Distributed Constraint Solving for Functional Logic Programming*

AITEC Research Funding Programme
Progress Report 1997

Prof. Bruno Buchberger
Dr. Wolfgang Schreiner
{Bruno.Buchberger,Wolfgang.Schreiner}@risc.uni-linz.ac.at
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, Linz, Austria

March 1, 1998

Abstract

We report on the development of a distributed software system consisting of a functional logic language interpreter on one machine, and a number of constraint solving engines running on other machines. The interpreter is written in the language of the computer algebra system Mathematica with communicating to the constraint solvers via the MathLink protocol.

Contents

1 Introduction

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 non-linear 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-Linz, 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 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 pursue in the frame of this project the integration of constraint solving packages 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 engine. 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.

2 Progress Report

Our project work started in September 1997. The first three months of the project were dedicated to the design of the interpretation mechanism: the lazy narrowing calculus developed by Prof. Ida was extended towards parallelism and constraint solving for which it was corresponingly equipped it with a type system. In September and October, Prof. Buchberger and Mircea Marin from RISC-Linz spent six weeks with the group of Prof. Ida at the University of Tsukuba to elaborate the details in cooperation.

The next tree months of the project (till the end of February 1998) were dedicated to specifying the extended calculus and to implementing the extensions in the functional logic language interpreter. The resulting language and calculus are documented in detail in Appendix A.

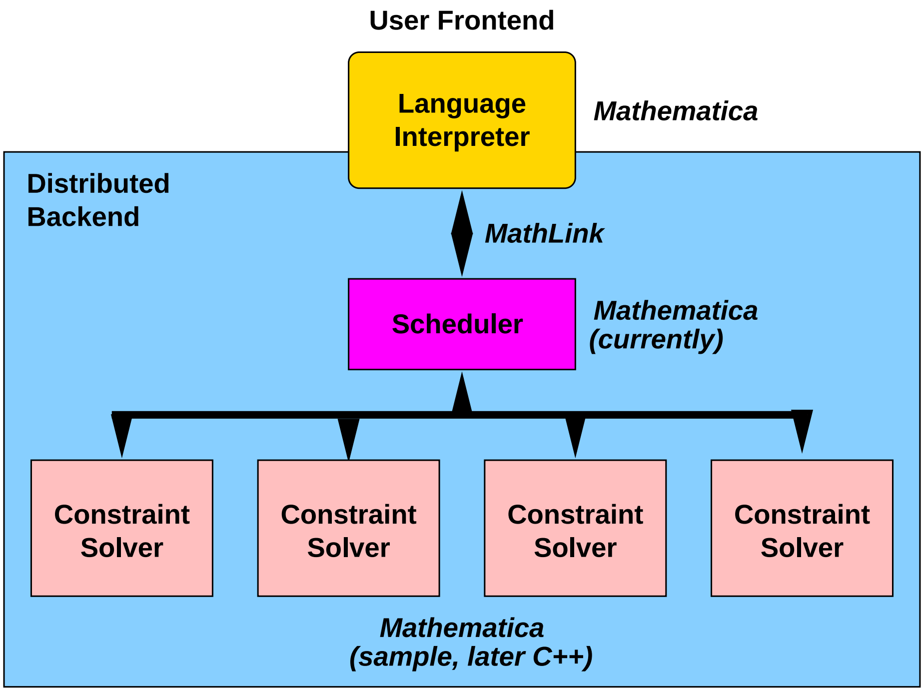

Figure 1: The Distributed Language Environment

The mplementation work included the development of an interface to external constraint solvers and the introduction of the following processes running external to the interpreter (see Figure 1):

- the prototype of a scheduler for distributing the constraints generated by the interpreter to different constraint solving engines;
- a linear constraint solver as a sample for the integration of external solvers with the interpreter.

For the time being, both the scheduler and the sample solver are implemented as separate Mathematica processes communicating via the MathLink protocol (but still running within one machine in a pseudo-parallel fashion). The scheduler asynchronously collects systems of constraints generated during the execution of the interpreter and forwards them to the constraint solvers. The simplified solutions are collected and transferred back to the interpreter.

The prototype implementation of these components in Mathematica has saved considerable development time for setting up the core skeleton of the system; however it already allows interesting experiments with the dynamic behavior of programs under various execution strategies.

3 Further Work

After half a year of project work, we have set up the basic skeleton of the system. Within the next year of the project, we will fill this skeleton with "flesh" pursuing research and development work along the following lines:

1. **Constraint Solvers:** The next step will be the integration of external solvers implemented in a compiled language (C/C++) over more powerful constraint domains (as have been developed at RISC-Linz). This will involve the definition of a corresponding communication interface on top of the MathLink protocol for the exchange of constraints between the Mathematica environment and the external solvers.

2. **Process Distribution:** Currently, all constraint solving processes run on the same machine as the language interpreter. This limitation will be overcome by placing some infrastructure on top of the current runtime environment by which external solvers are spawned on other machines and the scheduler connects to these solvers. Issues like load distribution and a basic form of fault tolerance have to be investigated in this phase.

3. **Calculus Extensions:** The current inference calculus has various restrictions that we would like to overcome in future versions. For instance, it only allows a restricted form of higher-order functions/predicates and control of parallelism within the program is still rather rudamentary. We will thus continue work on the calculus in close cooperation with Prof. Ida.

This work will be based on the development of more realistic application examples from various domains such as electrical engineering. The development of a library of such examples will be therefore of great importance.

A A Mathematica Implementation of the CFLP Calculus

This section is contained in a seperate document. It describes the theoretical framework of the language, the resulting execution calculus, and its embedding into the Mathematica environment.