New structures of symbolic constraint objects: sets and graphs
Extended abstract*

Carmen Gervet

European Computer-Industry Research Centre
Arabellastr. 17, D-8000 Munich 81, Germany
email: carmen@ecrc.de

March 17, 1993

1 Abstract

A lot of work has been done up to now in designing Constraint Logic Programming Languages in order to solve combinatorial problems. Built-in computational domains in CLP support simple expression of problems and their efficient solution. Building a new computational domain comprising sets and graphs, this paper presents new symbolic constraints on set and graph structures in a CLP environment. Its main aim is to offer to the programmer the possibility to describe and solve in a natural, concise, declarative, expressive and efficient manner real Operations Research problems which are based on set and graph theory.

2 A constraint object is more suitable than a variable

Usually the addition of new variables denoting sets to logic programs extends the unification algorithms to the involvement of these formulas [4]. As any added value to a language, it proves to be detrimental to the initial language performances. Moreover set unification is NP-complete [5].

Our constraint handler for sets and graphs provides efficient constraint propagation under the control of the system. So, defining sets and graphs as constraint

*WCLP’93, Marseille
object contributes, without consequential loss of efficiency, to enhance the language expressiveness. Our approach is then much closer to ALICE [1] and CLPS [3] than to SETL [7].

Our approach is also economical because the semantics of sets and graphs is built into our representation. This reduces the number of variables to represent particular constraints. Let us take an example coming from J.L. Laurièrè which illustrates our point. Usually a structure constraint such as the bijection (BIJ in ALICE) gives birth to \( \sum_{j \in J} x_{ij} = 1 \), conditions on boolean variables \( x_{ij} \). A single image for each element belonging to \( I \) means that only one \( x_{ij} \) is set to 1. This representation enforcement is very costly as soon as the number of variables increases [2]. Our representation, on the contrary, uses only two variables in each constraint: \( \text{card}(	ext{successor}(j))=1 \).

3 The semantic aspect

Let us now describe what are the theoretical foundations of set and graph constraints. The principal idea is the one of invariant handling. A set or a graph object variable definition, is bounded below by elements belonging to set or graph and above by the potential members. These bounds are given as part of the set and graph definition. Each time a constraint or an operator is applied to a set or a graph, it might increase the lower bounds as well as reduce the upper ones. Once the fixed point is reached the variable can be instantiated. Considering that both bounds are modified during the propagation of constraints, the fixed point is reached faster. If a set was represented by atomic values and a partially known set by domains, the domain size and the number of possible propagation steps would be very large. Our representation of a partially known set is structured so as to support economic representation and reduce the number of unnecessary propagation steps.

4 Set and relational operators

Our set terms can represent simple finite sets of which the cardinality might be known or bounded. The usual set operators (\( \cap \), \( \cup \), \( \text{card} \) and set constraints (\( =, \neq, \in, \subseteq \)) can be applied to the set terms above. The \( \text{card} \) operator contributes largely to maintain the invariant.

We define a binary relation as a bi-dimensional set and thus the constraints can directly represent the ones of relational algebra (function, bijection, application, surjection, successor, predecessor, ...). In fact a graph is now seen as a particular binary relation, an endomorphic one. In addition to the relational constraints, some particular constraints have been defined as symmetrical, connectivity, completeness, transitivity, path ... in parallel with operators hamiltonian cycle, transitive closure,... These descriptive and dynamic definitions should free the programmer from having to choose a representation to suit the applied constraints [6].
5 Conclusion

The definition of set and relational constraints has been conducted by the following arguments: the use of graphs is not rare in the Operations Research environment. Furthermore, the programmer should not encounter difficulties in defining his problem. Thus, a natural user-level syntax must be added to the constraint objects through the concise definition of graphs and sets. Defining set, relational and graph constraints, we added symbolic constraints to a CLP scheme. These structured objects enable more efficient propagation to be performed in terms of number of variables involved as well as propagation steps.

We are currently investigating the applicability of set and graph constraints to versions of the travelling salesman problem. Some work is still to be done though, both to expand the application domains of these constraints (for example valued and unoriented graphs) and integrate this work in the new ECRC Eclipse system. Some theoretic studies are still being done in terms of algorithms complexity.

References


