<u>**Research Proposal Form**</u>

Date: 9th June, 1997

1 Contractor or Principal Investigator

- **Organization**

  Research Institute for Symbolic Computation (RISC-Linz)

- **Address of the Organization**

  Johannes Kepler University, A-4040 Linz, Austria

- **Title**

  Professor

- **Name**

  Bruno Buchberger

- **Contact**

Address:
Research Institute for Symbolic Computation (RISC-Linz),
Johannes Kepler University, A-4040 Linz, Austria
TEL: +43 7236 3231 41
FAX: +43 7236 3231 30
e-mail: Bruno.Buchberger@risc.uni-linz.ac.at

2 Title of the Proposed Project

(1) Title of the Proposed Project:

Distributed Constraint Solving for Functional Logic Programming

(2) Research Period (one-year project or two-year project):

Two-years project.

(3) Background of the research (about a half page):

In February 1987, at the occasion of a research visit to ICOT, Prof. Bruno Buchberger proposed using advanced methods in computer algebra (the Gröbner bases method, quantifier elimination, etc.) for dealing with nonlinear constraints in logic programming. The use of algebraic methods in non-linear constraint logic programming has meanwhile been pursued by various groups in the world. At RISC, the emphasis was on non-linear and quantifier constraint solving including the development of distributed and parallel algorithms with a new degree of efficiency in particular in practical examples. At the same time, at the SCORE group of Professor Ida at Tsukuba University, the research line of functional logic programming was developed in detail, in particular with respect to powerful narrowing techniques. Recently, a strong interaction between the two groups evolved. It is the goal of the project proposed here to bring this joint expertise into the frame of the ICOT research and application program.

(4) Purpose of the research (about a half page):

We propose the development of a distributed software system consisting of

- a functional logic language interpreter on one machine,
- a number of constraint solving engines running on other machines.

The interpreter is based on an already existing (sequential) implementation of a functional logic language on the computer system Mathematica extended in two directions:

- the possibility to specify (non-linear) constraints over real numbers;
- the possibility to specify OR-parallelism among different clauses of a predicate.

The OR parallel features of the interpreter allow the decomposition of the solution space into different subspaces denoted by various sets of constraints; the individual sets are solved by different constraint solving engines in parallel and joined together to form the total solution set. This allows to investigate problems with large solution spaces using the computational power available in large computer networks.

(5) Contents of the research (around 1 to 2 pages):

The expressiveness of most constraint logic languages is restricted to linear constraints; however substantial progress has been made to incorporate also non-linear constraints. At the Research Institute for Symbolic Computation (RISC-Linz), Dr. Hong and his research group on constraint solving have developed prototypes of systems aiming at the support of non-linear constraints into constraint logic programming [2]. In particular, they have been building the software library "Sturm" aiming at a constraint solver that takes as its input a "theory" over the real numbers (specified by function and predicate definitions involving quantifiers) and a "question" (a quantified logic formula). The solver returns as an "answer" a formula that is (within the specified theory) equivalent to the question but does not contain any quantifiers. The library has been implemented in C++ and is portable to all kinds of Unix machines; parts of it have been parallelized on top of the parallel programming interface MPI.

Furthermore, the parallel computation group at RISC-Linz has a long standing experience in the integration of parallel programming languages and systems for purposes of symbolic computation. Dr. Schreiner has developed a para-functional programming system where a functional programming language serves as a coordination language for parallel computer algebra algorithms implemented in the library SACLIB on shared memory multiprocessors [3]. Other activities in this grop have integrated the guarded Horn clause language STRAND with the computer algebra system MAPLE running on networks of computers.

In recent years efficient implementations of the execution principles for functional logic languages have been developed. Most of them are based *narrowing*, a unification-based parameter passing mechanism which subsumes rewriting and SLD resolution. The Symbolic Computation Research Group (SCORE) of Prof. Ida at the University of Tsukuba has a long experience in the area of narrowing with a number of important results [1]. Recently they have developed a new deterministic lazy conditional narrowing strategy and implemented it in the computer algebra system Mathematica.

Based on our combined experience, we propose an integration of the constraint solving package developed at RISC-Linz into the lazy narrowing system developed at SCORE such that the functional logic language serves as a "coordination language" distributing sets of constraints to a network of constraint solving engines built on top of the Sturm library. For this purpose, the functional logic language and its Mathematica implementation are extended by the mechanism to specify non-linear constraints over the real numbers and to formulate OR parallelism among different clauses of a predicate such that parts of the solution space can be investigated independently and simultaneously of each other.

The actual distribution is performed by an external "scheduling program" implemented in C++ that is linked to the Mathematica kernel via the "MathLink" facility and provides the interface between the Mathematica language interpreter and the various constraint solving engines. The scheduler receives from the Mathematica kernel OR-parallel branches of the computation represented as Mathematica expressions containing sets of constraints to be solved. The scheduler forwards each constraint set to some constraint solving engine available in the network. Each engine implements a sequential constraint solver based on the Sturm package; it communicates with the scheduler retrieving a set of constraints and delivering the corresponding solution. The scheduler holds a set of those OR-parallel branches whose constraints have been solved and returns them to the language interpreter (which in turn may generate new branches). The Mathematica language interpreter itself therefore still executes in a sequential fashion cooperating with the scheduler which is in charge of the distributed execution.

The whole approach allows to investigate problems with large solution spaces (as arise in numerous scientific and engineering problems) using the computational power available in (local- or wide-area) computer networks.

Selected Publications

[1] Hoon Hong: RISC-CLP(Real): Constraint Logic Programming over Real Numbers. In: Constraint Logic Programming: Selected Research, F. Benhamou and A. Colmerauer (eds.), MIT Press, 1993.

[2] Tetsuo Ida et al: Lazy narrowing: strong completeness and eager variable elimination, Theoretical Computer Science, 167(1996), 95-130

(Conference Version: Proceedings of the 20th Colloquium On Trees In Algebra and Programming (CAAP) Lecture Notes in Computer Science 915, 394-409,1995)

[3] Wolfgang Schreiner: A Para-Functional Programming Interface for a Parallel Computer Algebra Package. Journal of Symbolic Computation, Special Issue on Parallel Symbolic Computation, Hoon Hong (ed.), volume 21, pp. 523-614, Academic Press, 1996.

3 Project Organization/Research Method/Schedule

(1) Project Organization

	Name	Affiliation	e-mail address
Princ. Investigator	H. Hong	RISC-Linz	hong@risc.uni-linz.ac.at
Coop. Researcher	T. Ida	Tsukuba	ida@score.is.tsukuba.ac.jp
Coop. Researcher	W. Schreiner	RISC-Linz	schreiner@risc.uni-linz.ac.at

(2) Research Method

The work will be carried out at the RISC-Linz by Dr. Hong and Dr. Schreiner. Guided by Dr. Hong (constraint solving expert), Dr. Schreiner (parallel/distributed computing expert) and in close contact with Prof. Ida (functional logic language expert) via the Internet, a PhD student will perform most of the actual development and implementation work. The main funds will be needed for paying this student. Several excellent candidates for this work are available. The group of Prof. Ida will continously improve the narrowing implementation in Mathematica, while the group of Dr. Hong will continue to extend the features of the Sturm library. Dr. Schreiner will supervise the work on the distribution and parallelization features of the proposed system.

(3) Research Schedule: schedule for two years for two-year project

- July 1997–October 1997: adaption and extension of the functional logic language interpreter.
- November 1997–April 1998: implementation of scheduler and constraint solving engine (communication interface).

- May 1998–June 1998: debugging and testing.
- June 1998: prototype version operational in local area networks.
- July 1998–December 1998: developing example applications, improving and extending the system.
- January 1999–March 1999: experimenting with the system on large area networks, final adjustments.
- April 1999–June 1999: development of final documentation and packaging.

4 Resulting Software

(1) The name of the software:

Distributed Functional Logic Constraint Solver (DIFLOCOS).

(2) Functions and features of the software (more than a half page)

The software will allow to write programs in a functional logic language based on a variant of narrowing (deterministic lazy conditional narrowing) including the possibility to specify non-linear constraints over real numbers and to formulate OR parallelism among different clauses of a predicate. When the language interpreter is started on a program and a query, an external scheduler process is created that contacts a list of hosts (taken from some standard file) and starts constraint solving engines on that hosts.

Program execution will consist of cooperation between the Mathematica language interpreter, scheduler, and constraint solving engines as described in the previous sections. On termination, the program delivers a return value and a description of the solution space expressed in terms of the free variables of the query.

The system will be implemented as an open library with well-documented interfaces, which allows its use as the core of a toolkit for distributed functional logic programming with constraints.

(3) Structure of the software (about a half page)

- Structure of the software

  The software consists of three parts:

1. An extension of an existing sequential implementation of deterministic lazy conditional rewriting developed at SCORE based on the computer algebra system Mathematica.
2. A scheduling program implemented in C/C++ communicating with the Mathematica kernel via the “MathLink” facility and with the constraint solving engines via MPI.
3. A constraint solving engine implemented in C/C++ on top of the “Sturm” library developed at RISC-Linz communicating with the scheduler via MPI.

Communication between the Mathematica/scheduler processes uses sockets as communication channels. Communication between scheduler and constraint solving engine communication is based on some of the available public domain implementation of MPI (such as MPICH, CHIMP, LAM, ...).

- If the resulting software will be a part of larger system, please explain the relation of the software to the total system.

  The resulting software can be considered as part of the Theorema project being pursued by Prof. Buchberger at RISC-Linz that aims at an integrated environment for mathematical problem solving based on the computer algebra system Mathematica. The proposed system will become one component of this environment representing the relationship between functional/logic programming and constraint solving.

- If the resulting software will be improvement of existing software by adding new functions and etc., please explain what kind of functions are added.

  The resulting software extends the power of the lazy conditional narrowing system developed at SCORE by facilties for constraint solving (non-linear constraints over the real numbers) and for distributed processing.

(4) If the software is related to specific IFS programs, please specify their names.

(5) Please specify the programming language/OS/software packages for the execution of the software, and please explain the portability of resulting software.

The software basis required for the proposed system are

- Mathematica 3.0 for the language interpreter,
- C/C++ for the scheduler,
- a distributed implementation of MPI (such as MPICH, CHIMP, or LAM),
- the C++ based library STURM developed at RISC-Linz.

Of these components, only Mathematica is a commercial program; it is available for a wide variety of machines running under different operating systems (Unix, Windows 95/NT, etc).

All other components are available in public domain implementations for numerous kinds of machines.

The development will be pursued at RISC-Linz on a Sun UltraSparc workstation (for the language interpreter) and a network of Silicon Graphics workstations (for the constraint solving engines).

(6) Expected size of the software (restricted to newly developed part)

The newly developed parts will be

- an extension of the language interpreter (about 500 lines of Mathematica source code),
- the scheduler (about 5000 lines of C/C++ code, a few hundred KBs of object code),
- the communication interface of the constraint solving engine (about 1000 lines of C/C++ code).

(7) Expected way of use of the software (more than a half page)

- How this software will be used, who will be expected users.

  The software will be used by researchers in logic programming languages and in functional logic programming languages as well as by researchers in constraint solving who will utilize its expressive power (solving systems of non-linear constraints over the reals within a functional logic programming language) and its computational power (using the power available in local-area computer networks).

  The software also represents an interesting experiment by the integration of the computer algebra system Mathematica in distributed applications; a large interest of the huge computer algebra community using Mathematica (and also by Wolfram Research, the developers of Mathematica) is guaranteed. The functional logic programming model provides a high-level interface to engineers and scientists dealing with large systems of constraints as they arise in numerous applications.

- Advantage of the software from users' point of view.

  The main advantage of the software will be its easy access (large portability by using industry standards such as Mathematica, C++, MPI) and the easy accessability of local computing resources (computer networks). Using Mathematica as the programming interface immediately provides all the support of the Mathematica environment concerning presentation and publishing and the reuse of the results in other computations (in contrast to proprietary stand-alone systems not integrated with computer algebra systems).

  Furtermore, the open design and modularity of the system will allow its use as a toolkit for own developments in distributed functional logic programming with constraints.

(8) Expected intermediate result at the end of the first fiscal year (if the project is a two-year project)

After the end of the first fiscal year, a first prototype version of the system for running experiments in local area networks will be available.

(9) Documents which will be added to the software

- Installation description and user manual: how to install the individual parts of the software and how to use them.
- Language specification: the functional logic language and its concurrency and constraint solving interface.
- System interface: the interface of the underlying library is documented to support the use of the system as the core of a toolkit.
- Tutorial: a tutorial description comprising a few programming examples for the system.
- A Web server: providing the newly developed parts software and all its documentation in electronic form. Links to the basis software (Mathematica, MPI package, Sturm library) will be included.

5 Bases of Funding Money

The following table lists the funds required for each of the two years of the research project:

Item	Amount	Bases	Remarks
Personnel Expenses	1.200.000,-	PhD student at RISC-Linz	
Travelling Expenses	100.000,-	travel	
Materials Purchase Expenses	50.000,-	literature	
Expendables Expenses	50.000,-	printing and copying	
Miscellaneous Expenses	100.000,-	system administration	