Calc/Cream: OpenOffice spreadsheet front-end for constraint programming

Naoyuki Tamura

Information Science and Technology Center, Kobe University,
1-1 Rokkodai, Nada, Kobe 657-8501, JAPAN,
tamura@kobe-u.ac.jp

Abstract. Calc/Cream is a constraint programming system with a spreadsheet front-end implemented on OpenOffice.org Calc and Java language. Constraint problems are described by users as cell expressions on a spreadsheet, and solutions are searched by the constraint solver and shown as cell values by the system. It is also possible to use Basic macros to customize the system.

1 Introduction

Constraint programming is widely used to develop various applications including constraint satisfaction problems and optimization problems, such as production planning and scheduling, etc.

Constraint programming is originally studied as an extension of logic programming languages. However, after 1990's, several constraint programming extensions of OOL (C++ and Java) are proposed including ILOG Solver[1], ILOG JSolver[2, 3], Koalog[4], JACK[5], JCL[6], and Cream[7, 8].

However, developing constraint programming applications is still a difficult task. It is important to provide a programming environment which is easy to use for constraint application developers.

In this paper, we describe a constraint programming system with a spreadsheet front-end implemented on OpenOffice.org Calc[9] and Java language. In this Calc/Cream system[1] developed as a port of HECS system[10, 11][2], users can write their constraint problems as cell expressions on a spreadsheet.

There are several works on constraint spreadsheet systems, such as Chew and David[12], Hyvönen[13], Intellisheet[14], and Knowledgesheet[15].

Compared with these systems, Calc/Cream is more flexible because it is developed as a simple add-in of OpenOffice.org Calc which is an open-source software and the functionality of the original spreadsheet system is not modified.

[1] http://bach.istc.kobe-u.ac.jp/cream/calc.html
[2] http://kaminari.istc.kobe-u.ac.jp/hecs/

2 Cream: Class Library for Constraint Programming in Java

2.1 Features of Cream

Cream[7, 8] is a class library helping Java programmers to develop intelligent programs requiring constraint satisfaction or optimization on finite domains.

The followings are features of Cream.

- 100% Pure Java: Whole programs are written in Java.
- Open source: Cream is distributed as a free software with source code [3].
- Natural description of constraints: Various constraints can be naturally described within Java syntax.
- Easy enhancements: Programmers can easily enhance/extend constraint descriptions and satisfaction algorithms.
- Various optimization algorithms: In addition to complete search algorithms, various local search algorithms are available, such as Simulated Annealing and Taboo Search, etc.
- Cooperative local search: Several local search solvers can be executed in parallel for cooperative local search.

2.2 Example program of Cream

```
import jp.ac.kobe_u.cs.cream.*;

public class FirstStep {
  public static void main(String args[]) {
    // Create a constraint network
    Network net = new Network();
    // Declare variables
    IntVariable x = new IntVariable(net);
    IntVariable y = new IntVariable(net);
    // x >= 0
    x.ge(0);
    // y >= 0
    y.ge(0);
    // x + y == 7
    x.add(y).equals(7);
    // 2x + 4y == 20
    x.multiply(2).add(y.multiply(4)).equals(20);
    // Solve the problem
    Solver solver = new DefaultSolver(net);
    Solution solution = solver.findFirst();
    int xv = solution.getIntValue(x);
    int yv = solution.getIntValue(y);
    System.out.println("x = "+xv+", y = "+ yv);
  }
}
```

Fig. 1. Example program of Cream

[3] http://bach.istc.kobe-u.ac.jp/cream/

Fig. 1 shows an example program of Cream.

- `Network` is a class for constraint network consisting of constraint variables and constraint conditions. New constraint network is created by the constructor invocation `new Network()`.
- New constraint variable in the network `net` is created by the constructor `new IntVariable(net)`.
- New constraint $x \geq 0$ is added by executing a method `x.ge(0)`. Similarly, constraints $y \geq 0$, $x + y = 7$, $2x + 4y = 20$ are added to the network.
- After adding all variables and constraints to the network, a constraint solver (using constraint propagation and backtrack) is created by the constructor `new DefaultSolver(net)`.
- First solution is searched by `solver.findFirst()`.
- The value of the variable `x` is extracted from the solution by using a method call `solution.getIntValue(x)`.

2.3 Other features of Cream

The followings are other unique features of Cream.

- Serialized Constraint: `Serialized` constraint can be used to describe a condition in which given intervals are not overlapped each other (useful for scheduling problems).
- Optimization: A `setObjective` method can be used to specify an objective variable for optimization problems.
- Local Search Algorithms: Local search algorithms including Simulated Annealing (`SASearch`) and Taboo Search (`TabooSearch`) can be used for some optimization problems.
- Cooperative local search: A `ParallelSolver` can be used for cooperative local search in which several local search solvers can be executed in parallel.

2.4 Java classes of Cream

As shown in the previous example program, Cream consists of following classes.

- `Network` class implements constraint networks. A constraint network consists of variables, constraints, and an objective variable (optional).
- `Variable` class implements variables. A variable is a component of a constraint network. A variable is constructed with an initial domain which specifies the set of elements over which the variable ranges.
- `IntVariable` class is a subclass of the `Variable` class, and implements integer variables.
- `Domain` is an abstract class for domains.
- `IntDomain` class implements integer domains.
- `Constraint` is an abstract class for constraints. The subclass should implement `satisfy` method which makes the constraint to be arc-consistent.

– `IntComparison`, `IntArith`, and `IntFunc` classes implements constrains on integer variables.
– `Serialized` class implements a constraint where given integer intervals are not overlapped each other.
– `Solver` is an abstract class for constraint solvers.
– `DefaultSolver` is a branch-and-bound solver.
– `LocalSearch` is a random-walk solver for problems including `Serialized` constraints.
– `SASearch` is a subclass of `LocalSearch`, and implements Simulated Annealing search.
– `TabooSearch` is a subclass of `LocalSearch`, and implements Taboo search.
– `ParallelSolver` is a subclass of `Solver`, and executes several solvers in parallel.

For more details of Cream implementation, please refer to to the Cream web page[7].

3 Calc/Cream: Constraint Spreadsheet

3.1 Features of Calc/Cream

Calc/Cream is a constraint programming system with a spreadsheet front-end implemented on OpenOffice.org Calc and Java language.

In Calc/Cream, constraint problems are described by users as cell expressions on a spreadsheet, therefore users do not need to study the use of programming languages.

Calc/Cream is developed as a simple add-in of OpenOffice.org Calc and the functionality of the original spreadsheet system is not modified.

3.2 Example of Calc/Cream

Fig. 2 shows an example of Calc/Cream solving an old Japanese elementary school problem:

> There are some cranes and tortoises. They are 7 in total, and their legs are 20 in total. How many cranes and tortoises are there?

The number of cranes is assigned to the cell `B2`, the number of tortoises is assigned to the cell `B3`, and a solution satisfying `B2+B3=7` and `2*B2+4*B3=20` will be searched by the solver.

The left hand side of the Fig. 2 shows cell expressions described by a user. By pressing a START button (not shown in the figure), a solution satisfying cell expressions specified by `CONSTRAINTS` is searched by changing cell values specified by `CVARIABLES`.

In this example, a solution satisfying the following conditions is searched.

– The cell value of `C2` is equal to `B2*2`.

	A	B	C
1		Number	Legs
2	Crane	0	=B2*2
3	Tortoise	0	=B3*4
4	Total	7	20
5	Condition	=B2+B3=B4	=C2+C3=C4
6		=CVARIABLES(B2:B3;0;100)	=CONSTRAINTS(B2:C5)

	A	B	C
1		Number	Legs
2	Crane	4	8
3	Tortoise	3	12
4	Total	7	20
5	Condition	TRUE	TRUE
6		...	...

Fig. 2. Example of Calc/Cream (Cranes and Tortoises)

- The cell value of `C3` is equal to `B3*4`.
- The cell value of `B2+B3=B4` is true.
- The cell value of `C2+C3=C4` is true.
- The cell values of `B2` and `B3` are within the interval $[0, 100]$.

The right hand side of the Fig. 2 shows the search result.

Fig. 3 shows an example finding a solution of a semi magic square problem which assigns different numbers of 1–9 to cells `B2`–`D4` so that the summations of each rows and columns are equal to 15.

	A	B	C	D
1	=CNOTEQUALS(B2:D4)	=SUM(B2:B4)=15	=SUM(C2:C4)=15	=SUM(D2:D4)=15
2	=SUM(B2:D2)=15	0	0	0
3	=SUM(B3:D3)=15	0	0	0
4	=SUM(B4:D4)=15	0	0	0
5	=CVARIABLES(B2:D4;1;9)			
6	=CONSTRAINTS(A1:D4)			

Fig. 3. Example of Calc/Cream (Semi Magic Square)

`CVARIABLES(B2:D4;1;9)` means each cell value of `B2`–`D4` is an integer from 1 to 9, and `CNOTEQUALS(B2:D4)` means these have different values each other. Expressions such as `SUM(B2:D2)=15` means the summation of each row or column is equal to 15.

3.3 How Calc/Cream works

The following shows how Calc/Cream works after pressing the "Start" button.

- The spreadsheet is scanned, and cells containing variables and constraints are parsed to tree structures, and they are added to constraint network of Cream.
- Cream solver is invoked to start solving.
- Values of the obtained solution is written back to the spreadsheet.

The following StarBasic macro performs the above procedure.

```
Const JAVA_SERVICE = "com.sun.star.loader.Java2"
Const CREAM_SERVICE = "jp.ac.kobe_u.cs.cream.uno.CreamAddins"

Dim object_cream As Object

Sub CStartButton
    # Create UNO service to use Cream
    CreateUnoService(JAVA_SERVICE)
    object_cream = CreateUnoService(CREAM_SERVICE)

    # Scan spreadsheet and add constraints
    Dim sheetIndex(0) As Object
    sheetIndex() = Array(0, 1, 2)
    object_cream.CScanDocument(ThisComponent, sheetIndex())

    # Start Cream solver
    If Not object_cream.CStart() Then
        MsgBox "No Solutions !!"
    EndIf
End Sub
```

The following macro implements the Next button.

```
Sub CNextButton
    CreateUnoService(JAVA_SERVICE)
    object_cream = CreateUnoService(CREAM_SERVICE)

    If Not object_cream.CNext() Then
        MsgBox "No More Solutions !!"
    End If
End Sub
```

4 Summary

This paper presents Cream class library for constraint programming in Java, and Calc/Cream constraint spreadsheet system which is developed as an add-in of OpenOffice.org Calc.

Both systems are still under development, and further enhancements are planned in near future.

Acknowledgments

We would like to give special thanks to Takashi Shinozaki, Hideaki Okamoto, and Mutsunori Banbara for helping to develop Calc/Cream system.

Calc/Cream was developed as a part of HECS (HEterogeneous Constraint Solver) system[10, 11] which was supported in part by METI and IPA (The

Information-technology Promotion Agency) under grant of 2003 Exploratory Software Project. Some improvement ideas of Calc/Cream are obtained through the discussion with Mr. Kino, a project manager of IPA Exploratory Software Project.

References

1. Puget, J.F.: A C++ Implementation of CLP. (ILOG) http://www.ilog.com/.
2. ILOG: (ILOG JSolver) http://www.ilog.com/.
3. Chun, A.H.W.: Constraint programming in Java with JSolver. In: Proceedings of the First International Conference on the Practical Application of Constraint Technologies and Logic Programming (PACLP99). (1999)
4. Koalog: (An Overview of Koalog Constraint Solver) http://www.koalog.com/.
5. Abddennadher, S., Krämer, E., Saft, M., Schumauss, M.: JACK: A Java constraint kit. In: Proceedings of the International Workshop on Functional and (Constraint) Logic Programming (WFLP 2001). (2001)
6. Artificial Intelligence Laboratory of EPFL Switzerland: (JCL: Java Constraint Library) http://liawww.epfl.ch/JCL/.
7. Tamura, N.: (Cream Programmers Guide) http://bach.istc.kobe-u.ac.jp/cream/.
8. Ohnishi, S., Tasaka, H., Tamura, N.: Efficient representation of discrete sets for constraint programming. In: Proceedings of the International Conference on Constraint Programming (CP-2003). (2003) 920–924
9. OpenOffice.org: (OpenOffice.org) http://www.openoffice.org/.
10. Banbara, M., Tamura, N., Inoue, K., Kawamura, T.: Java implementation of a heterogeneous constraint solving system. (final report of IPA Exploratory Software Project 2002) (2003)
11. Banbara, M., Tamura, N., Inoue, K., Kawamura, T., Hamaki, H.: Java implementation of a distributed constraint solving system. (final report of IPA Exploratory Software Project 2003) (2004)
12. Chew, T., David, J.M.: A constraint-based spreadsheet for cooperative production planning. In: Proceedings of the AAAI Sigman Workshop in Knowledge-Based Production Planning, Scheduling and Control. (1992)
13. Hyvönen, E., Pascale, S.D.: A new basis for spreadsheet computing: Interval solver for Microsoft Excel. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI 1999). (1999) 799–806
14. Adachi, Y.: Intellisheet: A spreadsheet system expanded by including constraint solvers. In: Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments (HCC'01). (2001) 173–179
15. Gupta, G., Akhter, S.F.: Knowledgesheet: A graphical spreadsheet interface for interactively developing a class of constraint programs. (In: Proceedings of the Second International Workshop on Practical Aspects of Declarative Languages (PADL 2000)) 308–323