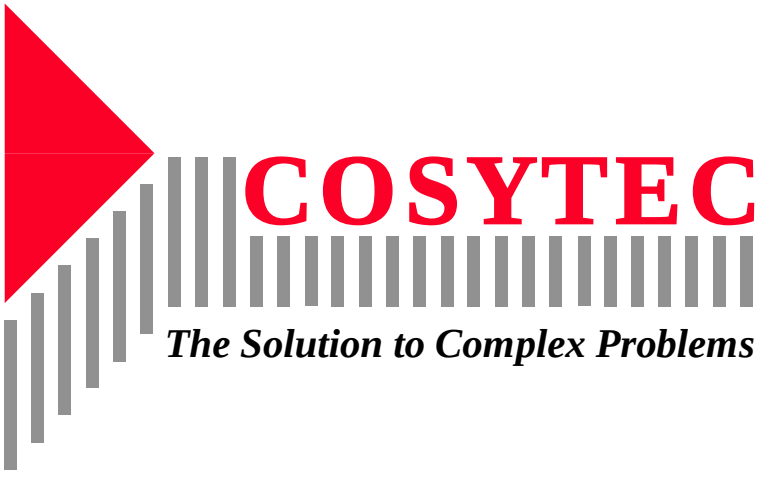
Application Development with the CHIP System



Limited Warranty:

This document is provided for information purposes only and subject to change without any prior notice. COSYTEC does not provide any warranties covering the information provided and specifically disclaims any liability in connection with this document.

Trademarks:

UNIX is a trademark of AT&T Bell Laboratories
MS-DOS is a trademark of Microsoft Corporation
Intel, Pentium, 486 are registered trademarks of Intel Corp.
Xwindows is a trademark of MIT
The CHIP C Library is the property of COSYTEC.

Authors:

Nicolas Beldiceanu
Helmut Simonis
Philip Kay
Peter Chan

Reference Number:

Reference: COSY/WHITE/002
Version: 1.2
Revision : A
Date : April 1997

COPYRIGHT:

COSYTEC SA
Parc Club Orsay Université
4, Rue Jean Rostand
F-91893 Orsay Cedex, France
Tel. +33 1 60 19 37 38
Fax. +33 1 60 19 36 20
Email. help@cosytec.fr

1. Introduction

In the last ten years, constraint logic programming has been increasingly studied as a tool to develop decision support systems for complex problems. Started as a research project at ECRC in Munich in 1986, the constraint logic programming system CHIP has evolved into an industrial tool for problem solving.

The CHIP system combines several constraint solving engines with other modules in a problem solving environment. A significant number of applications have been developed with CHIP both at COSYTEC and elsewhere.

2. Constraint Problems

In this section we briefly introduce constraint satisfaction problems and discuss the motivation for the use of constraint logic programming (CLP). We explain which type of problem is best suited for the CLP approach and where these problems occur in practice. They share a set of characteristics, which make them very hard to tackle with conventional problem solving methods. A more general introduction to constraint logic programming can be found in [JM94] [FHK92].

Combinatorial problems occur in many different application domains. We encounter them in Operations Research (for example scheduling and assignment), in hardware design (verification and testing, placement and layout), financial decision making (option trading or portfolio management) or even biology (DNA sequencing). In all these problems we have to choose among many possible alternatives to find solutions respecting a large number of constraints.

We may be asked to find an admissible, feasible solution to a problem or to find a good or even optimal solution according to some evaluation criteria. From a computational point of view, we know that most of these problems are difficult. They belong to the class of NP-hard problems. This means that no efficient and general algorithms are known to solve them.

At the same time, the problems themselves are rapidly changing. For example, in a factory, new machines with new characteristics may be added, which might completely change the production scheduling problem. New products with new requirements will be added. If a scheduling system can not be adapted to these changes, it will rapidly become useless.

Another aspect is that the complexity of the problems is steadily increasing. This may be due to increased size or complexity of the problem, or may be caused by higher standards on quality or cost effectiveness. Many problems which have been handled manually for a long time now exceed the capacity of a human problem solver.

At the same time, humans are typically very good at solving these complex decision making problems. They can have a good understanding of the underlying problem and often know effective heuristics and strategies to solve them. In many situation they also know which constraints can be *relaxed* or ignored when a complete solution is not possible. For this reason, it is necessary to build systems which co-operate with the user via friendly graphical interfaces and which can incorporate different rules and strategies as part of the problem solving mechanism.

Developing such applications with conventional tools has a number of drawbacks.

- The *development time* will be quite long. This will increase the cost, making bespoke application development very expensive.
- The programs are *hard to maintain*. Due to their size and complexity, a change in one place may well lead to a number of other changes which are required in other places.
- The programs are *difficult to adapt* and to extend. As new requirements arise during the life cycle of the program, changes become more and more difficult.

The use of constraint logic programming should lead to a number of improvements:

- The *development time is decreased*, as the constraint solving methods are reused.
- A declarative problem statement makes it *easier to change* and modify programs.
- As constraints can be added incrementally to a program, *new functionality can be added* without changing the overall architecture of the system.
- Strategies and *heuristics can be easily added* or modified. This allows to include problem specific information inside the problem solver.

We will now review some of the background techniques which are used inside CLP. Figure 1 shows the basic influences on constraint logic programming. In the next paragraphs, we will discuss these different influences.

2.1.1 Logic Programming

Logic programming is based on the idea that (a subset of) first order logic can be used for computing. The first logic programming language PROLOG was created by A. Colmerauer in 1972. Other important influences come from the work of J. A. Robinson on unification and of R. Kowalski on SL-resolution. Originally intended for natural language processing, PROLOG is now used as a general purpose language.

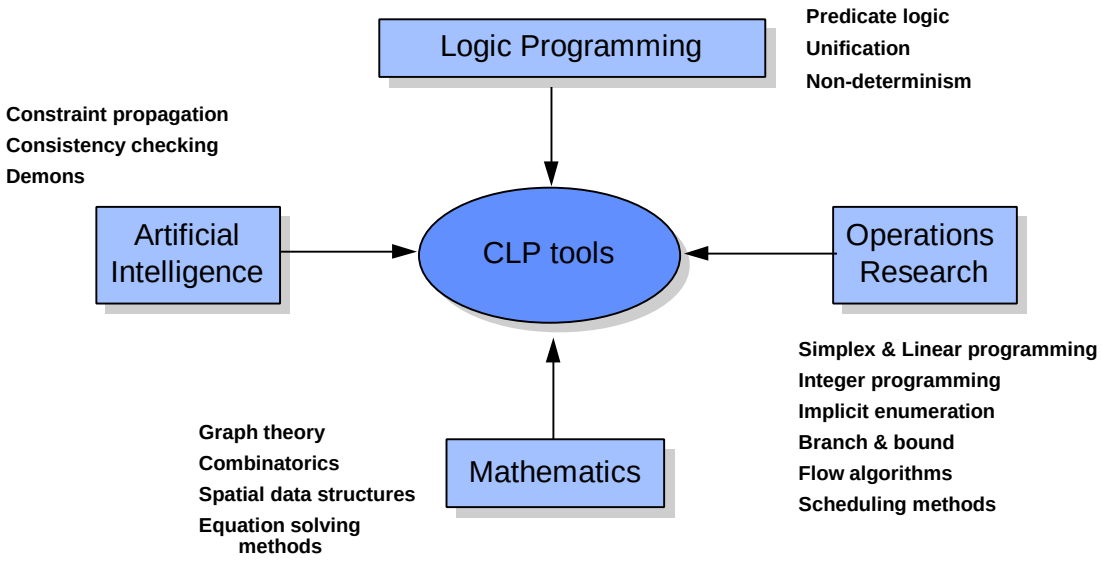


Figure 1: Techniques behind constraints

2.1.2 Constraint Handling

Constraint handling techniques are a well known method in the artificial intelligence area. Early traces can be found in work on computer graphics by Sutherland and Borning. They are also used by Sussman and Steele for circuit analysis or by Davis for circuit diagnosis. Another early use of constraints is in computer vision by Waltz. More general constraint handling systems are for example REF-ARF and ALICE. The basic idea is the declarative expression of problems as constraints and specialised constraint solvers over some restricted domains (numbers, finite sets) to efficiently find solutions to the constraints.

Consistency techniques form a important subclass of constraint handling methods. They have been introduced in order to express problems over finite domains. Problems are modelled with two components, variables, which can range over finite sets of possible values and constraints, which are relations between the variables. Different techniques, for example arc- and path- consistency, are used to remove inconsistent values from the variable domains until solutions are found.

2.1.3 Operations Research

A large number of techniques from Operations Research (OR) are used in CLP systems. The constraint solvers for linear constraints over rational numbers are based on linear programming techniques and the Simplex algorithm. The new global constraints [AB93] [BC94] over finite domains as well use techniques from scheduling and placement to check consistency.

2.1.4 Mathematics

CLP also uses a number of techniques from particular mathematical areas. Some of the problem solvers for Boolean problems for example use classical results from Boolean algebra [NM89]. Other results from finite mathematics and graph theory are used for global constraints [BC94].

3. The CHIP System

While constraints are the centrepiece of the CHIP application environment, a number of other components play an important role in developing applications. We now give an overview of this environment.

3.1 System Components

The CHIP system consists of a number of different components which together greatly simplify the development of applications. The tool extends the functionality of a *host language* with constraints and other sub systems. Two host languages are supported at the moment.

- The Prolog based version of CHIP uses intrinsic language features like *unification* and *backtracking search* to achieve a deep integration of the constraints with the host language. Constraints and search procedures can be easily defined and extended in the base language.
- The C/C++ versions of the CHIP system takes the form of a constraint library which can be used inside application programs. Since search and variable binding are not part of the host language, the integration is not as seamless as the Prolog version.

Other modules of the CHIP environment include XGIP, a graphical library based on Xwindows, QUIC, an SQL interface to relational databases and foreign language interfaces CLIC and EMC. The different modules can be used together with an object modelling system based on the object layer CHIP++ to build large scale end-user systems [KS95].

3.2 Behind the Scene

The CHIP system consists of around 100000 lines of C code. Table 1 gives an overview of the size of the different modules. The Prolog kernel accounts for only 10%, the different interfaces and the graphical system use 20%. The constraint kernel uses another 10% of the code. The remaining 60% of the system are used in the code of the global constraints, which combine multiple (up to 20) algorithms and constraint methods for each primitive.

Module	Size (lines of code)
CHIP kernel	80000
Prolog	8000
Core Constraints	12000
Global Constraints	60000
Interfaces (CLIC,EMC,QUIC)	3000
XGIP graphics	20000

Table 1: CHIP system lines of code

This distribution shows the importance of both the graphical interface XGIP and the in particular the global constraint handling methods inside CHIP.

3.3 History

CHIP [DVS88] was one of the first CLP systems together with PROLOG III [Col90] and CLP(R) [JL87]. It was developed at the European Computer-Industry Research Centre (ECRC) in Munich during the period 1985-1990 [DVS88] [DSV87] [DSV90]. Different constraint solvers were developed and integrated in a Prolog environment.

At the same time the technology was tested on a number of example applications. The CHIP system is now being marketed and developed by COSYTEC. The early versions of CHIP were also the basis for a number of derivatives by ECRC shareholder companies.

BULL developed CHARME, a constraint based system with a C like syntax, ICL produced DecisionPower based on the ECRC CHIP/SEPIA system and Siemens is using constraints in SNI-Prolog. ECRC has also continued research on constraints.

COSYTEC's current CHIP version V5 differs from the earlier constraint system by the inclusion of powerful global constraints [AB93] [BC94]. These high-level constraints (for example *cumulative*, *diffn* or *cycle*) work on sets of variables and express complex conditions which must hold among these variables.

They can be used to model resource constraints, placement problems or tour planning constraints, among others. The propagation in global constraints is based on semantic methods, which offer strong a priori reduction of the search space based on mathematical

and OR results, for example using spatial data structures, graph theory and network flow algorithms.

Global constraints are not special purpose constructs, they are used as building blocks or high-level constraint primitives in CHIP. For this reason they express very general concepts and can be linked together in multiple ways.

4. Application Studies

From the beginning, the development of CHIP was linked to application requirements. A large number of application studies showed the practical interest of constraint technology. Many of these studies were benchmark comparisons with either conventional, special purpose programs or Operations Research (OR) tools. Three initial areas of interest were covered, combinatorial problems from OR [DSV87] [DVS88] [VC88] [DSV90] [VH89] [DSV92] [DSV88] [DS91], circuit design applications [SND88] [SD93] [GVP89] and financial decision support [Ber90].

An overview of early application studies can be found in [DVS88a] [VH89]. Other application studies with CHIP (outside COSYTEC) are for example [BDP94] [BLP95].

Results on benchmark problems can also be found in [VSD92] [AB93] [BC94].

5. Industrial Applications

In this section we present some large scale, industrial systems developed with CHIP. These examples show that constraint logic programming based on Prolog can be used efficiently to develop end-user applications. The constraint solver is a crucial part of these applications. Other key aspects are graphical user interfaces and interfaces to data bases or other programs. Some background on how such applications can be developed with logic programming is given in [KS95]. We have grouped these applications in different categories according to the types of constraints which are included.

5.1 Network Problems

Applications in this group deal with generalised versions of the warehouse assignment problem already studied as a constraint problem in [VC88]. Limited capacity of warehouses or multiple product lines add new types of constraints to this otherwise well-understood problem from OR.

5.1.1 LOCARIM (France Telecom, Telesystemes, COSYTEC)

The **LOCARIM** application lays out the computer/phone network in large buildings. Telephone connections in different rooms must be linked together via concentrators, which have limited capacity. Cables must be run according to the building layout, which is scanned in from architectural plans. The system is used by France Telecom to propose a cabling and to estimate cabling costs for new buildings. The application was developed by the software house Telesystemes together with COSYTEC.

5.1.2 Distribution Planning (EBI)

The Turkish company EBI has build a distribution planning system which determines the placement of different products in selected warehouses. Customers for multiple products

are served from these warehouses. The objective is to minimise storage cost while achieving customer satisfaction with small delivery delays.

5.1.3 PlaNets (Enher, UPC/CSIC)

This application controls the reconfiguration of an electrical power network in order to perform repairs and maintenance on some segments of the network. The system re-routes energy in order to minimise end-user problems respecting multiple electrical and other constraints. The program was developed at the university of Catalonia in Barcelona for the power company Enher [CGR95].

5.1.4 Distributed Banking Network (ICON)

The Italian software house ICON has developed a system for facilities and load distribution in the Italian inter-banking network [CF95]. The program distributes applications over a network of mainframes in order to provide services to customers while balancing and minimising network traffic.

5.2 Personnel Assignment

Another important application group is personnel assignment problems. Applications in this domain are characterised by the variety of rules and restrictions which must be taken into account. These constraints arise from government regulations, union contracts or technical limits, and can become very complex.

5.2.1 Crew (Servair/GSI)

The **CREW** system developed for the catering company SERVAIR by the software house GSI performs personnel planning and assignment for the French TGV bar/restaurant personnel. It creates a four week planning assigning agents of the correct qualification to different catering jobs in the trains. The assignment respects travel times, rest periods and other hard constraints and tries to balance the overall workload for all agents.

5.2.2 TAP-AI (SAS Data/COSYTEC)

TAP-AI [BKC94] is a planning system for crew assignment for SAS. It schedules and reassigns pilots and cabin crews to flights and aircraft respecting physical constraints, government regulations and union agreements. The program is intended for day-to-day management of operations with rapidly changing constraints.

5.2.3 DAYSY (Lufthansa, SEMA, COSYTEC, U. Patras)

The ESPRIT **DAYSY** project aims at developing a personnel re-assignment system for daily operations of an airline. It reacts to delays, cancellation or addition of flights or changes in the availability of personnel by changing the existing crew assignment to satisfy the new constraints. The project is undertaken by Lufthansa, Sema Group, COSYTEC and the university of Patras, Greece.

5.2.4 OPTISERVICE (RFO/GIST/COSYTEC)

The **OPTISERVICE** system generates a personnel assignment for each of the TV stations of RFO in the French overseas departments. It allocates qualified personnel to each of

the tasks in the station, from reporting events to providing 24h service for technicians. The constraint system is combined with a rule-based module to identify working-time limits based on the different types of working contracts.

5.3 Production Planning/Scheduling

Production planning and detailed scheduling are major application areas of the CHIP constraint system. Typical constraints are temporal sequences and limits, resource capacity limits and resource allocation problems.

5.3.1 Plane (Dassault Aviation)

Dassault Aviation has developed several applications with CHIP. **PLANE** [BCP92] is a medium-long term scheduling system for aircraft assembly line scheduling. The objective of the program is to decide on changes of the speed of production on the assembly line, called the cadencing of production. By changing the speed often, production can follow varying demand more closely, decreasing inventory costs. On the other hand, each change in speed incurs a large cost for assigning/reassigning personnel and material.

5.3.2 Made (Dassault Aviation)

Another system developed by Dassault, **MADE** [CDF94] is a detailed scheduling system for a complex work-cell in the factory. The program controls a cutting press and related machines in a work-shop. As part of the scheduling, the system has to find a 2D placement of different products on sheets of metal, corresponding to different orders which are processed at the same time on the press.

5.3.3 Coca (Dassault Aviation)

The **COCA** system of Dassault is used for production step planning. The program decides which operations on one module of the aircraft can be performed together, depending on the rotational orientation of the aircraft and the place on the module where the operations are performed. Precedence constraints and safety rules must also be taken into account.

5.3.4 Production Schedule (Amylum/Beyers)

This program schedules production lines for a chemical factory in Belgium. Main constraints are the set-up restrictions, which exclude certain product sequences, and machine choices, which change production rates. The objective is to find a compromise between achieving due-dates for orders and using long production runs to minimise operating cost.

5.3.5 SAVEPLAN (SLIGOS)

The French software house Sligos has redeveloped an existing, FORTRAN based scheduling and inventory control system with the help of the CHIP constraint solver inside an object based development environment.

5.3.6 Concerto (Sema Group, COSYTEC)

COSYTEC has developed a project management tool for the **Concerto** CASE tool of Sema Group. It handles precedence constraints and resource requirements. The

integration of the tool provides information as soon as the project status changes and the planning must be reconsidered.

5.4 Transport

The transport area contains many different types of interesting constraint problems. A major concept is tour planning, finding for example the optimal sequence of deliveries for a set of lorries which transport goods from factories to customers. The CHIP cycle constraint was especially developed to model this type of problem.

5.4.1 EVA (EDF/GIST)

EVA is used by EDF to plan and schedule the transport of nuclear waste between reactors and the reprocessing plant in La Hague. This problem is highly constrained by the limited availability of transport containers and the required level of availability for the reactors. The program was co-developed by EDF and the software house GIST.

5.4.2 Transport planning (EBI)

As a second stage of the project described in section 6.1.2, EBI has developed a tour planning system for multiple warehouses/customers with capacity constraints on the lorry fleet. Each lorry can transport a limited amount of goods from some warehouses to customers. The sequence of deliveries is controlled by the urgency of the order and the objective to minimise travel costs.

5.4.3 TACT (COSYTEC)

The **TACT** system was developed for the transport department of a food manufacturing company in the UK. It covers the spectrum from production planning over detailed scheduling to resource assignment. In the production planning part, the optimal number of trips to satisfy the transport requirements for a day is generated. The scheduling part orders these trips in time, satisfying different constraints from producer/consumer limits of various factories to resource limits and sequencing conditions. The assignment part then assign drivers and lorries to the trips, while also keeping track of resources at the departure and arrival sides of the transport.

5.5 Others

There are quite a number of CHIP applications for other application areas, ranging from database query optimisation to military command and control systems. One of the first large scale CHIP systems in daily use was the SEVE application described in the next paragraph.

5.5.1 SEVE (CDC)

SEVE is a portfolio management system for CDC, a major bank in Paris, developed in-house. It uses non-linear constraints over rationals and finite domain constraints to choose among different investment options. The system can be used for “what-if” and “goal-seeking” scenarios, based on different assumptions on the development of the economy.

6. Case Studies

In this section we present two case studies in a bit more detail. One of the programs is a production scheduling system in a chemical factory in Belgium, the other a scheduling and simulation tool for oil refineries.

6.1 ATLAS

The ATLAS system is a scheduling application for one part of the MONSANTO plant in Antwerp. The program schedules the production of different types of herbicides. The production process is shown in figure 2. Production basically consists of two steps, a batch formulation part and a continuous packaging (canning) operation. In the formulation part, batches of several ten thousand litres of chemical products are produced. Each formulation batch takes several hours. Several processors with different characteristics are available. After the formulation, the product is kept in buffer tanks with a very limited capacity. Exceptionally, it can be stored in containers for a limited time period. From the buffer tanks, products are fed to the different canning lines where the products are filled in bottles and cans of different sizes. The canning lines differ in speed, manpower requirements and the type of products that they can handle. The bottles and cans are packed in cartons and palettes and stored in a holding area before delivery. Both the storage area for packaging material and the holding area are limited in size, which imposes constraints on the production process.

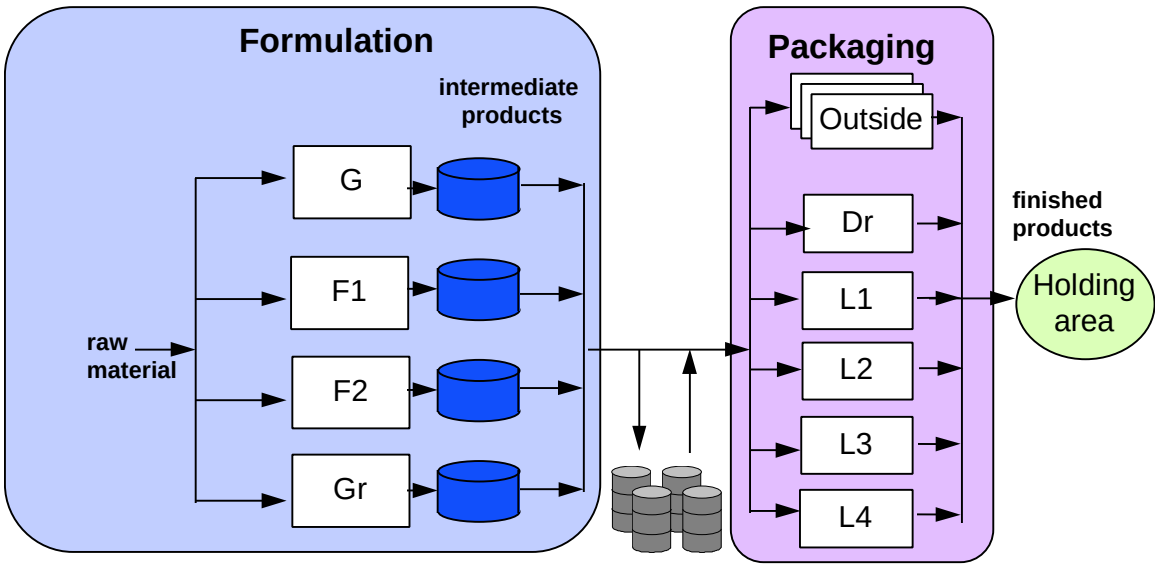


Figure 2: ATLAS production process

Besides many constraints associated with machine choice, set-up times and cumulative manpower constraints, producer/consumer constraints [SC95] play a major role in the problem solver. Intermediate products must be produced in the right quantity before they can be used on the packaging lines, and packaging material must be available in the warehouse before a task can be started.

The system works with data sets of several dozen intermediate products and more than one thousand packaging materials. A typical schedule may contain several hundreds of

batches and canning tasks split into many sub-tasks to handle the stock level constraints with an eight hour resolution. In addition, several thousand other constraints are needed to express other parts of the scheduling problem.

There is a production planning problem for the batch formulation part, where the system determines the correct number of batches for intermediate products and their optimal sizes. Since the storage tanks for intermediate products are limited in number and size, the correct amount of each product must be formulated to avoid left-overs which would block the storage tanks.

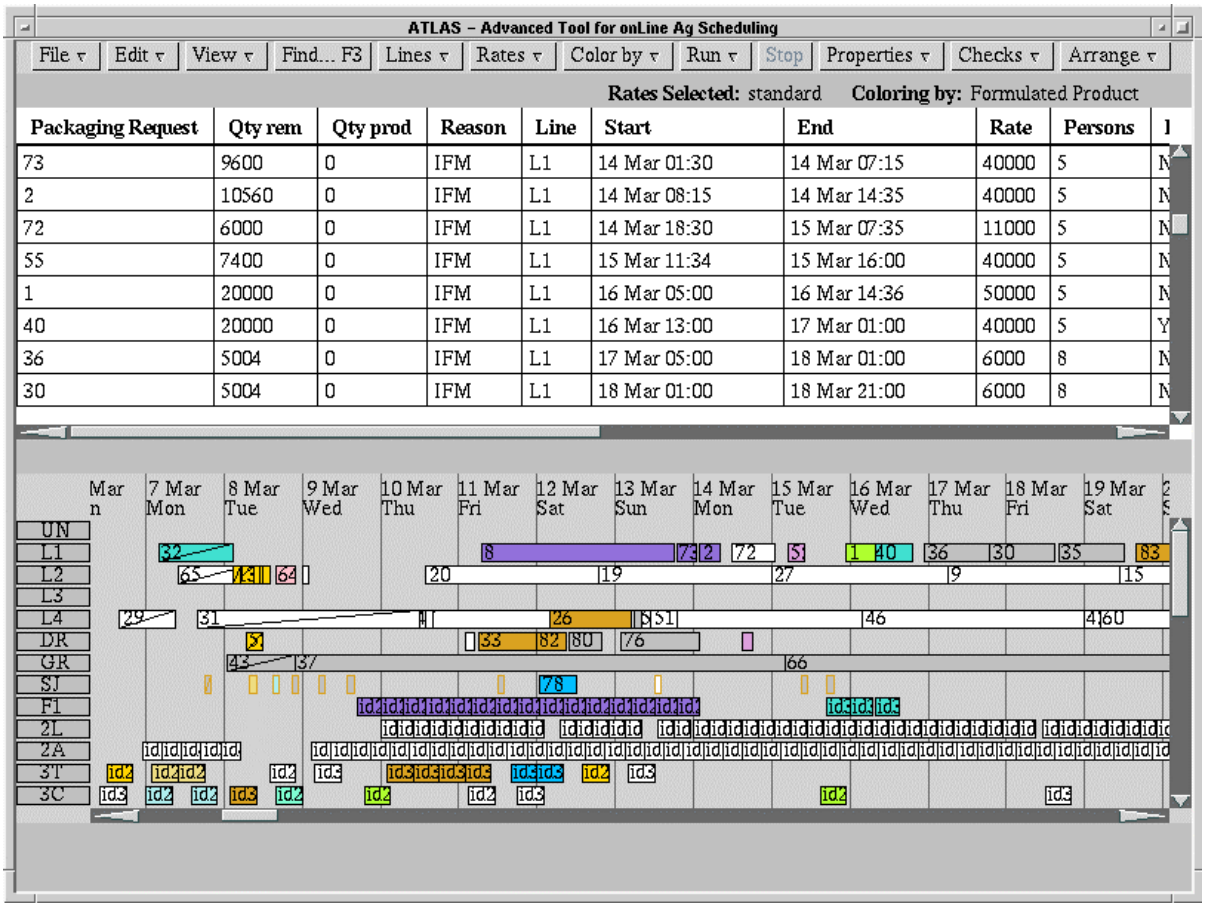


Figure 3: The ATLAS application

In figure 3, we show an example screen dump of the ATLAS application. The schedule is shown both in the form of a Gantt chart and as a textual list. The user can interact with the program via direct manipulation of items on the screen. A number of different views is provided to manipulate schedules, resource availability or stock levels.

The system is used by several users with different responsibilities. The scheduler for the factory works together with people responsible for scheduling the formulation and the inside and outside packaging lines. Other persons control the inventory and are responsible for ordering packaging materials.

These users continuously exchange information via the system. An overview of the dynamic scheduling environment is given in figure 4. Orders and sales forecasts are

added to the data base from external sources. Information about stocks of packaging material and deliveries are also available.

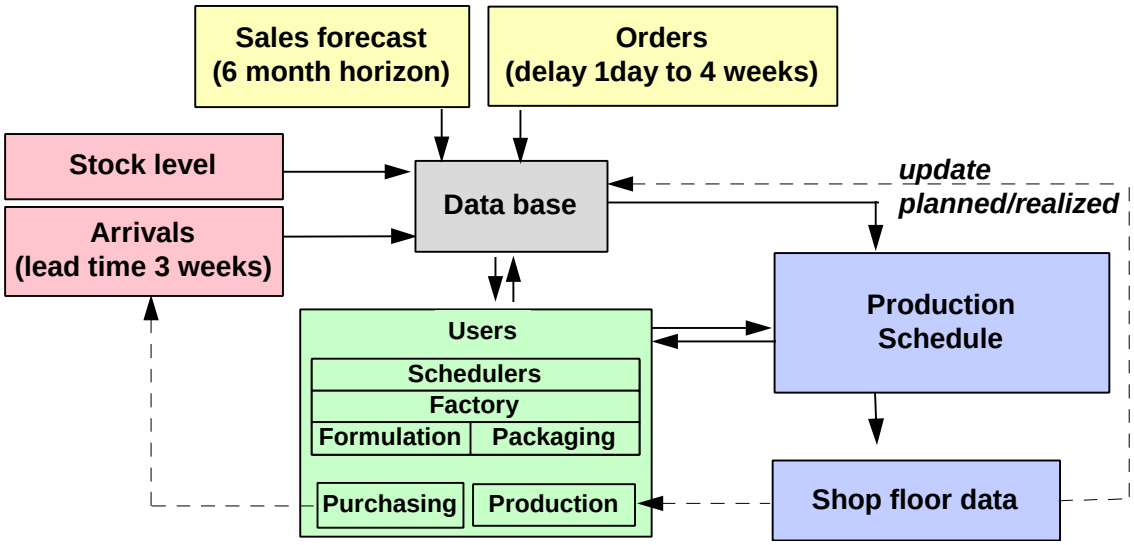


Figure 4: The scheduling process

Shop floor data is entered to update the current situation on the manufacturing side. The current production schedule is updated on demand and is used to determine requirements of raw materials and packaging materials. The program can be used on a "what if" basis in order to see the impact of accepting new orders or changing delivery due dates.

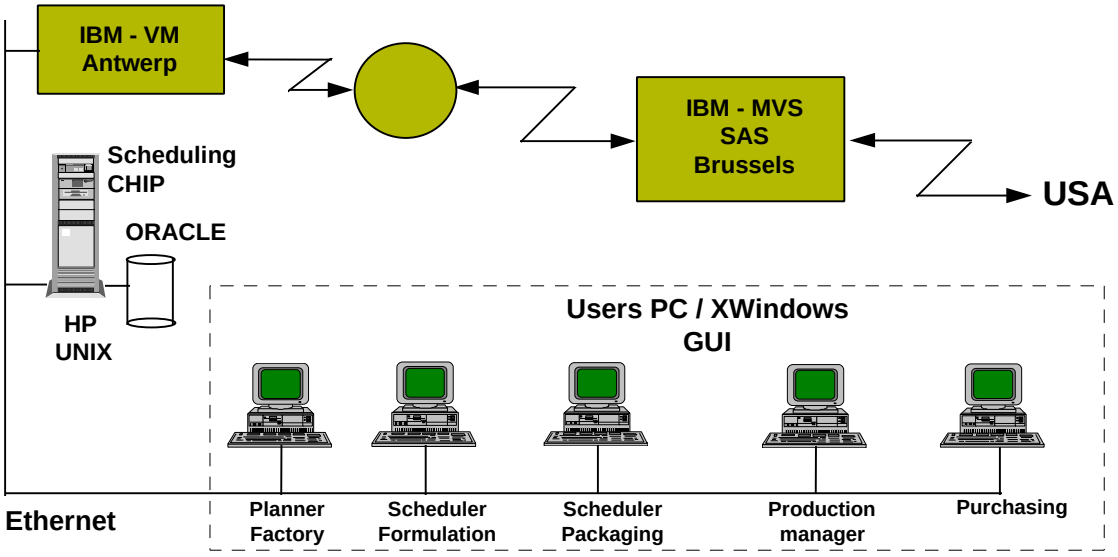


Figure 5: ATLAS system architecture

The system architecture of the ATLAS system is shown in figure 5. The application runs on a UNIX workstation together with an Oracle data base. The different users connect to this application via their personal computers running an Xwindows emulation. The workstation is also connected to different mainframe computers which keep information about orders, stock levels and deliveries. The master schedule is kept inside the data

base, but each user can create and store local scenarios in file form. These scenarios can be "what-if" type evaluations, or alternative production plans, which can be exchanged between users.

The ATLAS system is operational since July 93, the complete project ended in spring 94. As an end-user system, the constraints are not visible to the user. The interaction is via the Gantt chart and stock level diagrams. The user can move tasks manually and override the automated scheduler or change its strategy.

The system shows the type of complex scheduling problem which is well suited to the constraint approach. It is a real-life, not a pure problem, and does not fit easily into existing OR problem classifications. On the other hand, solutions which satisfy most constraints are hard to find. There are many forms of interaction between constraints and conflicts between different constraint types which can not be resolved automatically. The system is used as a decision support system, which helps the human scheduler, but does not replace him.

In important aspect during development was the possibility to incrementally express the constraints of the program. Rather than starting with a complete set of conditions, constraints were added by groups to understand the effect and requirements of each constraint type.

We discuss some implementation aspects below in section 9. A significant development effort was required to model and solve the problem and to generate the integration and graphical interface parts.

6.2 FORWARD

The second case study is the FORWARD system, a scheduling and simulation tool for oil refineries. It is used to plan refinery operations for production changes on a rather detailed level for next three days and at an overview level for the next three weeks. The system is currently in use at three sites in Europe. Each refinery has its own particular features and operation procedures. This means that the program must be extensively customised for each site. The first installation is operational since September 94. FORWARD is a joint development by the engineering company TECHNIP and COSYTEC.

It combines a non-linear simulation tool written in FORTRAN with scheduling and optimisation part, which were developed in CHIP together with the graphical user interface.

To explain the function of FORWARD, we present an (idealised) overview diagram of a refinery in figure 6. Crude oils arrive by tanker or pipeline and are stored in large tanks in the tankfarm. Different types of crude may be mixed to feed the process part of the refinery, starting with the crude distillation unit (CDU). A large number of other units are connected to different branches of the CDU output. They are used to maximise the percentage of light (valuable) products, which are obtained from the crude oil. The mixture of crude oils determines the throughput of the processing units and thus the economic efficiency of the refinery.

At the end of the process part semi-finished products are stored and then blended together to obtain commercial products like gasoline of defined qualities. Up to 12 different components can be used to generate gasoline for example. The ratio of the different products determines the cost of the finished product and therefore the margin of

the refinery against the market price. At the output side of the refinery, the off-lift of the finished products must be scheduled to satisfy customer demand and optimise resource use (pumps, berths, etc).

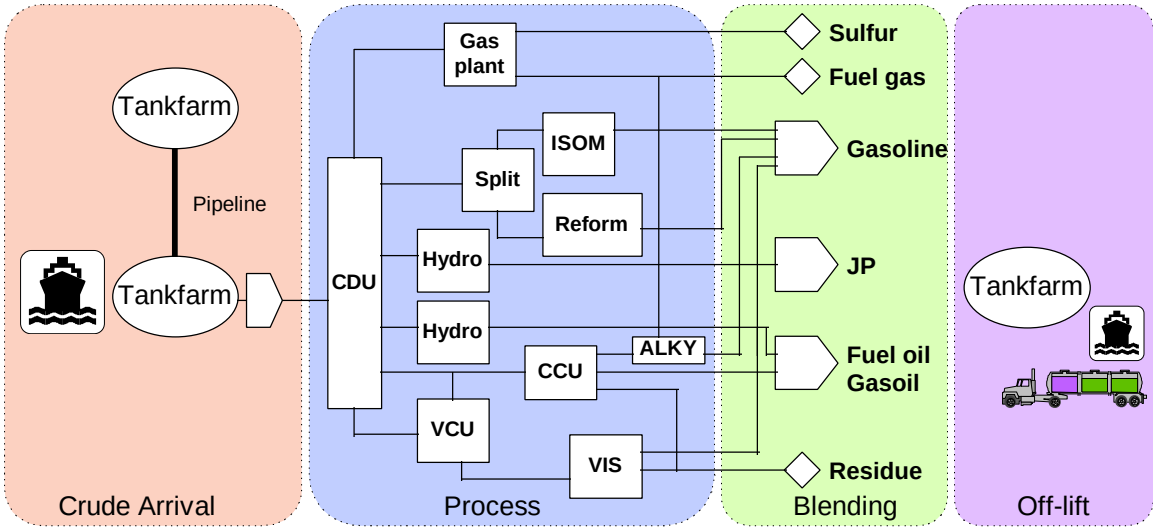


Figure 6: Refinery schematic diagram

A refinery will have perhaps 30 different types of processing units, 250 tanks of different sizes in different parts of the refinery, connected by thousands of pipes and pumps which allow many, but not all product movements.

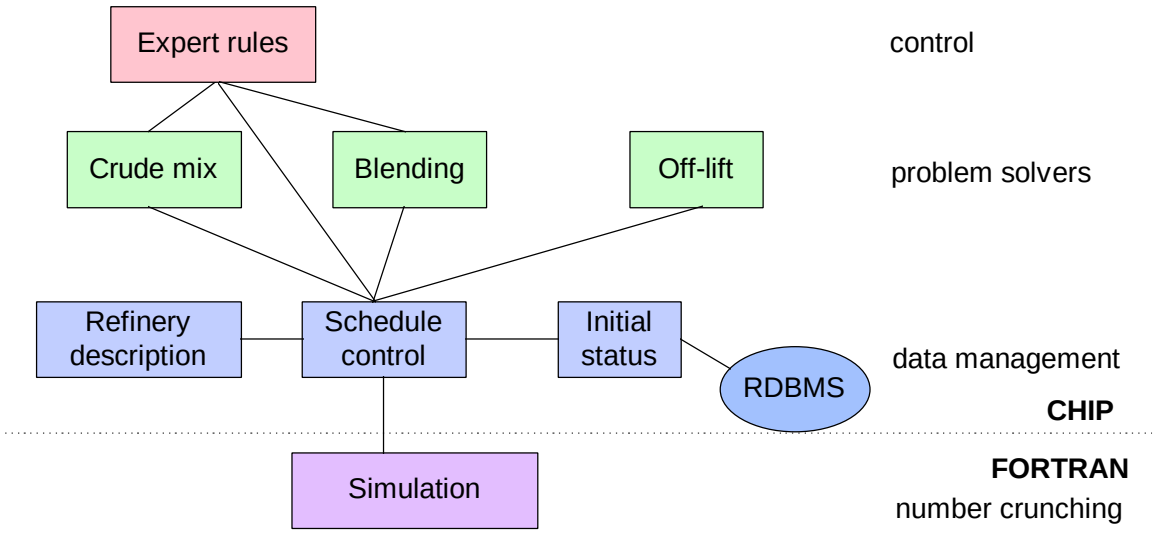


Figure 7: FORWARD system modules

Oil mixes with non-linear blending laws for many properties. The FORWARD system tracks around 70 measured values and uses up to 1000 data points for each oil mix in the system. There is a library of several hundred crude oils, defining their properties in detail. The FORTRAN part of the FORWARD system uses these data and complex non-linear models of processing units to calculate flow-rates, yields and properties for each unit in

the refinery. Different operating modes determine the type of behaviour expected from the unit.

The system modules in the FORWARD system are shown in figure 7. The simulation part in FORTRAN handles the calculation of unit yields and flow rates. It exchanges information about the refinery structure with the scheduling control system. This event processing system written in CHIP is also linked to the other parts of the system. The refinery description module is a graphical editor to enter and modify refinery descriptions interactively. All units, tanks and pipes in the refinery are entered using this module.

In the initial status module, data from the refinery data base are loaded to initialise the simulation. An automatic comparison with the results of previous simulation runs detects inconsistencies and unforeseen events. On top of this data management part, there are several problem solvers, written in CHIP. One module controls the mix of crude oils in order to maximise the throughput of the CDU. A large non-linear optimisation module is responsible for blending optimisation. It is implemented on top of the linear rational solver of CHIP.

For each blend, it calculates the optimal recipe to produce end products with very strict quality limits at minimal cost. As this optimiser is used inside the simulation tool, we often encounter over constrained problems, which allow no solution. A special explanation tool, written as a meta program on top of the problem solver, applies constraint relaxation in a variable, four-level hierarchy. It either finds the maximal achievable quantity or determines the missing component.

Another problem solver is responsible for the scheduling of ship arrivals at the refinery. This has to take the limited berthing capacity at the refinery into account as well as the storage limits for finished products.

On top of these modules, an expert rule system allows to recognise and react to standard situations in the refinery. These rules explain for example which tank to use next when a crude oil tank becomes empty or full. The rules are entered graphically using a special rule editor, which guarantees syntactic correctness of the rules.

The FORWARD system clearly shows the flexibility obtained by the CLP approach. The three current users belong to different companies, with different management styles and slightly different objectives in running the refinery. Each site requires custom modifications to take new units or special physical properties into account. At some sites, pipeline operations must be scheduled concurrently with the refinery operations.

Different users use different blending laws which directly affects the blending optimiser. This problem solver is customised for each user to reflect their different view point. The interfaces to other tools and to the data base are customised as well. As a large part of the system is developed in CHIP using CHIP++ objects, it is easy to extend or modify the function of different modules.

7. Application Framework

We now briefly discuss some aspects of the application framework used for many of the applications shown above. A typical decision support application consists of different aspects. A central data model is used to drive the graphical user interface, the problem solver and reporting modules. The data are stored in a data base which exchanges data

with the internal data model. Libraries are used for many of these functions in order to minimise application specific code.

The application framework for CHIP provides such building blocks for the data management and graphical user interface. Based on a central data model as a data description file, it automatically creates CHIP++ class descriptions, generates data base tables and code to upload and download data from the data base. In the same way, it is used to manipulate graphical interface modules like lists or dialog panels. All these components share the data description which is a single point of reference for the data model of the application. The framework also contains libraries for often required graphical tools like Gantt charts or diagrams.

The system is general enough for different application domains, it has been used for problems of manufacturing, transport, chemical industry production scheduling, or human resource management. There are obvious benefits in the code size and development speed, but also improved quality and a decrease in maintenance requirements.

8. Analysis

In this section we want to evaluate some aspects of the application development with constraints. We first look at an analysis of the different parts of the ATLAS application and then compare a number of large scale CHIP applications.

8.1 Needs of an Industrial System

Table 2 shows the percentage of the overall size of different parts of the ATLAS application. The largest amount is taken up by the data base and integration parts, which are developed using SQL*Forms and Reports from Oracle. Of the CHIP part, the application specific graphics and graphical libraries on top of XGIP take a third of the overall program size. The problem solver only accounts for 9%, while the remainder is split between fixed parts for I/O, data management and dialogs.

Part of application	Percentage of overall size
Data base/integration	46%
Graphics	19%
Graphics libraries	14%
Problem solver	9%
Static program parts (I/O, dialog description)	6%
Others	6%

Table 2: Percentage of application size

This table gives the percentages of code size of the finished application. In terms of development effort, a larger percentage must be attributed to the constraint solving part. Prototyping and experimenting with different strategies leads to re-writes or modifications of major parts of the constraint module. In terms of overall effort on the application, the solver accounts for perhaps 20% of the overall development time.

The problem solver is also the high development risk part of the application. Very often, it is not clear from the beginning which aspects of the system can be integrated in the

solver. Yet without a problem solver, the overall application does not perform its job. This means that prototyping and risk reduction are key aspects to a successful CLP project.

But even taking this into account, the table clearly shows the importance of the integration and data manipulation parts of the finished system. With the application framework, we are working on reducing the amount of special purpose code required for this part of the system in the same way as the constraint system has reduced the amount of work on the problem solver.

8.2 Application Comparison

The following table 3 compares a number of large scale CHIP applications co-developed by COSYTEC. The entries are given in temporal sequence of their project start. The table shows the name of the system and the constraint solver used inside the program (F means the finite domain solver, R the linear rational solver). The next two columns state whether the program contains a graphical user interface and data management parts. The number of lines of the complete application is given next. An asterisk means that the complete system contains parts written in other languages, which are not counted here. The last two columns give the number of lines in the constraint solving part of the application and the elapsed project time. This number is *not* the total size of the project in man months.

System	Solver	GUI	Data Mgnt	Lines	Solver	Duration
Locarim	F	yes	yes	70000	3000	24 month
Forward	R,F	yes	yes	60000*	5000	24 month
ATLAS	F,R	yes	no	15000*	4000	18 month
TAP-AI	F	no	no	7000*	6000	6 month
TACT	F	yes	yes	36000	7000	10 month

Table 3: Application comparison

It is interesting to see that graphical user interfaces and data management modules account for a large percentage of the overall code size. Building a complete end-user system also requires a significant time period for specification and changes of the business process to use the new system to advantage. The TACT application shows how the use of the application framework increasingly reduces both project time and code size.

The constraint solver parts are of a rather similar size. As new global constraints are added to the system, we can handle more and more complex problems which were unmanageable before. We therefore see an increase of complexity in more recent applications which has no significant impact on the project duration.

8.3 Why use Prolog for CLP?

All applications mentioned above use the PROLOG version of the CHIP system. While the declarative structure of logic programming is often advocated as a main advantage of Prolog, we find other aspects of at least equal importance.

For application development, the interpreting environment of Prolog is very important. Changes and revisions can be loaded without recompiling and linking the complete system by re-consulting small parts of the system. This speeds up the development of the

problem solver as well as the coding of the graphical user interface. The built-in relational form of Prolog is also very convenient to store data and parameters inside the application.

The backtracking mechanism of Prolog is most important for developing complex assignment strategies in the problem solver. While pre-defined search procedures can give satisfactory results on small problems, they are not sufficient to solve large, complex problems. Developing such search strategies in a conventional language like C or C++ dramatically increases complexity.

Another useful aspect of logic programming are meta-programming facilities which can analyse programs and generate new code dynamically. Several of the presented applications use model generator programs or diagnosis engines as explanation facilities. Expert rule systems written in Prolog (with suitable graphical interfaces) allow end-user programming for example in the FORWARD or OPTISERVICE systems.

On the other hand, there are several problems which must be overcome when using logic programming for large scale application programming. The largest problem clearly is the lack of acceptance of PROLOG in the industry, which creates both commercial and technical problems. Writing efficient logic programs is clearly possible, but requires training and experience. This problem is aggravated for constraint programming where there is no strict separation between modelling and implementing tasks.

Other problems are of a more technical nature. Prolog does not immediately support a graphics model based on callbacks. This requires exchange of global data between queries, which is normally not well supported in Prolog. It also lacks proper data abstraction tools. Symbolic terms are rather weak data structures, which cause problems for re-use and specialisation of existing code. These problems are overcome in CHIP with the help of the CHIP++ object layer, which adds feature-terms, objects and class structure to PROLOG.

9. Does CLP Deliver?

In section 3, we mentioned several key advantages which were claimed for constraint logic programming. Short development time was the first of these advantages. This is clearly true for many of the prototypes which have been developed very rapidly in CHIP in order to understand and define the problem to be solved. For large scale applications, time limits are usually given by the project framework and possible changes to the business environment. Basically constraint systems enable us to remove the problem solver from the critical path in the project planning.

Compared to a standard approach, constraint applications also use a lot less code than conventional programs. The use of the application framework increases this tendency by also shrinking the amount of custom programming for other parts of a system like graphics or data management. The programs are also quite easy to modify and to extend. A good example is the FORWARD system, which has been quite easily adapted to three different users and problems as described above.

The performance of the problem solver is typically not the limiting factor in the application. Answers are achieved from within seconds to within a few minutes for the complex constraint problems. This is usually quite acceptable compared to the time required to enter data or to make manual modifications.

10. Conclusions

We have shown in this paper a number of large scale applications developed with CHIP, a constraint logic programming system. We have seen that complex end-user systems can be built with this technology. On the other hand, constraint logic programming does not offer a “silver bullet” for problem solving. There is a significant learning period before the technology can be mastered. This also requires application domain knowledge to model problems and to express solution strategies. Other aspects, like integration and graphical user interface, form a significant part of the overall development requirements. Clearly, constraint logic programming is now an industrial reality which can be applied to many domains where high quality decision support systems are required.

11. References

[AB93] A. Aggoun, N. Beldiceanu
Extending CHIP in Order to Solve Complex Scheduling Problems
Journal of Mathematical and Computer Modelling, Vol. 17, No. 7, pages 57-73
Pergamon Press, 1993

[BKC94] G. Baues, P. Kay, P. Charlier
Constraint Based Resource Allocation for Airline Crew Management
ATTIS 94, Paris, April 1994

[BC94] N. Beldiceanu, E. Contejean
Introducing Global Constraints in CHIP
Journal of Mathematical and Computer Modelling, Vol 20, No 12, pp 97-123, 1994

[BCP92] J. Bellone, A. Chamard, C. Pradelles
PLANE -An Evolutive Planning System for Aircraft Production.
First International Conference on the Practical Application of Prolog. 1-3 April 1992, London.

[Ber90] F. Berthier
Solving Financial Decision Problems with CHIP
Proc 2nd Conf Economics and AI, Paris 223-238, June 1990

[BLP95] R. Bisdorff, S. Laurent, E. Pichon
Knowledge Engineering with CHIP - Application to a Production Scheduling Problem in the Wire-Drawing Industry
PAP95, Paris, April 1995

[BDP94] P. Bouzimault, Y. Delon, L. Peridy
Planning Exams Using Constraint Logic Programming
2nd Conf Practical Applications of Prolog, London, April 1994

[CDF94] A. Chamard, F. Deces, A. Fischler
A Workshop Scheduler System written in CHIP
2nd Conf Practical Applications of Prolog, London, April 1994

[CF95] C. Chiopris, M. Fabris
Optimal Management of a Large Computer Network with CHIP
2nd Conf Practical Applications of Prolog, London, April 1994

[CGR95] T. Creemers, L. R. Giralt, J. Riera, C. Ferrarons, J. Rocca, X. Corbella
Constrained-Based Maintenance Scheduling on an Electric Power-Distribution Network
PAP95, Paris, April 1995

[Col90] A. Colmerauer
An Introduction to Prolog III
CACM 33(7), 52-68, July 1990

[DVS88] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf and F. Berthier.
The Constraint Logic Programming Language CHIP.
In Proceedings of the International Conference on Fifth Generation Computer Systems (FGCS'88), pages 693-702, Tokyo, 1988.

[DSV87] M. Dincbas, H. Simonis, P. Van Hentenryck.
Extending Equation Solving and Constraint Handling in Logic Programming.
In Colloquium on Resolution of Equations in Algebraic Structures (CREAS), Texas, May 1987.

[DSV90] M. Dincbas, H. Simonis and P. Van Hentenryck.
Solving Large Combinatorial Problems in Logic Programming.

Journal of Logic Programming - 8, pages 75-93, 1990.

[DS91] M. Dincbas, H. Simonis
APACHE - A Constraint Based, Automated Stand Allocation System
Proc. of Advanced Software Technology in Air Transport (ASTAIR'91)
Royal Aeronautical Society, London, UK, 23-24 October 1991, pages 267-282

[DSV92] M. Dincbas, H. Simonis, P. Van Hentenryck
Solving a Cutting-Stock Problem with the Constraint Logic Programming
Language CHIP
Journal of Mathematical and Computer Modelling, Vol. 16, No. 1, pp. 95-105,
Pergamon Press, 1992

[DSV88] M. Dincbas, H. Simonis, P. Van Hentenryck.
Solving the Car Sequencing Problem in Constraint Logic Programming.
In European Conference on Artificial Intelligence (ECAI-88),
Munich, W. Germany, August 1988.

[GVP89] T. Graf, P. Van Hentenryck, C. Pradelles, L. Zimmer
Simulation of Hybrid Circuits in Constraint Logic Programming
IJCAI, Detroit, August 1989

[JL87] J. Jaffar, J.L. Lassez
Constraint Logic Programming
Proc. 14th POPL, Munich ,1987

[JM94] J. Jaffar M. Maher
Constraint Logic Programming: A Survey
Journal of Logic Programming, 19/20:503-581, 1994

[KS95] P. Kay, H. Simonis
Building Industrial CHIP Applications from Reusable Software Components
PAP95, Paris, April 1995

[SND88] H. Simonis, N. Nguyen, M. Dincbas
Verification of Digital Circuits using CHIP
In G. Milne (Ed.), The Fusion of Hardware Design and Verification, pages 421-442,
North Holland, Amsterdam, 1988

[SD93] H. Simonis, M. Dincbas
Propositional Calculus Problems in CHIP
In A. Colmerauer and F. Benhamou, Editors,
Constraint Logic Programming - Selected Research, pages 269-285, MIT Press, 1993

[SC95] H. Simonis, T. Cornelissens
Modelling Producer/Consumer Constraints
Proc. Principles and Practice of Constraint Programming, Cassis, France, September 1995

[VH89] P. Van Hentenryck.
Constraint Satisfaction in Logic Programming.
MIT Press, Boston, Ma, 1989.

[VSD92] P. Van Hentenryck, H. Simonis, M. Dincbas
Constraint Satisfaction using Constraint Logic Programming
Journal of Artificial Intelligence, Vol.58, No.1-3, pp.113-161, USA, 1992

[VC88] P. Van Hentenryck, J-P. Carillon.
Generality versus Specificity: an Experience with AI and OR Techniques. In American Association for Artificial Intelligence (AAAI-88), St. Paul, Mi, August 1988.