The Knowledge Engineering Review, Vol. 10:4, 1995, 405–410

Workshop Report
Constraint languages/systems and their use in problem modelling

ROLAND H. C. YAP

Department of Information Systems and Computer Science, National University of Singapore, Singapore 119260
(email: ryap@iscs.hus.sg)

1 Introduction

The workshop on *Constraint Languages/Systems and their Use in Problem Modelling* was one of the post-conference workshops organised in conjunction with the 1994 International Logic Programming Symposium and was held at Ithaca, New York, USA, on November 18, 1994. This report is a summary of the papers which were presented at the workshop.

Recently much work has focused on combining constraints and logic programming languages into the Constraint Logic Programming (CLP) framework. This has proven to be quite successful, and CLP systems have been used in the solving and modelling of complex and difficult problems. The main motivation of this workshop was to provide a forum to emphasize practical issues with the use, application, implementation and development of constraint languages and CLP. The majority of the papers presented at the workshop were related to CLP, and as such, this report has been written primarily from a CLP perspective.

2 Background

Constraint Logic Programming (CLP) is a new approach to high level programming languages which combines constraints with logic programming. In a CLP language, the constraints serve to directly model relationships in a particular domain; and the logic programming aspect provides a programming component and a control mechanism. Typically, the kinds of constraint domains encountered in CLP are well defined mathematical objects such as real numbers, integers, booleans, strings, sets, etc. Constraints over these domains are usually quite powerful and expressive, so much so that solving these constraints in their fullest generality tends to be computationally intractable.

CLP by combining the two declarative frameworks of constraints and logic programming leads to high level programming languages which are not only powerful and expressive but also have a simple semantics. Because constraints are integrated within a programming language, CLP provide a natural modelling language as well as a programming language for solving complex problems, particularly where there are many complex relationships to be solved, modelled, or reasoned about. For example, consider industrial production-type scheduling problems. Since CLP languages offer a flexible constraint language together with a programming language component, the kinds of *ad hoc* conditions/constraints which can arise in scheduling problems may be more readily expressed in a finite domain CLP system than with traditional modelling techniques.

There are now a number of readily available CLP languages and systems. For example, there are both research as well as commercial systems such as: CHIP, CLP(fd), ECLiPSe, CLP($\mathcal{R}$), PROLOG III, CLP(BNR), SICStus Prolog 3.0, etc. The main kinds of constraints offered by these systems deal with finite domain constraints on bounded integers, arithmetic constraints (primarily

linear) over the real numbers of rationals, relational interval constraints, booleans constraints and list concatenation. There are also many other kinds of constraint domains and concurrent CLP languages as well as other constraint programming paradigms which haven't been mentioned.

3 Workshop details

The papers accepted for presentation at the workshop program fell into two main categories: those on applying CLP in building and modelling applications; and those on the design and implementation of CLP languages and systems. The workshop was organised by Pierre Lim (European Computer-Industry Research Centre), Jean Jourdan (LCR Thomson-CSF) and Roland H. C. Yap (University of Melbourne), and papers were selected with the help of an additional program committee consisting of Joxan Jaffar, Kim Marriott, Michael Maher, Micha Meier, Spiro Michaylov, Helmut Simonis, Peter Stuckey and Mark Wallace.

The workshop was very well attended with approximately 50 participants. Proceedings of the workshop are available as European Computer-Industry Research Centre technical report ECRC-94-38 for the papers on applications and modelling and as Department of Computer Science technical report 94/19, University of Melbourne, for the papers on design and implementation of CLP. A total of 18 papers presented from the list of papers below were presented:

- D. Chemla, D. Diaz, P. Kerlirzin and S. Manchon, "Using clp(FD) to Support Air Traffic Flow Management".
- H. Simonis and T. Cornelissens, "Modelling Producer/Consumer Constraints".
- P. Boizumault, C. Gueret and N. Jussien, "Efficient Labeling and Constraint Relaxation for Solving Time Tabling Problems".
- M. Fromherz and B. Carlson, "Optimal Incremental and Anytime Scheduling".
- Y. Fattah and C. Holzbaur, "Constraint Logic Programming for Modelling and Simulation of Dynamic Physical Systems".
- R. Skuppin, "Studies in CLP Applicability: The ROSAT Example".
- W. Older, "Applications of Relational Interval Arithmetic to Ordinary Differential Equations".
- T. Hickey, "CLP(F) and Constrained ODEs".
- A. Mezhoud and J-C Dufourd, "Components Placement in VLSI Datapath based on Constraint Programming".
- A. Perini and F. Ricci, "Constraint Reasoning and Interactive Planning: An Application to Forest Fire Fighting".
- F. Ambert, B. Legeard and E. Legros, "Constraint Logic Programming on Sets and Multi-Sets".
- P. Baptiste, B. Legeard and H. Zidoum, "Sequences Constraint Solver based on P-Q-R Trees".
- B. Li and D. Elliman, "A New CSP Algorithm for CLP (Interval)".
- C. K. Chiu and J. H. M. Lee, "Interval Linear Constraint Solving using the Preconditioned Interval Gauss-Seidel Method".
- J. Burg, P. J. Stuckey, J. C. H. Tai and R. H. C. Yap, "Approaches to Linear Equation Solving in CLP".
- D. Arnow, K. McAloon and C. Tretkoff, "Parallel Integer Goal Programming".
- S. Prestwich and S. Mudambi, "Cost-Parallel Branch and Bound in Constraint Logic Programming".
- F. Stolzenburg and P. Baumgartner, "Constraint Model Elimination and a PTTP-Implementation".

4 Applications and modelling

The expressiveness of CLP is well suited for describing and modelling many complex problems because the high level constructs allow a direct and straightforward formulation of the "problem

constraints" as well as providing a more direct level of reasoning about the problem and its solution. Since reasoning is at the level of constraints, the flexibility of the CLP approach means that the CLP solution can solve not only existing problems, but also new categories of problems. For example, the papers in the section on differential equations show that not only can initial value problems be solved, but also final value problems, constrained path problems or sensitivity analysis can be done using the same method.

The papers which were presented on application and modelling can be divided into three main categories: scheduling and resource allocation; physical modelling and differential equation solving; and finally other applications of constraints.

4.1 Scheduling and resource allocation

One major problem area which CLP has been used to solve real-life industrial problems is in combinatorial search problems such as planning, scheduling and resource allocation. Operations research techniques such as linear programming have been traditionally used for such problems. CLP systems with finite domains offer a different approach to the modelling and solving of such problems primarily using a combination of consistency techniques, heuristics and search where the underlying language is a more expressive one than that offered by linear programming.

The paper by Chemla et al. applies clp(FD), a CLP language with finite domain constraints, to solve an air traffic flow management problem to schedule departure slots for aircraft. An `atmost interval` constraint which is implemented using the primitive constraints of clp(FD) allows easy modelling of the capacity requirements on air traffic. This is then extended to incorporate flow rate constraints on the amount of aircraft traffic.

The paper on producer/consumer constraints by Simonis and Cornelissens deals with the kinds of scheduling problems that arise with consumable resources such as raw materials and expresses the requirement that more resources must be available or produced than consumed. The CLP language CHIP provides global constraint solving techniques in addition to local consistency techniques for the finite domain constraints. A framework is then described for problems with product/consumer constraints which uses the global cumulative constraints in CHIP. This is then extended to model other requirements such as stock margins, variable resource consumption, splitting of resource use and limited storage.

Timetabling is another classic example of difficult combinatorial problems. The paper by Boizumault et al. starts by giving a survey of approaches to timetabling and then describes modelling a university timetabling problem in CHIP. They show the importance of choosing intelligent labelling strategies for controlling the combinatorial explosion and obtain good computation results using first fit enumeration. An extension for relaxing the problem constraints is then discussed.

The paper by Fromherz and Carlson describes scheduling of reactive applications which have to cope with continually changing environments. The problem here is to optimize schedules where there are incremental constraints being added because of the reactive problem environment. This is applied to the scheduling of reprographic machines where the scheduler's job is to determine how to produce a desired output sequence of printed sheets when the document pages are being received incrementally. They propose a technique for reactive scheduling which combines optimisation with minimal commitment which they call projected optimisation.

4.2 Physical modelling and differential equations

Another area of modelling work which is well suited for CLP is in physical modelling since the physical relationships can be described by constraints. The problem of solving differential equations is closely related to problems in physical modelling, since much of the behaviour of a model is described in terms of differential equations. Describing models in a relational and

declarative fashion leads naturally to a relational ordinary differential equations (ODE) solving using constraints rather than traditional numerical ODE solving.

The paper by El Fattah and Holzbaur presents an approach to modelling and simulating dynamic systems using bond graphs and CLP. The conventional approach to model generation would proceed by using the bond graphs to compute causal directions. Their approach to integrating the modelling uses constraints to specify the relationships on the state, power and output variables in the bond graphs model directly. The dynamic system relationships can then be determined by projection. Simulation of the bond graph model can be done using relational ODE techniques. A variety of examples of dynamical physical models are given to demonstrate their approach.

The paper by Skuppin presents some experimental results with using a CLP approach to solve a problem to do with the estimation of gyroscope errors in the ROSAT satellite. The experiments in this paper demonstrate that it was feasible to apply CLP based ODE solvers to this problem. The advantages were that the solution was both declarative and simple as well as being flexible. For example, the same program could be used for estimating gyroscope parameters as well as for performing the simulation. The disadvantage was that the CLP systems used were slower than the classical imperative approaches. However one has to take into account that the CLP systems used, Prolog III and CLP($\mathcal{R}$), are semi-interpreted while the C and Fortran programs were native code.

The paper by Older uses relational interval arithmetic in CLP(BNR) to solve ODEs. There are three main advantages gained over traditional numerical approaches. The interval ODE methods here guarantee correctness of the solutions obtained for two reasons: firstly, interval techniques allows the remainder terms in the Taylor expansion to be retained, and this can compute an inclusion of the true trajectory; and secondly, the kinds of numerical errors from roundoff and truncation which complicate conventional methods are replaced by an interval of uncertainty in the solution. The second advantage is that relational ODEs are symmetrical, thus arbitrary boundary conditions can be solved for. Thirdly, the same flexibility aids in stability analysis of the solution as well as the problem parameters.

The paper on CLP(F) by Hickey builds upon relational ODE solving to define a constraint language, CLP(F), which is intended for sound and declarative scientific constraint programming. The domain of CLP(F) consists of booleans, integers, real numbers, real-valued functions of one variable and vectors. Functions can be constrained by describing their values and derivatives as a finite number of points as well as through other constraints. The constraint solver is based on the underlying relational interval arithmetic constraints in CLP(BNR) and a power series method for handling the ODEs.

4.3 Other applications of constraints

The two papers described in this section do not utilise a CLP approach; however, they do apply constraint techniques for problem solving. The paper by Perini and Ricci describe an interactive planner which deals with the problem of fighting forest fires. The planner integrates case-based reasoning with constraint reasoning on temporal relations. The temporal constraints provide plan fitting and adaptation as well as resource scheduling. The temporal constraint solver here is built in an object oriented framework and combines CSP consistency techniques for real domains with shortest path algorithms for producing the minimal temporal network. Constraint reasoning here is used in the context of a bigger decision support system aimed at supporting the user in the whole process of forest fire management.

The paper by Mezhoud and Dufourd present a method for solving min-cut graph linearisation problems which arises from the problem of component placement in VLSI design. This method is based on a constraint methodology, and it first decomposes the original non-directed graph into elementary paths which give ordering constraints on the nodes. The second step merges these locally consistent constraint systems into a single coherent one. The constraint system is then solved to minimise an objective function using a variation on depth-first branch and bound search.

5 Design and implementation

While constraints are very expressive and powerful, this is not without cost. Many of the constraint problems which we wish to solve are intrinsically hard problems, and this makes devising efficient execution strategies difficult. In addition, the typical CLP execution strategy requires that constraints be solved and processed incrementally and backtracking needs to be supported. These requirements together with the program execution model which is quite general, makes efficient implementation of CLP languages a challenging problem.

The design of a CLP language and constraint system which is suitable for the desired class of problems is also very important. There are many possibilities for the choice of constraints for modelling particular classes of problems, and the choices may also vary considerably in terms of expressive power and computational efficiency. Thus the goal of obtaining practical CLP languages and systems lies both in the appropriate design of the CLP language and the constraint and programming constructs therein, as well as the development of the underlying constraint technology and algorithms.

The papers which were presented can be divided into four categories: constraints on sets; arithmetic constraint solving; parallelism and concurrency; and a paper on theorem proving with constraints.

5.1 Constraints on sets

The use of sets is quite pervasive in both mathematics and computer science. As such there have been a number of different proposals for CLP languages with constraints over sets. The two presentations in this section both deal with sets in the CLPS language.

The paper by Ambert et al. describes the CLPS language, which incorporates constraints on sets and multi-sets which can be nested hierarchically. Apart from the set constraints, it offers restricted universal quantifiers and set intensional definitions on goals. Constraint solving of the sets consists of a combination of partial consistency techniques associated with the set constraints and their operations. The set constraint solver is also integrated with a finite domain solver.

The other paper by Baptiste et al. describes extending the sets in CLPS to deal with constraints on sequences. Sequence constraints are used to express that individual elements of a sequence are consecutive, define partial orders on sequences, concatenation and other kinds of metric constraints as well as set relations such as equality, membership, subset, etc. Sequences are represented using a novel normal form based on a tree-like representation called P-Q-R trees. This is combined with finite domain techniques for dealing with the metric constraints.

5.2 Arithmetic constraint solving

Dealing with arithmetic constraints is very fundamental for many problems and a number of CLP languages provide arithmetic constraints over real numbers or rationals. In the main, the CLP constraint solvers have dealt primarily with linear arithmetic constraints as non-linear constraints are more intractable. Another approach to dealing with arithmetic constraints is to consider arithmetic constraints over intervals instead of single points, and this has lead to successful application of interval arithmetic methods for dealing with non-linear arithmetic constraints.

The paper by Burg et al. studies algorithms for linear equation solving in CLP with emphasis on the requirements for CLP such as incrementality, detection of fixed variables and backtracking. A number of related algorithms are described according to how they make use of forward elimination and backward substitution, and the degree to which this is done which also corresponds to the amount of eagerness or laziness in the equation solving technique. These various algorithms are then compared in the context of their sustainability for a CLP system, and their performance is empirically evaluated on a suite of typical programs. The experimental results show that the most

appropriate choice of solving depends on the amount of determinism in the program and whether or not it is necessary to detect all fixed variables.

The paper by Chiu and Lee describes an efficient linear equation solver for CLP languages based on interval arithmetic. The interval arithmetic constraint solvers based on interval narrowing which have been typically used in other CLP interval systems work well for non-linear constraints but result in wide intervals when it comes to linear equations and are hence inaccurate. This paper describes an efficient and accurate method for solving linear equations for CLP (Interval) which uses an interval version of the Gauss–Seidel method together with a preconditioner to ensure convergence. This incremental version of the algorithm has $O(n^3)$ complexity where n is the number of variables. Experimental results demonstrate that good accuracy and efficiency compared with interval narrowing and a previous Gaussian-elimination based solver.

The paper by Li and Elliman describe an algorithm for solving non-linear interval constraints. This algorithm combines both interval reasoning and numerical methods. The idea is to narrow the intervals by projecting the function onto a minimum and maximum function of one variable for each dimension which can be solved using numerical techniques. This gives tighter intervals than interval narrowing techniques alone. The same kind of pruning interval pruning process can then also be used instead of simple interval bisection to do intelligent interval splitting.

5.3 Parallelism and concurrency

Parallelism and concurrency is also an important area of CLP. CLP like logic programming is inherently parallelisable both at a fine grain level as well as a more coarse grain distributed level. Parallelism is important because it gives opportunities for obtaining execution speedup. In addition, concurrency in CLP languages is also an important and useful programming paradigm.

The paper by Arnow et al. presents a CLP-like language called 2LP, which is extended to support distributed disjunctive programming and applies it toward parallel integer goal programming. 2LP is a constraint programming language designed for linear and integer programming. Distributed integer goal programming is accomplished with only a few small modifications to the 2LP integer goal program and model.

The paper by Prestwich and Mudambi describes an approach to parallelising branch and bound in the context of CLP called cost-parallel branch-and-bound. One common approach to parallelising these problems is to apply OR-parallelism. One drawback of OR-parallelism with such problems is that the amount of parallelism may be limited. The cost-parallel strategy searches for solutions at different costs in parallel and uses the results to update the bound and to further control the parallel search. They show for some job-shop scheduling problems that cost-parallelism is considerably faster than OR-parallelism.

5.4 Theorem proving with constraints

In CLP, the logic programming component is based on Horn clauses. The paper by Baumgartner and Stolzenburg investigate adding constraints to general non-Horn theorem proving calculi. They introduce a sound and complete calculus for theorem proving which combines model elimination with constraint solving. This has the advantage that both CLP technology and Prolog technology can be reused by applying the Prolog Technology Theorem Prover (PTTP) technique of Stickel. They show an example of terminological reasoning based on the PROTEIN theorem prover, and combine it with the terminological constraint solver implemented by the constraint handling rules (CHR) in ECLiPSe.