strategy = new BoundBacktrackingSearchStrategy(100); //steps
  solver.setSearchStrategy(strategy);
  ```
Common Implementation Package
“javax.constraints.impl.search”

- Provides default implementations for:
  - solutionIterator()
  - findSolution(…)
  - findOptimalSolution(…)
  - findAllSolutions(…)
  - Most Variable and Value Selectors
  - The only abstract method is:

    abstract public Solution findSolution(ProblemState restoreOrNot);
public static void main(String[] args) {
    //======== Problem Representation ==============
    Problem problem = new Problem("Queens");
    String arg = (args.length < 1) ? "1000" : args[0];
    int size = Integer.parseInt(arg);
    problem.log("Queens " + size);
    // create 3 arrays of variables
    Var[] x = problem.varArray("x", 0, size-1, size);
    Var[] x1 = new Var[size];
    Var[] x2 = new Var[size];
    for (int i = 0; i < size; i++) {
        x1[i] = x[i].add(i);
        x2[i] = x[i].sub(i);
    }
    // post "all different" constraints
    problem.allDifferent(x).post();
    problem.allDifferent(x1).post();
    problem.allDifferent(x2).post();
}
Queens Example (resolution)

// ========== Problem Resolution ===========

// Find a solution
Solver solver = problem.getSolver();
solver.setTimeLimit(600000); // milliseconds
SearchStrategy strategy = solver.getSearchStrategy();
strategy.setVars(x);
strategy.setVarSelectorType(VarSelectorType.MIN_DOMAIN_MIN_VALUE);
strategy.setValueSelectorType(ValueSelectorType.MIN);
Solution solution = solver.findSolution();
if (solution == null)
    problem.log("no solutions found");
else {
    solution.log();
}
solver.logStats();

Queens 1000
*** Execution Profile ***
Number of Choice Points: 996
Number of Failures: 8
Execution time: 1093 msec
TCK – Technology Compatibility Kit

- org.jcp.jsr331.tests
  - JUnit tests Unit tests with asserts for expected results

- org.jcp.jsr331.samples
  - A library of well-known CSPs implemented using a pure JSR-331 API
JSR-331 Implementations

- CP Vendors
  - Support
  - Compromises

- Initial Implementations with open source CP Solvers:
  - Constrainer – available in beta
  - Choco – available in alpha
  - JaCoP – under development
  - Others are welcome
CP Community Support

- Thanks to all contributors!
  - Especially: H.Simonis, M.Z.Lagerkvist, P.Stucky, R.Szymanek, …

- To do:
  - Add Real and Set variables implementations
  - More global constraints
  - TCK Implementation
  - Integration:
    - LP/MIP tools
    - Vizualizer
  - Adding verticals (e.g. Scheduler, Router,…)
  - Discuss controversial concepts:
    - New custom constraints
    - New custom search strategies (goals)
**Appeal**

- **CP Vendors:**
  - Contact us to get an SVN access and start your own JSR-331 implementation

- **CP Practitioners:**
  - Contact us to get a working JSR-331 implementation and start using it for your own problems!
  - Testers are needed
  - Need more constructive critique – comment at [www.cpstandards.org Discussion Forum](http://www.cpstandards.org)