Abstract. PrologIA has been designing and marketing AI languages and tools since 1984. The company occupies a active position in research into CLP and Natural Language processing. PrologIA has created expert systems for diagnosis aid, file rationalization etc ... and develops turn-key real-life applications. This paper presents two applications based on CLP technology : a decision support system in the field of middle term banking planning grounded on the Constraint Logic Programming technology and a crew scheduling application mixing Constraint Logic Programming and Set Partitioning Modeling.

1 SD-Solver : CLP-based simulation tool

The CEFI research center and La Hénin bank are building a Decision Support System (DSS) in the field of middle term banking planning. The first requirement for this banking DSS was to offer both forward and backward (i.e “multidirectional”) simulations on the basis of a single dynamic model. Since the problem appeared to be quite general, it has been decided to develop a generic CLP-based simulation tool on the top of Prolog III. This tool, named SD-Solver [1], has been used to implement a banking application.

The decision process of the bank relies on a quarterly model. This model describes the dynamic behavior of the bank. Exploring the impact of alternative decisions (“what if” analysis) helps the decision maker to understand the dynamic behavior of the system he’s supposed to control. However, his final objective is rather to make a decision which leads this system to a given position (“what for” analysis). Standard optimization packages are devoted to this task, they usually prove not flexible.

Within this context, the main objective has been to design a banking DSS using CLP technology on which a flexible multidirectional simulation system can be grounded in terms of constraints.

The system gives to the user a direct control on the solving process. A first reason for that comes from to fact that the definition of a well-posed problem will only be achieved after a “trial and error” process. At any moment the system
should be able to add constraints, to inspect the reached state and to remove constraints before performing a new trial.

A second reason is related to the backward simulation and the control on the search space. The user should be allowed to freely "navigate" on the search space, even to prune it "on the fly" in order to converge to an interesting solution. The third reason advocate for a real conversational mode. The main objective is to provide the decision maker with a tool at promoting a deep understanding of the system he manages.

Of course, SD-Solver inherits its main properties from Prolog III, but offers in many cases an easier way to implement applications grounded on array oriented numerical models. A first version of the banking DSS has been implemented using this environment. A shortened variant of the quarterly model has been written in a text file, and a specific graphical human interface has been implemented on top of SD-Solver.

The first results are demonstrative and the new compiler Prolog IV is expected to bring a better efficiency.

2 Crew Scheduling

The objective of this application [4] is to provide to a French airline company a flexible tool in order to help the experts to minimize crew costs while satisfying the many constraints imposed by governmental and labour laws. Flexibility is required to adapt the schedule to the evolution of these laws and to the changes of commercial strategies. The structure of the company's network, the existence of multiple bases and services of middle sized cities make the crew scheduling problem difficult.

In order to solve this difficult problem, we have joined the advantages of CLP and set partitioning modeling. Since set partitioning involves linear integer programming, our approach uses linear relaxation followed by a specialized enumeration procedure.

The method used in this application consists in

1. selecting a significant sub-problem of the initial problem.
2. using then optimal integer solution of the sub-problem as an upper bound to filter the initial solution
3. solving the initial system (The system obtained by filtering is small enough to be solved exactly)

Each of three steps described above are performed using the same Simplex algorithm. On top of three steps, pre-processing is used to reduce the size of the problem.

In order to tackle large size problem, some parts of this tool - based on Prolog - is written in C, either for reasons of performance or for reasons of memory. It is the case for the kernel of the software : a specialized version of the simplex algorithm designed for set partitioning matrices.