RAPUNZEL
Choco
Schokoaufstrich ohne Zusatz von Nüssen & Milch
vegan
BIO
sans ajout de noisettes & lait
senza aggiunta di nocciole & latte

CHOCO

Chic, un Outil Contraintes avec des Objets

Francois Laburthe, Narendra Jussien
Amadeus, Ecole des Mines de Nantes

A brief History

- 1999: a first CLAIRE implementation within the OCRE project, a national initiative for an open constraint solver for both teaching and research (Nantes, Montpellier, Toulouse, Bouygues, ONERA)
- 2003: a java first implementation (portability, ease of use for newcomers, etc.
- 2008: Choco V2
  a clear separation between the model and the solving machinery, complete re-factoring, a user-oriented version

An open constraint solver

- An open system
  - a source forge project
  - BSD license for all possible usages
- A glass box
  - designed for both teaching and research
  - readable yet efficient

A choco example

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

A choco example

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

A choco example

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

A choco example

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

A choco example

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

A choco example

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();

//5- read the model and solve it
s.read(m);
s.solve();
if (s.isFeasible()) {
    do {
        for (int i = 0; i < n; i++) {
            System.out.print(s.getVar(vars[i]).getVal());
        }
        System.out.println("");
    } while (s.nextSolution());
}
```

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();

//5- read the model and solve it
s.read(m);
s.solve();
if (s.isFeasible()) {
    do {
        for (int i = 0; i < n; i++) {
            System.out.print(s.getVar(vars[i]).getVal());
        }
        System.out.println("");
    } while (s.nextSolution());
}
```

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();

//5- read the model and solve it
s.read(m);
s.solve();
if (s.isFeasible()) {
    do {
        for (int i = 0; i < n; i++) {
            System.out.print(s.getVar(vars[i]).getVal());
        }
        System.out.println("");
    } while (s.nextSolution());
}
```

```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();
```

```
//5- read the model and solve it
s.read(m);
s.solve();
if (s.isFeasible()) {
    do {
        for (int i = 0; i < n; i++) {
            System.out.print(s.getVar(vars[i]).getVal());
        }
        System.out.println("");
    } while (s.nextSolution());
}
```

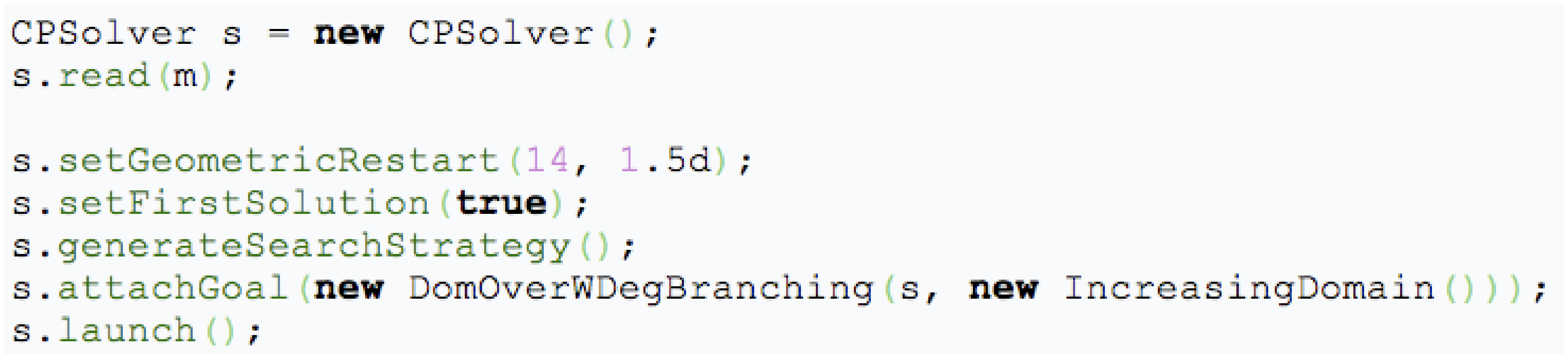
```
//1- Create the model
Model m = new CPModel();
int n = 6;
//2- declaration of variables
IntegerVariable[] vars = makeIntVarArray("v", n, 0, 5, "cp:enum");
IntegerVariable obj = makeIntVar("obj",0,100,"cp:bound");

//3- add some constraints
String regexp = "(1|2)(3*)(1|4|5)";
m.addConstraint(regular(regexp, vars));
m.addConstraint(neq(vars[0], vars[5]));
m.addConstraint(eq(scalar(new int[]{2,3,1,-2,8,10}, vars), obj));

//4- Create the solver
Solver s = new CPSolver();

//5- read the model and solve it
s.read(m);
s.solve();
if (s.isFeasible()) {
    do {
        for (int i = 0; i < n; i++) {
            System.out.print(s.getVar(vars[i]).getVal());
        }
        System.out.println("");
    } while (s.nextSolution());
}
```

Choco features

- Variables : Integer (Bound and Enum), Set, Real, Composite (plus, minus, scalar, mult, ...)
- Constraints :
  - arithmetic : equal, less than, not equal ...
  - extentional : AC3r, GAC3rm (positive, negative), AC2001....
  - global : alldifferent (bound and ac), gcc (bound and ac), cumulative, regular, occurence, disjonctive, lex, ...
  - exclusive : Tree, Geost, Cost regular
  - reified : or, and, not, implies...

Choco features

- Search :
  - solve, solveAll, iterate over the solutions
  - minimize or maximize
  - solve with restarts
- Heuristics : MinDomain, DomOverWDeg, Impact



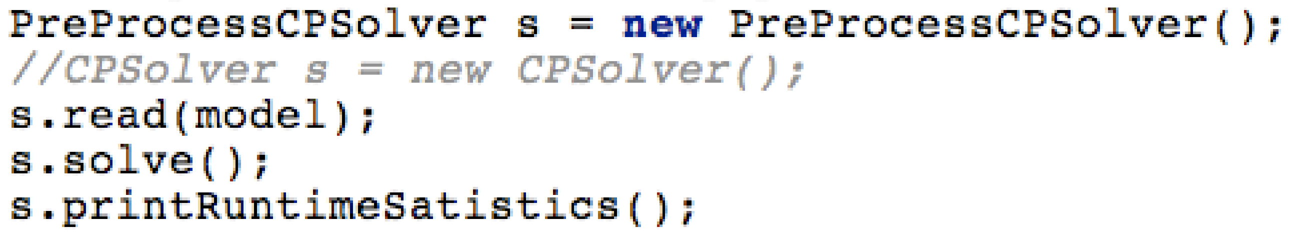
```
CPSolver s = new CPSolver();
s.read(m);

s.setGeometricRestart(14, 1.5d);
s.setFirstSolution(true);
s.generateSearchStrategy();
s.attachGoal(new DomOverWDegBranching(s, new IncreasingDomain()));
s.launch();
```

Choco choices

- Event-based propagation engine (variable oriented)
  - Fine grained events (instantiation, value removals, bound modification)
  - A constraint propagation queue to postpone heavy constraints
- Trailing (copying and recomputation available as well)

Choco as a blackbox

- Try to use the intensional constraints of choco :
  - Alldifferent, Disjunctive (on cliques), Distance constraints (|x - y| = z), linear equations, tables for equality and disequalities.
- Try to decide the level of consistency (complex predicates) :
  - Arc-consistency, decomposition by introducing intermediate variables, forward-checking



```
PreProcessCPSolver s = new PreProcessCPSolver();
//CPSolver s = new CPSolver();
s.read(model);
s.solve();
s.printRuntimeSatistics();
```

Conclusion

- Choco philosophy :
  - Open for Teaching and Research
  - Living community
  - User-friendly to discover CP
- http://choco.emn.fr/
- Choco V2 is available since the 10 Sept

Conclusion

Slow and steady wins the race

- Choco philosophy :
  - Open for Teaching and Research
  - Living community
  - User-friendly to discover CP
- http://choco.emn.fr/
- Choco V2 is available since the 10 Sept