Designing a Symbolic Solver for Arithmetic Constraints for Computer Assisted Learning

Ana Paula Tomás, Nelma Moreira, Nuno Pereira

Technical Report Series: DCC-05-06

Departamento de Ciência de Computadores – Faculdade de Ciências

&

Laboratório de Inteligência Artificial e Ciência de Computadores

Universidade do Porto

Rua do Campo Alegre, 823, 4150 Porto, Portugal

Tel: +351+22+6078830 – Fax: +351+22+6003654

http://www.dcc.fc.up.pt/Pubs/treports.html

Designing a Symbolic Solver for Arithmetic Constraints for Computer Assisted Learning *

Ana Paula Tomás, Nelma Moreira, Nuno Pereira
R. do Campo Alegre, 823
4150-180 Porto, Portugal
{apt,nam,nfp}@dcc.fc.up.pt

May 2005

Abstract

We present a solver for arithmetic and membership constraints over real numbers, for computer assisted learning (CAL) in math. The solver works as a rewrite system. What makes it novel are the proposed rewriting rules that go beyond simple algebraic reductions. Instead they work at high abstraction level and use some knowledge about the functions behaviour to shortcut solving steps. Designed with pedagogic concerns, they are useful to produce step-by-step solutions that look like ones worked out by students. In this way the solver may be more advantageous in some learning environments than sophisticated mathematical software, which is certainly the choice for scientific applications and advanced algebraic manipulations. The proposed solver is correct and although it is complete for a relevant set of problems arising in high-school math curricula, it is incomplete in general. Indeed, this is inherent to the addressed problem. A prototype is being developed in Prolog for a specific application domain, reusing some modules of Demomath [14] for symbolic manipulation of sets and exact arithmetic, that employ CLP($\mathbb{Q}$) and CLP($\mathbb{R}$) to some extent. This work is part of AGILMAT project in which a web application to automatically generate and solve math drills is being developed.

1 Introduction

The fast growth of the Internet has fostered a significant breakthrough in CAL. Sophisticated web-based learning environments are being developed also for math education, some offering authoring facilities for teachers to create courseware, assignments and exams, some being used for training, assessment and contests [9, 4, 7, 15, 16]. Teachers are urged to develop innovative learning scenarios. Nevertheless, designing courseware material for e-learning is often quite time-consuming, even when teachers may count on e-learning authoring tools [6, 15]. It is worth observing that a large number, if not the majority, of e-learning tools that create math drills are oriented for the generation of multiple choice questions. Some of these applications may support interaction with computer algebra or other mathematical systems to reduce the effort of writing a solver [15].

*Work partially supported by Fundação para a Ciência e Tecnologia (FCT) and Program POSI, co-financed by EC fund FEDER, under project AGILMAT (contract POSI/CHS/48565/2002).

Nevertheless, what seems to be the most current practice is to write a full implementation of an ad-hoc solving (or decision) procedure for each problem template [6, 15]. The problems each template yields very often look like quite simple manipulations of a basic pattern. Advances in computer technology shall therefore be exploited to develop really interactive, re-usable and customizable contents. In the spirit of ActiveMath, the LeActiveMath project targets an innovative third-generation e-learning system: user-adaptive, interactive, web-based environment and employing intelligent technologies [9, 4]. Also, in the MowGli project XML-based tools are developed to enhance accessibility, searching, retrieving, and elaboration of mathematical knowledgement over the Web [7].

Our project – AGILMAT – whose full title is *Automatic Generation of Interactive Drills for Mathematics Learning*[1], aims at the design and implementation of a system to automatically generate and solve math exercises. Along the lines of our former simple prototype Demomath[2], the study of algebraic algorithms taught at high-school and of the way they condition the drills that may be created and solved automatically plays a major role in AGILMAT [14]. The form of exercises is described by grammars. For instance, a grammar that characterizes a wide range of the function expressions that may be found in high school textbooks and whose zeros and domains can be exactly computed was proposed in [14]. Several other problems depend on this central issue, as for instance, the analytic study of monotonicity, concavity and convexity of real-valued functions, so that its interest goes far beyond the problem itself. Algebra and Infinitesimal Calculus at pre-College levels are the major applications domains of our work. By imposing extra conditions on the more generic grammars through constraints, users may cast exercises to different curricula or level of education almost for free. For example, for Demomath, the teacher/user may make the system produce very distinct expressions for the exercises, by filling in a sequence of forms to define user constraints, in a web browser.

Experiments with Demomath gave us fundamental support to continue using Constraint Logic Programming languages, namely Prolog-based ones, to develop core modules of AGILMAT: the exercise generators and the solvers. Whereas computer algebra systems are quite adequate for exploratory learning, they would not work well for our target application. In particular, our experience with *Maple*[3] has shown that the algebraic simplifications it does are troublesome. Additional constraints must be imposed on the expressions that arise in the exercises, to avoid inconsistencies in explanations. For instance, it is not possible to pretty print $3(x^2 + 5)$ in *Maple* since it will naturally yield $3x^2 + 15$. By a similar reason, we would better not ask the student to find the domain of a rational function defined by $f(x) = (x - 1)^2/(x - 1)$, because that expression would be printed as $f(x) = x - 1$, and hence 1 belongs to domain of the latter but not of the former one. This incorrect and misleading behaviour of mathematical software and the resulting disadvantages in some learning environments are discussed also in [2]. In addition, we need great, if not full, control of the system, to be able to produce explanations.

Logic Programming based languages offer natural support for implementing symbolic representations and to do symbolic manipulations. Declarativeness is of help to specify the form of the expressions and of the problem templates. Moreover, for some problems in mathematics, we have to do exact computations and present the results in simplified forms. As we mentioned already, to achieve re-usability, the application shall be well parametrized

[1] `http://www.ncc.up.pt/AGILMAT/`.

[2] `http://www.ncc.up.pt/~apt/demomath.html`. It shall not be viewed as a prototype of a user-friendly CAL system.

[3] with Maple V. The release version now is Maple 9.5 (`www.maplesoft.com`).

to easily cater for different curricula or user-defined constraints. CLP seems to offer the right expressiveness to encode this kind of control in an elegant way through constraints [1, 3, 8].

The solver we implemented for DEMOMATH followed a fairly ad-hoc strategy for solving arithmetic constraints, quite close to simple algebraic manipulation. This renders the program large and makes it difficult to extend it to present step-by-step resolutions, with pedagogic interest. In this paper we present a novel approach for solving arithmetic and also membership constraints, founded on a deeper analysis of the problem. By reasoning on a higher abstraction level we show how we may achieve a much more concise and modular solver. The new solver is being implemented in Prolog using CLP($\mathbb{Q}$) and CLP($\mathbb{R}$) [5, 13, 12].

The paper is organized as follows. We first recall some basic notions of real-valued functions that are needed in the paper. In Section 3 we introduce our representation for problems and constraints and show how to convert between membership and arithmetic constraints. Section 4 is devoted to the presentation of the solver as a rewrite system. The solver is complete for a family of problems involving functions described by the grammar given in the paper, which extends the one proposed in [14].

2 Some Notation and Mathematical Background

In this section we recall some notions about real-valued functions that are used in the paper and we introduce some notation. $\mathbb{R}$ stands for the set of the real numbers, a, b, c, k for real constants, f, g, h for generic real-valued functions over $\mathbb{R}$, and x, y, z for real valued variables. As usual, $\mathcal{D}_f$ is the domain of function f, i.e., the subset of $\mathbb{R}$ for which f is defined and $f(\mathcal{D}_f) = \{f(x) : x \in \mathcal{D}_f\}$ is the image (a.k.a., range) of f. The restriction of f to $D \subseteq \mathcal{D}_f$ is represented by $f|_D$ and f^{-1} is the inverse of function f whenever it exists. A function f is a strictly monotonic on $D \subseteq D_f$ if and only if it either strictly increases or strictly decreases on D. We also call f a monotonic increasing or a monotonic decreasing function. If f is strictly monotonic over D, then $f|_D$ is invertible.

A function f such that $f(x) = f(-x)$ for all $x \in \mathbb{R}$, is called even. When $f(x) = -f(-x)$, for all $x \in \mathbb{R}$, then f is called odd.

Table 1 shows the basic functions usually studied in math at high school, if we exclude the trigonometric functions (sine, cosine, tangent, co-tangent), generic polynomial functions $pol_{a_n,\ldots,a_0} : x \mapsto \sum_{i=0}^{n} a_i x^i$ and the quadratic function $pol_{a,b,c} : x \mapsto ax^2 + bx + c$.

The composition, sum, difference, product and quotient of functions (herein represented by $\circ$, $+$, $-$, $\times$, $/$) are also studied, which enable the use and construction of more complex functions. They are given by $f \circ g : x \mapsto f(g(x))$ and $f \odot g : x \mapsto f(x) \odot g(x)$, for $\odot \in \{+, -, \times, /\}$, with $\mathcal{D}_{f\circ g} = \mathcal{D}_g \cap \{x : g(x) \in \mathcal{D}_f\}$, $\mathcal{D}_{f\odot g} = \mathcal{D}_f \cap \mathcal{D}_g$ for $\odot \in \{+, -, \times\}$ and $\mathcal{D}_{f/g} = \mathcal{D}_f \cap \mathcal{D}_g \setminus \{x : g(x) = 0\}$. The functions $c_k \times f$ and c_k/f are particular cases of these, and $c_{-1} \times f$ is written also $-f$. We note also that although $p_k = c_k \times id$, we are going to keep all three as basic functions, an usual practice in high school. Table 1 is seen as useful domain knowledge that students learn and thus the solver shall also refer to. Working as a knowledge base, it may be extended if appropriate and also reduced. As in DEMOMATH, the interface of AGILMAT system will make tools available for users to configure the solvers and generators. In particular, we may prevent some rewriting rules from being used by imposing constraints on the type of functions addressed in the exercises.

f	$\mathcal{D}_f$	$f(\mathcal{D}_f)$	Behavior in $\mathcal{D}_f$	Inverse function
$id : x \mapsto x$	$\mathbb{R}$	$\mathbb{R}$	strictly increases, odd	$id^{-1} = id$
$c_k : x \mapsto k$	$\mathbb{R}$	$\{k\}$	constant, even	—
$p_k : x \mapsto kx,\ k \neq 0$	$\mathbb{R}$	$\mathbb{R}$	strictly increases if $k > 0$ strictly decreases if $k < 0$ odd	$p_k^{-1} : x \mapsto \frac{1}{k}x$
$pol_{a,b} : x \mapsto ax + b$	$\mathbb{R}$	$\mathbb{R}$	strictly increases if $a > 0$ strictly decreases if $a < 0$ odd if $b = 0$	$pol_{a,b}^{-1} : x \mapsto \frac{1}{a}x - \frac{b}{a}$
$pow_{2n+1} : x \mapsto x^{2n+1}$	$\mathbb{R}$	$\mathbb{R}$	strictly increases, odd	pow_{2n+1}^{-1}=rad_{2n+1}
$pow_{2n} : x \mapsto x^{2n}$	$\mathbb{R}$	$\mathbb{R}_0^+$	symmetric w.r.t. $x = 0$ $pow_{2n}\|_{\mathbb{R}_0^+}$ strictly increases even	$(pow_{2n}\|_{\mathbb{R}_0^+})^{-1} = rad_{2n}$
$rad_{2n+1} : x \mapsto \sqrt[2n+1]{x}$	$\mathbb{R}$	$\mathbb{R}$	strictly increases odd	$rad_{2n+1}^{-1} = pow_{2n+1}$
$rad_{2n} : x \mapsto \sqrt[2n]{x}$	$\mathbb{R}_0^+$	$\mathbb{R}_0^+$	strictly increases	$rad_{2n}^{-1} = pow_{2n}\|_{\mathbb{R}_0^+}$
$abs : x \mapsto \|x\|$	$\mathbb{R}$	$\mathbb{R}_0^+$	symmetric w.r.t. $x = 0$ $abs\|_{\mathbb{R}_0^+}$ strictly increases even	$(abs\|_{\mathbb{R}_0^+})^{-1} = id\|_{\mathbb{R}_0^+}$
$exp_a : x \mapsto a^x$	$\mathbb{R}$	$\mathbb{R}^+$	strictly increases if $a > 1$ strictly decreases if $0 < a < 1$	$exp_a^{-1} = log_a$
$log_a : x \mapsto \log_a x$	$\mathbb{R}^+$	$\mathbb{R}$	strictly increases if $a > 1$ strictly decreases if $0 < a < 1$	$log_a^{-1} = exp_a$

Table 1: Some basic functions, their domain, range, behavior and inverse.

3 Constraints and Problems

We would like to solve problems that may involve arithmetic and membership constraints, because both types coexist in some math problems. For example, if the problem was *to find all* $x \in \mathbb{R}$ *such that* $\sqrt{x-2} \geq 1$ *and* $g(x) \notin \{1, -3, 4\}$ *for* $g : x \mapsto x^2 - 2x + 1$. We define *atomic* and *complex* constraints as follows.

Definition 1 *We call* $f(x) \trianglelefteq k$ an atomic arithmetic constraint*, iff* f *is a real valued function on reals,* $k \in \mathbb{R}$ *a constant, and* $\trianglelefteq \in \{=, \neq, >, <, \leq, \geq\}$*. If* $\trianglelefteq \in \{\in, \notin\}$*, then* $f(x) \trianglelefteq S$ *is called* an atomic membership constraint*, for* $S \subseteq \mathbb{R}$ *(we would rather write* S *instead of* k *to make clear that* S *is a set). The conjunction and disjunction of a finite number of constraints in the variable* x *is* a (complex) constraint*, denoted* $C(x)$*.*

This form for atomic arithmetic constraints is generic enough since $f(x) \trianglelefteq g(x)$ may be written as $(f - g)(x) \trianglelefteq 0$, whenever g is a non-constant function. We often write C instead of $C(x)$, omitting the variable, which shall cause no confusion. Each problem must involve a unique variable.

The inverse of the binary relation $\trianglelefteq$ is denoted by $\trianglelefteq^{-1}$, for $\trianglelefteq \in \{=, \neq, \leq, \geq, >, <\}$. Clearly, $\leq^{-1}$ is $\geq$, $<^{-1}$ is $>$, and $=^{-1}$ is $=$ and $\neq^{-1}$ is $\neq$. We now go through our representation for *problems* and *problems in solved form.*

Definition 2 *The* problem *of finding all* $x \in D$ *that satisfy the constraint* $C(x)$ *is denoted by a tuple* $\langle C(x), x, D\rangle$*. A problem is in* solved form *iff it is defined as* $\langle x \in D, x, D\rangle$*. In this case* D *is called* the solution set *of the problem. To refer to a particular problem* P*, we shall associate a label to the representation, writing* $P : \langle C(x), x, D\rangle$*, and denote the solution set of* P *by* $\texttt{sols}(P)$*.*

3.1 Membership versus arithmetic constraints

It is important to be able to convert membership to arithmetic constraints and reciprocally. For that purpose we have two particular representations for sets.

Definition 3 *We say that a set is in a* canonical form *if it is either* $\emptyset$ *or the union of a finite sequence* $S_1, \ldots, S_n$ *of non-empty intervals and/or finite sets of* $\mathbb{R}$*, that are pairwise disjoint and such that* $\sup(S_i) \leq \inf(S_{i+1})$ *for all* $1 \leq i < n$ *and if* $\sup(S_i) = \inf(S_{i+1})$ *then* $\sup(S_i) \notin S_i$ *and* $\inf(S_{i+1}) \notin S_{i+1}$*. The infimum and supremum of each set may be* $-\infty$ *and* $+\infty$*. A* constraining set *is every subset of* $\mathbb{R}$ *that may be written in canonical form.*

Although *constraining sets* do not fully represent all subsets of $\mathbb{R}$, they cater for the most frequent types of sets that occur in math exercises, if the trigonometric functions are not involved. In the near future, we plan to extend them to deal with solution sets for constraints that involve also trigonometric functions. It is worthwhile noting that for the success of CAL applications we do not need cover all sorts of exercises that a teacher may invent and propose in a classroom. Neither this would be theoretically possible in general.

Example 1 *The set* $([-3, -1[\cup\{2, 17\} \cup [8, 11[\cup]11, 14[) \setminus \{10\}$ *is a constraining set and* $[-3, -1[\cup\{2\} \cup [8, 10[\cup]10, 11[\cup]11, 14[\cup\{17\}$ *is the same set presented in canonical form.*

This canonical form is like a picture of the set represented in the real axis. The following form, which we call *the reduced normal form* is the one we need to get a more compact arithmetic representation of each constraining set, which is relevant for CAL. The reduced normal form is unique.

Definition 4 *We say that a constraining set is* in reduced normal form *(*`rnf`*, for short) if it is either* $\mathbb{R}$*,* $\emptyset$*, a finite non-empty set,* $\cup_{i=1}^{n} S_i$*,* $(\cup_{i=1}^{n} S_i) \setminus S_{n+1}$*,* $\mathbb{R} \setminus S_{n+1}$*,* $((\cup_{i=1}^{n} S_i) \setminus S_{n+1}) \cup S_{n+2}$*, or* $(\cup_{i=1}^{n} S_i) \cup S_{n+2}$*, for a finite sequence of non-empty and non-universal intervals* $S_1, \ldots, S_n$ *with* $\sup(S_i) < \inf(S_{i+1})$*, for* $1 \leq i < n$ *and* S_{n+1}, S_{n+2} *non-empty disjoint finite sets such that* $S_{n+1} \subset \cup_{i=1}^{n} S_i$ *and* $S_{n+2} \cap (S_i \cup \{\inf(S_i), \sup(S_i)\}) = \emptyset$*, for every* $i \leq n$*.*

Example 2 *We may check that*

$$\texttt{rnf}([-3,-1[\cup \{2,17\} \cup [8,11[\cup]11,14[) \setminus \{10\})$$

is $(([-3,-1[\cup [8,14[) \setminus \{10,11\}) \cup \{2,17\}$*.*

To transform membership constraints to arithmetic constraints in a straightforward way, we shall express each set S given in reduced normal form in terms of $\mathcal{S}^k_{\lhd}$'s, for suitable k's and $\lhd$'s.

Definition 5 *We use* $\mathcal{S}^k_{\lhd}$ *to refer to the set* $\{x \in \mathbb{R} : x \lhd k\}$*, with* $\lhd \in \{=, \neq, >, <, \leq, \geq\}$ *and* $k \in \mathbb{R}$*.*

E.g., $\mathcal{S}^{-3}_{\geq}$ denotes $[-3, +\infty[$, and $\mathcal{S}^{5}_{<}$ and $\mathcal{S}^{2}_{\neq}$ denote $]-\infty, 5[$ and $\mathbb{R} \setminus \{2\}$.

The reduction function is denoted by τ_1 and defined as follows. We note that $\mathbb{R}$ and $\emptyset$ have a double meaning, identifying a set and a name we give to it.

Definition 6 *Let* τ_1 *be a map that rewrites sets given in* `rnf` *in terms of sets* $\mathcal{S}^k_{\lhd}$*'s and is inductively defined by:*

- $\tau_1(\mathbb{R}) = \mathbb{R}$*,* $\tau_1(\emptyset) = \emptyset$*,* $\tau_1(\{a_1, \ldots, a_n\}) = \cup_{i=1}^{n} \mathcal{S}^{a_i}_{=}$
- $\tau_1([a,b]) = \mathcal{S}^a_{\geq} \cap \mathcal{S}^b_{\leq}$*,* $\tau_1([a,b[) = \mathcal{S}^a_{\geq} \cap \mathcal{S}^b_{<}$*,* $\tau_1(]a,b]) = \mathcal{S}^a_{>} \cap \mathcal{S}^b_{\leq}$*,* $\tau_1(]a,b[) = \mathcal{S}^a_{>} \cap \mathcal{S}^b_{<}$*,* $\tau_1([a,+\infty[) = \mathcal{S}^a_{\geq}$*,* $\tau_1(]-\infty,a]) = \mathcal{S}^a_{\leq}$*,* $\tau_1(]a,+\infty[) = \mathcal{S}^a_{>}$*,* $\tau_1(]-\infty,a[) = \mathcal{S}^a_{<}$*, for* $a, b \in \mathbb{R}$*,*
- *and, finally,* $\tau_1(S \setminus \{a_1 \ldots, a_n\}) = \tau_1(S) \cap (\cap_{i=1}^{n} \mathcal{S}^{a_i}_{\neq})$ *and* $\tau_1(\cup_{i=1}^{n} A_i) = \cup_{i=1}^{n} \tau_1(A_i)$*.*

The conversion map τ_1 does a syntactic transformation of the set's presentation, that is quite convenient for the subsequent conversion of membership constraints into arithmetic constraints by τ_2.

Definition 7 *Let* τ_2 *be a map that rewrites membership constraints* $f(x) \in S$*, with* S *given in terms of* $\mathcal{S}^k_{\lhd}$*'s, into arithmetic constraints and that is inductively defined by:*

- $\tau_2(f(x) \in \mathbb{R}) = (f(x) \in \mathbb{R})$*,* $\tau_2(f(x) \in \emptyset) = (f(x) \in \emptyset)$*,* $\tau_2(f(x) \in \mathcal{S}^k_{\lhd}) = (f(x) \lhd k)$*,*

- $\tau_2(f(x) \in \cup_{i=1}^n S_i) = (\vee_{i=1}^n \tau_2(f(x) \in S_i))$ *and*
- $\tau_2(f(x) \in \cap_{i=1}^n S_i) = (\wedge_{i=1}^n \tau_2(f(x) \in S_i))$.

Moreover, for every given constraining set S (presented in any form) and function f (that may be id), we shall write $\Gamma(f(x) \in S)$ as an abbreviation of $\tau_2(f(x) \in \tau_1(\texttt{rnf}(S)))$.

Clearly, $\tau_2(f(x) \in \tau_1(\texttt{rnf}(S)))$ is an arithmetic constraint that is equivalent to $f(x) \in S$. It is worth noting that, we consider $x \in S$ simpler than $\Gamma(x \in S)$, so that, while specifying the rewriting rules, we shall distinguish in some situations the cases $f = id$ and $f \neq id$.

Example 3 *If $S = [-3, -1[\cup[8, 11[\cup]11, +\infty[)$, we may rewrite, for instance, $\Gamma(f(x) \in S)$ as follows*

$$\begin{aligned}
\Gamma(f(x) \in S) &= \tau_2(f(x) \in \tau_1(([-3, -1[\cup[8, +\infty[) \setminus \{11\})) = \\
&= \tau_2(f(x) \in ((\mathcal{S}_{\geq}^{-3} \cap \mathcal{S}_{<}^{-1}) \cup \mathcal{S}_{\geq}^{8}) \cap \mathcal{S}_{\neq}^{11}) = \\
&= \tau_2(f(x) \in (\mathcal{S}_{\geq}^{-3} \cap \mathcal{S}_{<}^{-1}) \cup \mathcal{S}_{\geq}^{8}) \wedge (f(x) \neq 11) = \\
&= ((f(x) \geq -3 \wedge f(x) < -1) \vee f(x) \geq 8) \wedge f(x) \neq 11
\end{aligned}$$

This translation is used in the solving procedure, if $f \neq id$. Let us imagine, for example, that f was $rad_3 \circ pol_{2,-7}$, i.e., $f(x) = \sqrt[3]{2x-7}$. To solve $\langle f(x) \in S, x, \mathbb{R}\rangle$, for S as above, students have to translate the membership constraint into an arithmetic one, so that we will make the solver do the same.

Proposition 1 *For all constraining sets S, the problem $\langle f(x) \in S, x, D\rangle$ is equivalent to $\langle \Gamma(f(x) \in S), x, D\rangle$.*

The inverse transformation is useful to simplify some arithmetic constraints. Indeed, to solve, for instance,

$$(f(x) > 8 \wedge f(x) \geq -3) \wedge f(x) < 2$$

clever students first inspect whether this constraint may be simplified. In this particular case, it is indeed inconsistent, because

$$(\mathcal{S}_{>}^{8} \cap \mathcal{S}_{\geq}^{-3}) \cap S_{<}^{2}$$

should be reduced to $\emptyset$. We shall not introduce this inverse map formally, but it will be implicit in some of the rewriting rules. Clearly, for example, $\wedge_{i=1}^n (f(x) \trianglelefteq_i k_i)$ gets into $f(x) \in \texttt{rnf}(\cap_{i=1}^n \mathcal{S}_{\trianglelefteq_i}^{k_i})$ and $\vee_{i=1}^n (f(x) \trianglelefteq_i k_i)$ into $f(x) \in \texttt{rnf}(\cup_{i=1}^n \mathcal{S}_{\trianglelefteq_i}^{k_i})$.

3.2 Implementation

Each of these reductions between different set representations was implemented in Prolog by a predicate. For the implementation we reused a module developed for DEMOMATH for operating constraining sets in canonical form. As we noted above, the canonical form is like a picture of the set represented in the real axis, so that, for their Prolog representation we use ordered lists that look like

```
[a(-infty),f(8),i(12),i(17),f(1000),a(1002), a(1002),a(infty)]
```

meaning $]-\infty, 8] \cup \{12, 17\} \cup [1000, 1002[\cup]1002, \infty[$. It stands for a union of intervals and of sets of isolated points, `a(X)` and `f(X)` mean *open* and *closed* at `X`, respectively, and `i(X)` says that `X` is an isolated point. All set operations with this type of sets are supported by the package. A set in `rnf` is represented in Prolog in a simple way, using the same form as above for denoting each finite set and each union of intervals (or single interval). The main union and set difference is translated by operators `cup` and `setminus`. For instance, for the Example 2, this yields

```
[f(-3),a(-1),f(8),a(14)] setminus [i(10),i(11)] cup [i(2),i(17)].
```

The emptyset is `[]` and $\mathbb{R}$ is translated by `[a(-infty),a(infty)]`. Some symbolic representations were introduced for $\mathcal{S}^k_{\leqslant}$, namely, `s(real)`, `s([])`, `s(K,eq)`, `s(K,lt)`, `s(K,leq)`, ..., and, finally, `cap` denotes the intersection.

Atomic constraints are represented by terms of the form `c(F,Op,SorK)` where `F` represents a function; `Op` is a Prolog atom `lt,eq,neq,leq,...,in,notin`, and `SorK` is a set in `rnf` representation for set constraints or a constant. For constants we use the arithmetic module define for DEMOMATH that does exact arithmetic. Terms of the form `and([C_1,...,C_n])` represent a conjunction of constraints $\wedge_{i=1}^n C_i$ and, analogously, terms of the form `or([C_1,...,C_n])` represent a disjuntion of constraints $\vee_{i=1}^n C_i$. Problems are represented by terms of the form `p(C,X,D)` where `C` is a (complex) constraint, `X` a variable and `D` is a set in `rnf`.

We shall now go on to describe how problems are solved.

4 Solving problems

For all the rewriting rules we shall describe, we assume that in every problem $\langle C(x), x, D\rangle$ all expressions that occur on the left-hand side of atomic constraints are defined on D or a superset of D. This means that one of the first steps in the solving procedure is the computation of the domain of $C(x)$, that is the set of values of x for which we may evaluate $C(x)$.

4.1 Computing the domains of the involved expressions

In the implementation of the predicate that computes the domain of a function, the information given in Table 1 is translated to clauses and consulted when required. Then, the domain is computed by actually solving some problems, so that the solving procedure and domain determination are not independent.

- $\mathcal{D}_{f\circ g} \equiv \texttt{sols}(P)$, for $P : \langle g(x) \in \mathcal{D}_f, x, \mathcal{D}_g\rangle$.
- $\mathcal{D}_{f\odot g} \equiv \texttt{sols}(P)$ for $P : \langle x \in \mathcal{D}_f \cap \mathcal{D}_g, x, \mathbb{R}\rangle$, for $\odot \in \{+, =, \times\}$.
- $\mathcal{D}_{f/g} \equiv \texttt{sols}(P)$ for $P : \langle x \in \mathcal{D}_f \cap \mathcal{D}_g \wedge g(x) \neq 0, x, \mathbb{R}\rangle$

Actually, these problems give rise to sequences of dependent problems, because this definition is recursive. For instance, before solving $P : \langle g(x) \in \mathcal{D}_f, x, \mathcal{D}_g\rangle$, the domains of f and g are computed.

Example 4 *For instance, to compute the domain of* $rad_2 \circ pol_{1,-3} \circ rad_2 \circ pol_{1,-1}$ *that is, of the expression* $\sqrt{\sqrt{x-1}-3}$*, the solver has to solve* P

$$P : \langle (pol_{1,-3} \circ rad_2 \circ pol_{1,-1})(x) \in \mathcal{D}_{rad_2}, x, \texttt{sols}(P_1) \rangle$$

with $P_1 : \langle (rad_2 \circ pol_{1,-1})(x) \in \mathcal{D}_{pol_{1,-3}}, x, \texttt{sols}(P_2) \rangle$ *and* P_2 *such that* $\texttt{sols}(P_2)$ *is* $\mathcal{D}_{rad_2 \circ pol_{1,-1}}$*, and so forth.*

Although for the solver we give priority to right association for function composition, $f_1 \circ f_2 \circ \cdots \circ f_n \equiv f_1 \circ (f_2 \circ (\cdots \circ f_n))$, we may also explore in some cases other forms of unification for $\circ$. That is, we may also have

$$\mathcal{D}_{rad_2 \circ pol_{1,-3} \circ rad_2 \circ pol_{1,-1}} = \mathcal{D}_{(rad_2 \circ pol_{1,-3}) \circ (rad_2 \circ pol_{1,-1})}$$

although in our opinion this latter association is less intuitive for CAL. Nevertheless, to be able to have more flexible representations not only for composition, but also for $+$, $-$, $\times$ and $/$, we will need to go beyond syntactic unification somehow, and support other forms of matching and unification [11]. Again, in this setting, our goal is not to guarantee that the system may find common unifiers, but rather that it may do it, firstly for a quite controlled subset of expressions and secondly with not many manipulations. The reason is that, to be relevant for CAL applications, the depth of such transformations is somehow small.

4.2 Canonical forms for complex constraints

It is important for CAL that the solving steps remain close to what students would do, to be interesting to show and explain solutions step-by-step. For this reason, we cannot manipulate problems and constraints in an arbitrary way. Hence, for example, we need a solver that can work on both disjunctive and conjunctive constraints. Nevertheless, we consider that constraints may be put into a canonical form. The idea is to capture at a surface level constraints that shall be gathered and treated together to get a resolution that is better from a pedagogical point of view.

Definition 8 *A constraint is in* a canonical form *if it is either atomic or a complex constraint in conjunctive canonical form (CCF) or in disjunctive canonical form (DCF). The constraint* $\wedge_{i=1}^{n} C_i$ *(resp.,* $\vee_{i=1}^{n} C_i$*) where all* C_i*'s are atomic constraints and* $n \geq 1$*, is called a* simple constraint *in CCF (resp,. in DCF). In general,* $\wedge_{i=1}^{n} C_i$ *is* in CCF *if it is a simple CCF constraint or all the* C_i*'s are non-atomic and in DCF, except possibly* C_1 *that may be also a simple CCF constraint. Similarly,* $\vee_{i=1}^{n} C_i$ *is* in DCF *if it is a simple DCF constraint or all the* C_i*'s are are non-atomic and in CCF, except possibly* C_1 *that may be also a simple DCF constraint.*

For the sake of some generality, we do not introduce any order for the atomic constraints. However, for CAL, it is convenient that, for instance, in every conjunct and disjunct of atomic constraints the membership and the arithmetic constraints that have the same left-hand side $f(x)$ are gathered, for each f. Nevertheless, in the implementation, this is to be dealt with selectors and strategies for the application of rewriting rules.

We shall describe the solver as a rewriting system [11], with $\Rightarrow$ denoting the (one-step) rewriting relation and $\Rightarrow^+$ its transitive irreflexive closure (one or more rewriting steps). Each rule is represented by $\dfrac{P_1 \quad \cdots \quad P_n}{P}$ *Cond* meaning that from $P_1 \quad \cdots \quad P_n$ we may obtain P under condition *Cond*.

4.3 Rewriting rules

4.3.1 Rewriting $\wedge$- and $\vee$-constraints

Rule 1 *(*ConjunctiveDisjunctive*) If $j \in \{1,\ldots,n\}$ and $\emptyset \neq I \subseteq \{1,\ldots,n\}$ then*

$$\frac{\langle \wedge_{i=1}^{n} C_i, x, D\rangle}{\langle \wedge_{i=1,i\neq j}^{n} C_i, x, \mathtt{rnf}(D \cap S)\rangle} \text{ if } C_j = (x \in S)$$

$$\frac{\langle \vee_{i=1}^{n} C_i, x, D\rangle}{\langle x \in D, x, D\rangle} \text{ if } C_j = (x \in D)$$

$$\frac{\langle \vee_{i=1}^{n} C_i, x, D\rangle}{\langle \vee_{i=1,i\neq j}^{n} C_i, x, D\rangle} \text{ if } C_j = (x \in \emptyset)$$

$$\frac{\langle \wedge_{i=1}^{n} C_i, x, D\rangle \quad \forall i \in I\ \langle C_i, x, D\rangle \Rightarrow^{+} \langle C_i', x, D_i\rangle}{\langle \wedge_{i\in I} C_i' \wedge (\wedge_{i\in\{1,\ldots,n\}\setminus I} C_i), x, D\rangle}$$

$$\frac{\langle \vee_{i=1}^{n} C_i, x, D\rangle \quad \forall i \in I\ \langle C_i, x, D\rangle \Rightarrow^{+} \langle C_i', x, D_i\rangle}{\langle \vee_{i\in I} C_i' \vee (\vee_{i\in\{1,\ldots,n\}\setminus I} C_i), x, D\rangle}$$

$$\frac{\langle \vee_{i=1}^{n} C_i, x, D\rangle \quad \langle \vee_{i\in I} C_i, x, D\rangle \Rightarrow^{+} \langle C', x, D'\rangle}{\langle C' \vee (\vee_{i\in\{1,\ldots,n\}\setminus I} C_i), x, D\rangle}$$

$$\frac{\langle \wedge_{i=1}^{n} C_i, x, D\rangle \quad \langle \wedge_{i\in I} C_i, x, D\rangle \Rightarrow^{+} \langle C', x, D'\rangle}{\langle C' \wedge (\wedge_{i\in\{1,\ldots,n\}\setminus I} C_i), x, D\rangle}$$

In the conjunctive case, we could have considered

$$\langle \wedge_{i\in I} C_i' \wedge (\wedge_{i\in\{1,\ldots,n\}\setminus I} C_i), x, \mathtt{rnf}(D \cap (\cap_{i\in I} D_i))\rangle$$

and

$$\langle C' \wedge (\wedge_{i\in\{1,\ldots,n\}\setminus I} C_i), x, \mathtt{rnf}(D \cap D')\rangle$$

instead. We decided for the simpler form since if $D_i \subset D$ for some i, then C_i' is $(x \in D_i)$. Hence, D gets simplified in a subsequent rewriting step. The same applies to D' and C'.

4.3.2 Normalization step

Rule 2 *(*AggregateNormalize*) Let I_1 and I_2 be finite subsets of $\mathbb{N}$ such that either $|I_1 \cup I_2| \geq 2$ or $I_2 \neq \emptyset$.*

$$\frac{\langle (\wedge_{i\in I_1} f(x) \trianglelefteq_i k_i) \wedge (\wedge_{i\in I_2} f(x) \in S_i), x, D\rangle}{\langle \Gamma(f(x) \in (\cap_{i\in I_1} \mathcal{S}^{k_i}_{\trianglelefteq_i}) \cap (\cap_{i\in I_2} S_i)), x, D\rangle} \text{ if } f \neq id$$

$$\frac{\langle (\wedge_{i\in I_1} x \trianglelefteq_i k_i) \wedge (\wedge_{i\in I_2} x \in S_i), x, D\rangle}{\langle x \in \mathtt{rnf}((\cap_{i\in I_1} \mathcal{S}^{k_i}_{\trianglelefteq_i}) \cap (\cap_{i\in I_2} S_i)), x, D\rangle}$$

$$\frac{\langle (\vee_{i\in I_1} f(x) \trianglelefteq_i k_i) \vee (\vee_{i\in I_2} f(x) \in S_i), x, D\rangle}{\langle \Gamma(f(x) \in (\cup_{i\in I_1} \mathcal{S}^{k_i}_{\trianglelefteq_i}) \cup (\cup_{i\in I_2} S_i)), x, D\rangle} \; \textit{if } f \neq id$$

$$\frac{\langle (\vee_{i\in I_1} x \trianglelefteq_i k_i) \vee (\vee_{i\in I_2} x \in S_i), x, D\rangle}{\langle x \in \mathtt{rnf}((\cup_{i\in I_1} \mathcal{S}^{k_i}_{\trianglelefteq_i}) \cup (\cup_{i\in I_2} S_i)), x, D\rangle}$$

AggregateNormalize) cannot be applied consecutively, for the solver termination. This rule aims to simplify several atomic constraints involving the same $f(x)$ and it is useful to detect inconsistency or validity.

4.3.3 Simple Cases

In some cases, the solver may trivially reach a solved form.

Rule 3 *(*TrivialSolve*)*

$$\frac{\langle C, x, \emptyset\rangle}{\langle x \in \emptyset, x, \emptyset\rangle} \; \textit{if } C \neq (x \in \emptyset)$$

$$\frac{\langle x \in S, x, D\rangle}{\langle x \in \mathtt{rnf}(D \cap S), x, \mathtt{rnf}(D \cap S)\rangle} \; \textit{if } S \neq D$$

$$\frac{\langle x \notin S, x, D\rangle}{\langle x \in \mathtt{rnf}(D \setminus S), x, \mathtt{rnf}(D \setminus S)\rangle}$$

$$\frac{\langle x \trianglelefteq k, x, D\rangle}{\langle x \in \mathtt{rnf}(D \cap \mathcal{S}^{k}_{\trianglelefteq}), x, \mathtt{rnf}(D \cap \mathcal{S}^{k}_{\trianglelefteq})\rangle}$$

$$\frac{\langle c_k(x) \trianglelefteq k', x, D\rangle}{\langle x \in D, x, D\rangle} \; \textit{if } k \trianglelefteq k'$$

$$\frac{\langle c_k(x) \trianglelefteq k', x, D\rangle}{\langle x \in \emptyset, x, \emptyset\rangle} \; \textit{if } k \not\trianglelefteq k'$$

$$\frac{\langle f(x) \in \emptyset, x, D\rangle}{\langle x \in \emptyset, x, \emptyset\rangle} \; \textit{if } f \neq id$$

4.3.4 Shortcuts for solving

In some cases, we may decide that the constraint is valid or inconsistent by performing an almost straightforward analysis of the problem. Proposition 2 characterizes some of them.

Proposition 2 *If $k \notin f(\mathcal{D}_f)$ then $\langle (f \circ g)(x) \trianglelefteq k, x, D\rangle$ is either valid (i.e., it is equivalent to $\langle x \in D, x, D\rangle$) or inconsistent (i.e., it is equivalent to $\langle x \in \emptyset, x, \emptyset\rangle$), for all $f \in \{pow_{2n}, rad_{2n}, abs, exp_a, pol_{a,b,c}\}$.*

In general, we may also try to use some knowledge about the image set to help decide whether a constraint is trivially valid or inconsistent. This is the idea behind Rule 4, which is quite relevant in math education.

Rule 4 (ValidInconsistent) *Given generic functions f and g, with $f, g \neq id$, let $\mathcal{E}$ be such that $f(\mathcal{D}_f) \subseteq \mathcal{E}$. Then,*

$$\frac{\langle (f \circ g)(x) \in S, x, D\rangle}{\langle x \in D, x, D\rangle} \text{ if } \mathcal{E} \subseteq S$$

$$\frac{\langle (f \circ g)(x) \in S, x, D\rangle}{\langle x \in \emptyset, x, \emptyset\rangle} \text{ if } S \cap \mathcal{E} = \emptyset$$

$$\frac{\langle (f \circ g)(x) \trianglelefteq k, x, D\rangle}{\langle x \in D, x, D\rangle} \text{ if } \mathcal{E} \subseteq \mathcal{S}^k_{\trianglelefteq}$$

$$\frac{\langle (f \circ g)(x) \trianglelefteq k, x, D\rangle}{\langle x \in \emptyset, x, \emptyset\rangle} \text{ if } \mathcal{S}^k_{\trianglelefteq} \cap \mathcal{E} = \emptyset$$

If $g = id$, the same rewriting rules still apply, but it is enough that $f(D) \subseteq \mathcal{E}$. In general, $\mathcal{E}$ may be any set that contains the image of $f|_{g(D)}$.

Rule 4 becomes particularly interesting if either the image of f (or of $f|_{g(D)}$) is known or we may "easily" deduce a (non-trivial) superset $\mathcal{E}$ of it. Several constructive properties may be applied for such purpose and the exact meaning of "easily" depends both on which of them are used in inference and on how deep we let the inference go. For educational purposes, it may often not go beyond one or two deduction steps, although this may depend on the goal of the exercise.

Example 5 *By employing Rule 4 and considering $\mathcal{E} = abs(\mathbb{R}) = [0, +\infty[$, the solver straightforwardly obtains that the problem*

$$\langle (abs \circ rad_3 \circ pol_{-1,2})(x) \leq -4, x, \mathbb{R}\rangle$$

has no solution, i.e., the constraint $|\sqrt[3]{-x+2}| \leq -4$ is inconsistent. So, it is rewritten as $\langle x \in \emptyset, x, \emptyset\rangle$. By the same rule, it deduces that

$$\langle (abs \circ rad_3 \circ pol_{-1,2})(x) \geq 0, x, \mathbb{R}\rangle$$

can be rewritten in solved form $\langle x \in \mathbb{R}, x, \mathbb{R}\rangle$.

For a basic implementation of the solver, we may confine the application of this rule to the cases when the required information is explicitly given in Table 1. However, some math problems are often solved just with a glance, and, clearly, a brief reasoning. To translate this, and following a strategy alike the one used by some constraint programming solvers for propagation, we may implement the solver to perform some inference. Table 2 contains some properties that may be used as inference rules to obtain $\mathcal{E}$, for a given complex function f. The usual extension of the binary operations and relations over real numbers to $\mathbb{R} \cup \{-\infty, +\infty\}$ is considered.

Some other useful properties state well-known results from Calculus, that students usually learn in math, and are likely more effective for finding smaller supersets $\mathcal{E}$. For example:

- if $\mathcal{D}_f = \mathbb{R}$ and f is invertible, $f(\mathcal{D}_f) = \mathcal{D}_{f^{-1}}$. This requires that the expression of f^{-1} can be computed, which is not always possible. For that purpose, it is useful also to use the identity $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$.

$\inf(f \pm c_k) = \inf(f) \pm k$ and $\sup(f \pm c_k) = \sup(f) \pm k$
$\inf(-f) = \sup(f)$ and $\sup(-f) = \inf(f)$
$\inf(c_k \times f) = \min(k \times \inf(f), k \times \sup(f))$ and $\sup(c_k \times f) = \max(k \times \inf(f), k \times \sup(f))$
$\inf(f/c_k) = \inf((1/c_k) \times f)$ and $\sup(f/c_k) = \sup((1/c_k) \times f)$
$\inf(f + g) \geq \inf(f) + \inf(g)$ and $\sup(f + g) \leq \sup(f) + \sup(g)$
$\inf(f - g) \geq \inf(f) - \sup(g)$ and $\sup(f - g) \leq \sup(f) - \inf(g)$
$\inf(f \times g) \geq \min\{\inf(f) \times \inf(g), \inf(f) \times \sup(g), \sup(f) \times \inf(g), \sup(f) \times \sup(g)\}$
$\sup(f \times g) \leq \max\{\inf(f) \times \inf(g), \inf(f) \times \sup(g), \sup(f) \times \inf(g), \sup(f) \times \sup(g)\}$
$\inf(1/g) \geq 1/\sup(g)$ and $\sup(1/g) \leq 1/\inf(g)$, if $g > 0$ or $g < 0$

Table 2: Estimating bounds of a function's range: if the minimum and/or maximum values are defined for f and g, properties can be rephrased in terms of min and max.

- Provided f is continuous, Bolzano theorem gives a way to infer $\mathcal{E}$ for $f|_{[a,b]}$. Indeed, if f is continuous and monotonic on $[a, b]$ then $[\min(f(a), f(b)), \max(f(a), f(b))]$ is the image of $f|_{[a,b]}$. A similar result holds for any generic interval I if we take $\lim_{x \to c, x \in I} f(x)$, for each extreme point c of I instead.
- Another important feature is that the image of $f \circ g$ is the image of $f|_{g(\mathcal{D}_g)}$. Therefore, if $\mathcal{D}_f \subseteq g(\mathcal{D}_g)$, the image of $f \circ g$ is the image of f. So, if $g(\mathcal{D}_g) = \mathbb{R}$, the image of $f \circ g$ is $f(\mathcal{D}_f)$.

The following is also a simplification rule and the solver needs it to keep the solving procedure close to what students usually do.

Rule 5 (ReducesToDiscreteSet) *If $\trianglelefteq \notin \{=, \neq\}$ and $\mathcal{E}$ is such that $f(\mathcal{D}_f) \subseteq \mathcal{E}$ then when $\mathcal{S}^k_{\trianglelefteq} \cap \mathcal{E}$ is a finite set*

$$\frac{\langle (f \circ g)(x) \trianglelefteq k, x, D \rangle}{\langle \Gamma((f \circ g)(x) \in \mathcal{S}^k_{\trianglelefteq} \cap \mathcal{E}), x, D \rangle}$$

and when $\mathcal{S}^k_{\trianglelefteq} \cap \mathcal{E} = \mathcal{E} \setminus S$ for a finite set S

$$\frac{\langle (f \circ g)(x) \trianglelefteq k, x, D \rangle}{\langle \Gamma((f \circ g)(x) \notin S), x, D \rangle}$$

Actually, $\mathcal{E}$ may be any set that contains the image of $f|_{g(D)}$ that is $f(g(D))$.

Example 6 *By Rule 5, the solver deduces that* $\langle (abs \circ rad_3 \circ pol_{-1,2})(x) \leq 0, x, \mathbb{R}\rangle$ *is equivalent to* $\langle (abs \circ rad_3 \circ pol_{-1,2})(x) = 0, x, \mathbb{R}\rangle$ *i.e., that* $|\sqrt[3]{-x+2}| \leq 0$ *can be rewritten as* $|\sqrt[3]{-x+2}| = 0$. *Again,* $\mathcal{E} = abs(\mathbb{R}) = [0, +\infty[$.

We shall now analyse the cases where the solver has to perform some more elaborate algebraic manipulations. If we do not explicitly state the contrary, g may be *id*.

4.3.5 Strictly monotonic functions

Rule 6 applies to all functions f that are strictly monotonic, which is the case of $pol_{a,b}$, p_k, pow_{2n+1}, rad_{2n+1}, log_a and exp_a.

Rule 6 *(*StrictMonotonic*) If* f *is a strictly monotonic and* $k \in f(\mathcal{D}_f)$ *then*

$$\frac{\langle (f \circ g)(x) \trianglelefteq k, x, D\rangle}{\langle g(x) \trianglelefteq f^{-1}(k), x, D\rangle} \text{ if } f \text{ is strictly increasing}$$

$$\frac{\langle (f \circ g)(x) \trianglelefteq k, x, D\rangle}{\langle g(x) \trianglelefteq^{-1} f^{-1}(k), x, D\rangle} \text{ if } f \text{ is strictly decreasing}$$

where g *may be also the identity function id.*

It is worth noting that the condition $k \in f(\mathcal{D}_f)$ imposes no restriction if

$$f \in \{pol_{a,b}, p_k, pow_{2n+1}, rad_{2n+1}, log_a\},$$

because $f(\mathcal{D}_f) = \mathbb{R}$. Moreover, to apply Rule 6, the expression that defines f^{-1} has to be computable. This is the case of all the functions just mentioned and also of exp_a.

Example 7 *By Rule 6, the solver rewrites* $\langle (rad_3 \circ pol_{-1,2})(x) \leq 2, x, \mathbb{R}\rangle$ *to*

$$\langle pol_{-1,2}(x) \leq 8, x, \mathbb{R}\rangle$$

and, again, by Rule 6, it obtains

$$\langle x \geq -6, x, \mathbb{R}\rangle$$

i.e., $\sqrt[3]{-x+2} \leq 2 \Leftrightarrow -x+2 \leq 8 \Leftrightarrow x \geq -6$. *By Rule 3, it finally deduces the solved form*

$$\langle x \in [-6, +\infty[, x, [-6, +\infty[\rangle.$$

4.3.6 Functions symmetric w.r.t. a vertical line

Another relevant class of functions consists of the ones that present some symmetry w.r.t. to a vertical line and have a strictly monotonic branch.

The graph of f is symmetric w.r.t. $x = a$ iff $f(a + x) = f(a - x)$, for all x such that $a + x \in \mathcal{D}_f$. This is equivalent to saying that $f(x) = f(2a - x)$, for all $x \in \mathcal{D}_f$. It requires that the domain of f be symmetric w.r.t. $x = a$, that is, that $x \in \mathcal{D}_f$ iff $2a - x \in \mathcal{D}_f$, for all x.

Notation. Let us denote $\mathcal{D}_f \cap \mathcal{S}^a_{\geq}$ by $\mathcal{D}_f^{\geq a}$ and define $\mathcal{B}^{\delta}_{a,\trianglelefteqslant}$ by $\{x \in \mathbb{R} \ : \ |x-a| \trianglelefteqslant \delta\}$.
E.g., $\mathcal{B}^{-3}_{0,\geq} = \mathbb{R}$, $\mathcal{B}^{5}_{0,<} =]-5,5[$, $\mathcal{B}^{2}_{-1,=} = \{-3,1\}$, $\mathcal{B}^{0}_{1,<} = \emptyset$.

Rule 7 (AxialSymMonotonicBranch) *For f symmetric w.r.t. $x=a$ and strictly monotonic on $\mathcal{D}_f^{\geq a}$, and $k \in f(\mathcal{D}_f)$, let $k' = (f|_{\mathcal{D}_f^{\geq a}})^{-1}(k) - a$. Then,*

$$\frac{\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle}{\langle \Gamma(g(x) \in \mathcal{B}^{k'}_{a,\trianglelefteqslant}), x, D\rangle} \text{ if } f|_{\mathcal{D}_f^{\geq a}} \text{ increases}$$

$$\frac{\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle}{\langle \Gamma(g(x) \in \mathcal{B}^{k'}_{a,\trianglelefteqslant^{-1}}), x, D\rangle} \text{ if } f|_{\mathcal{D}_f^{\geq a}} \text{ decreases}$$

Indeed, when the graph of f is symmetric w.r.t. $x=a$ and f is strictly monotonic in $\mathcal{D}_f^{\geq a}$, the restriction of f to $\mathcal{D}_f^{\geq a}$, that we represent by $f|_{\mathcal{D}_f^{\geq a}}$, is a one-to-one function and thus it is invertible. Although the same holds for f restricted to $\mathcal{D}_f \cap \mathcal{S}^a_{\leq}$, this branch is not so useful for our purpose.

Example 8 *By Rule 7, the solver rewrites $\langle (pow_2 \circ pol_{-1,2})(x) < 4, x, \mathbb{R}\rangle$ to*

$$\langle pol_{-1,2}(x) > -2 \wedge pol_{-1,2}(x) < 2, x, \mathbb{R}\rangle$$

since $(pow_2|_{\mathbb{R}_0^+})^{-1} = rad_2$, $pow_2|_{\mathbb{R}_0^+}$ is monotonic increasing and $\mathcal{B}^{\sqrt{4}}_{0,<} =]-2,2[$.

4.3.7 Some rule instances

If f is even (i.e., $f(x) = f(-x)$, for all $x \in \mathbb{R}$) then $\mathcal{D}_f^{\geq 0}$ is $\mathbb{R}_0^+$ and if, in addition, $f|_{\mathbb{R}_0^+}$ is strictly monotonic then Rule 7 may be applied. This is the case of *abs* and pow_{2n}, for $n \geq 1$. It is interesting also to see that, from Table 1 and Rules 6 and 7, we may automatically infer that, for instance,

$$\frac{\langle (pol_{a,b} \circ f)(x) \trianglelefteqslant k, x, D\rangle}{\langle f(x) \trianglelefteqslant (k-b)/a, x, D\rangle} \text{ if } a > 0.$$

$$\frac{\langle (pol_{a,b} \circ f)(x) \trianglelefteqslant k, x, D\rangle}{\langle f(x) \trianglelefteqslant^{-1} (k-b)/a, x, D\rangle} \text{ if } a < 0.$$

$$\frac{\langle (abs \circ f)(x) \trianglelefteqslant k, x, D\rangle}{\langle \Gamma(f(x) \in \mathcal{B}^{k}_{0,\trianglelefteqslant}), x, D\rangle} \text{ if } k \geq 0.$$

$$\frac{\langle (pow_n \circ f)(x) \trianglelefteqslant k, x, D\rangle}{\langle f(x) \trianglelefteqslant \sqrt[n]{k}, x, D\rangle} \text{ if } n \text{ is odd.}$$

$$\frac{\langle (pow_n \circ f)(x) \trianglelefteqslant k, x, D\rangle}{\langle \Gamma(f(x) \in \mathcal{B}^{\sqrt[n]{k}}_{0,\trianglelefteqslant}), x, D\rangle} \text{ if } n \text{ is even and } k \geq 0.$$

so that, Rules 6 and 7 are actually quite convenient rule schemes.

The quadratic function. For the quadratic function $pol_{a,b,c} : x \mapsto ax^2 + bx + c$, that is symmetric w.r.t. $x = -\frac{b}{2a}$ and strictly monotonic on $\mathcal{D}_{pol_{a,b,c}}^{\geq -b/(2a)} = [-\frac{b}{2a}, +\infty[$, we have $pol_{a,b,c}|_{[-\frac{b}{2a},+\infty[}$ strictly increasing (resp., decreasing) function iff $a > 0$ (resp. $a < 0$). Students learn the formula giving the roots of a quadratic equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

and that the roots are real numbers iff $\Delta = b^2 - 4ac \geq 0$. They learn how to solve constraints of form $pol_{a,b,c}(x) \trianglelefteqslant 0$ through the analysis of the function's graph, its zeros and $sign(a)$. We may deduce the same laws they learn either directly from Rule 7 or using both Rules 6 and 7. For the latter, we must observe that

$$pol_{a,b,c} = pol_{a,-\Delta/(4a)} \circ pow_2 \circ pol_{1,b/(2a)}.$$

We have also implemented a specific rule for QUADRATIC. Indeed, we have

$$ax^2 + bx + c = a\left(x + \frac{b}{2a}\right)^2 - \frac{b^2 - 4ac}{4a} = a\left(x + \frac{b}{2a}\right)^2 - \frac{\Delta}{4a}.$$

We also see that $pol_{a,b,c}(\mathbb{R}) = [-\frac{\Delta}{4a}, +\infty[$ if $a > 0$ and $pol_{a,b,c}(\mathbb{R}) =]-\infty, -\frac{\Delta}{4a}]$ if $a < 0$. From

$$pol_{a,b,c}^{-1} = pol_{1,b/(2a)}^{-1} \circ pow_2^{-1} \circ pol_{a,-\Delta/(4a)}^{-1}$$

we have

$$(pol_{a,b,c}|_{\mathbb{R}_0^+})^{-1}(x) = -\frac{b}{2a} + \frac{\sqrt{ax + \Delta}}{|2a|}$$

which is needed in Rule 7. Nevertheless, to explain solutions step-by-step, it may make some sense to introduce the following reduction step

$$\frac{\langle (pol_{a,b,c} \circ g)(x) \trianglelefteqslant k, x, D\rangle}{\langle (pol_{a,b,c-k} \circ g)(x) \trianglelefteqslant 0, x, D\rangle} \text{ if } k \neq 0.$$

that students often perform and to write explicitly a rule for tackling $(pol_{a,b,c} \circ g)(x) \trianglelefteqslant 0$, instead of using Rule 7.

Rule 8 *(*QUADRATIC*) Let* $\Delta = b^2 - 4ac$ *and* $k' = \sqrt{\Delta}/|2a|$.

$$\frac{\langle (pol_{a,b,c} \circ g)(x) \trianglelefteqslant k, x, D\rangle}{\langle (pol_{a,b,c-k} \circ g)(x) \trianglelefteqslant 0, x, D\rangle} \textit{ if } k \neq 0.$$

$$\frac{\langle (pol_{a,b,c} \circ g)(x) \trianglelefteqslant 0, x, D\rangle}{\langle (\Gamma(g(x) \in B^{k'}_{-b/(2a),\trianglelefteqslant}), x, D\rangle} \textit{ if } \Delta \geq 0 \textit{ and } a > 0.$$

$$\frac{\langle (pol_{a,b,c} \circ g)(x) \trianglelefteqslant 0, x, D\rangle}{\langle (\Gamma(g(x) \in B^{k'}_{-b/(2a),\trianglelefteqslant^{-1}}), x, D\rangle} \textit{ if } \Delta \geq 0 \textit{ and } a < 0.$$

If $\Delta < 0$, the solver may apply Rules 4 and 5, with $\mathcal{E} = pol_{a,b,c}(\mathbb{R}) = [-\frac{\Delta}{4a}, +\infty[$ if $a > 0$ and $\mathcal{E} = pol_{a,b,c}(\mathbb{R}) =]-\infty, -\frac{\Delta}{4a}]$ if $a < 0$. In this case, students often use a less restrictive set $\mathcal{E}$, considering that $pol_{a,b,c}(\mathbb{R}) \subseteq \mathbb{R}^+ = \mathcal{E}$ if $a > 0$ and $pol_{a,b,c}(\mathbb{R}) \subseteq \mathbb{R}^- = \mathcal{E}$ if $a < 0$.

Example 9 *The problem* $\langle (pol_{1,2,1} \circ pow_3)(x) \leq 0, x, \mathbb{R} \rangle$ *becomes*

$$\langle \Gamma(pow_3(x) \in \{-1\}), x, \mathbb{R} \rangle$$

that is $\langle pow_3(x) = -1, x, \mathbb{R} \rangle$*, since* $\Delta = 0$*,* $\mathcal{B}^0_{-1,\leq} = \{-1\}$*. Then, by Rule 6, the problem is rewritten to* $\langle x = -1, x, \mathbb{R} \rangle$*.*

4.3.8 Piecewise functions

A piecewise function f is of form (f_i, D_i), for $i = 1, \ldots, n$, with $n \geq 2$ and is studied for $x \in D_1$

$$f(x) = \begin{cases} f_1(x) & \text{for } x \in D_1 \\ \vdots & \\ f_n(x) & \text{for } x \in D_n \end{cases}$$

for $n \geq 2$. Piecewise functions are studied in high-school math courses, arising in a natural way when the *abs* function is involved in the problems. Usually, the f_i's are no longer piecewise functions.

Rule 9 *(*Piecewise*) If* f *is a piecewise and* (f_i, D_i)*, for* $i = 1, \ldots, n$ *are its branches, then*

$$\frac{\langle (f \circ g)(x) \trianglelefteq \beta, x, D \rangle}{\langle \vee_{i=1}^{n} ((f_i \circ g)(x) \trianglelefteq \beta \wedge g(x) \in D_i), x, D \rangle} \text{ if } \beta \in \mathbb{R} \text{ or } \beta \subseteq \mathbb{R}.$$

Example 10 *The problem* $\langle f(x) \leq 7, x, \mathbb{R} \rangle$ *for*

$$f = (\, (rad_2 \circ pol_{1,-3}, [-3, +\infty[),\ (pol_{1,-6,9},]-\infty, -3[)\,)$$

is rewritten as

$$\langle\ ((rad_2 \circ pol_{1,-3})(x) \leq 7 \wedge x \in [-3, +\infty[)\ \vee\ \ (pol_{1,-6,9}(x) \leq 7 \wedge x \in]-\infty, -3[),\ x,\ \mathbb{R}\ \ \rangle$$

4.3.9 The most complex cases

Constraints that involve $f \odot g$, for $\odot \in \{+, -, \times, /\}$ may be hard to solve, and, indeed, they often render the problem undecidable. Hence, if we want to guarantee that there exists a solving procedure for the problem, we have to restrict the type of expressions that may arise in constraints. For the application we have in mind, this is not dramatic at all. On one hand, what we would like is that the system may help students understand a topic. On the other hand, the constraints students are asked to solve by hand are usually not too complex and, hopefully, decidable (if they apply the concepts and results they should master). We confine to the expressions described by the grammar given in [14]. The grammar given in Table 3 is a reformulation of that one and includes some minor extensions.

$function$	$\rightarrow$	$prodexpr \mid divexpr$;
$divexpr$	$\rightarrow$	$prodexpr/prodexpr \mid c_k/prodexpr \mid primop \circ divexpr$;
$prodexpr$	$\rightarrow$	$factor \mid factor \times prodexpr \mid primop \circ prodexpr$;
$factor$	$\rightarrow$	$sumexpr \mid basic \mid vxip$;
$sumexpr$	$\rightarrow$	$pk \circ sumexpr \mid abs \circ sumexpr \mid rad_n \circ sumexpr \mid pow_n \circ sumexpr \mid bsum$;
$basic$	$\rightarrow$	$pol_{a,b} \mid pol_{a,b,c} \mid pol_{a,b,c} \circ pow_n \mid pol_{a,b} \circ basic \mid primop \circ basic$;
$vxip$	$\rightarrow$	$pk \circ vxip \mid abs \circ vxip \mid rad_n \circ vxip \mid pow_n \circ vxip \mid$ $expand(pow_n \times (pol_{a,b,c} \circ pow_m)) \mid expand(pow_n \times (pol_{a,b} \circ pow_m))$;
$bsum$	$\rightarrow$	$pol_{a,b} \circ vquot_{120} \mid spbasic \mid specialbsum \mid$ $quot_{120} + spbasic$ where $dgden(quot_{120}) + deg(spbasic) \leq 2 \mid$ $sumquot_{120}$;
$vquot_{120}$	$\rightarrow$	$pow_n \circ vquot_{120} \mid rad_n \circ vquot_{120} \mid abs \circ vquot_{120} \mid quot_{120} \mid pol_{a,b} \circ vquot_{120}$
$quot_{120}$	$\rightarrow$	$k/spbasic \mid spbasic/k \mid spbasic/spbasic$ $abs \circ quot_{120} \mid pol_{a,b} \circ quot_{120}$;
$spbasic$	$\rightarrow$	$pbasic + pbasic \mid pbasic + spbasic$;
$pbasic$	$\rightarrow$	$basic_{\leq 2} \mid sbasic_{=1} \times sbasic_{=1}$;
$specialbsum$	$\rightarrow$	$krad_n \circ spbasic + krad_n \circ spbasic \mid$ $kpow_n \circ spbasic + kpow_n \circ spbasic \mid$ $kpow_n \circ spbasic + kpow_{2n} \circ spbasic_{=1} \mid$ $krad_{2n} \circ spbasic + krad_n \circ spbasic_{=1} \mid$ $krad_2 \circ spbasic + spbasic_{=1} \mid$ $kpow_2 \circ spbasic_{=1} + spbasic$;
$sumquot_{120}$	$\rightarrow$	$quot_{120} \mid$ $quot_{120} + sumquot_{120}$ if for each subterm $f + g, dgden(f) + dgnum(g) \leq 2$
$basic$	$\rightarrow$	$abs \mid pow_n \mid rad_n \mid pol_{a,b} \mid pol_{a,b,c} \mid abs \circ basic \mid pow_n \circ basic \mid rad_n \circ basic \mid$ $pol_{a,b} \circ basic \mid pol_{a,b,c} \circ pow_n$;
$krad_n$	$\rightarrow$	$rad_n \mid pk \circ rad_n$;
$kpow_n$	$\rightarrow$	$pow_n \mid pk \circ pow_n$;
$primop$	$\rightarrow$	$abs \mid pow_n \mid rad_n \mid pk$;

Table 3: A grammar that describes some functions whose zeros and sign variation may be computed automatically. Here, function pk is the expansion of $c_k \times id$ that is, $pk : x \mapsto k\,x$.

The main idea underlying it was to characterize expressions for which students may compute domains and zeros provided they can solve the following problems:

$aX + b = 0$
$aX^2 + bX + c = 0$
$aX^n + b = 0$
$a\sqrt[n]{X} + b = 0$
$X^n \pm Y^n = 0$, for $n \geq 2$ and $deg(X), deg(Y) \leq 2$,
$\sqrt[n]{X} \pm \sqrt[n]{Y} = 0$, for $n \geq 2$ and $deg(X), deg(Y) \leq 2$,
$X/Y \pm Z/T = 0$, with $deg(XT) \leq 2$ and $deg(YZ) \leq 2$.

The expressions of a given *degree* evaluate to polynomials of that degree when simplified or expanded to get rid of *abs* and pow_n, and shall not contain quotients and radicals. For the latter, the degree is undefined.

Definition 9 *The degree of a function is inductively defined by:*

$$\begin{aligned}
deg(abs) &= 1 \\
deg(id) &= 1 \\
deg(c_k) &= 0 \\
deg(pow_n) &= n \\
deg(pol_{a,b}) &= 1 \\
deg(pol_{a,b,c}) &= 2 \\
deg(f \circ g) &= deg(f) \times deg(g) \\
deg(f \times g) &= deg(f) + deg(g) \\
deg(f \pm g) &= \max\{deg(f), deg(g)\} \\
deg(f/g) &= deg(rad_n) = deg(exp_a) = deg(log_a) = \infty
\end{aligned}$$

We defined $deg(f/g) = deg(rad_n) = deg(exp_a) = deg(log_a) = \infty$ to make the definition generic, i.e., independent of particular casualities or arrangements of coefficients involved in expressions.

We introduce also *dgden* and *dgnum* to characterize the degree of the denominator and numerator for some expressions that involve quotients.

$$\begin{aligned}
dgden(f/g) &= dgden(g) \\
dgden(abs \circ f) &= dgden(f) \\
dgden(pk \circ f) &= dgden(f)
\end{aligned}$$

$$\begin{aligned}
dgnum(f/g) &= dgnum(f) \\
dgnum(abs \circ f) &= dgnum(f) \\
dgnum(pk \circ f) &= dgnum(f)
\end{aligned}$$

Rule 10 *If $f \neq id$ is strictly monotonic then*

$$\frac{\langle (f \circ g)(x) - (f \circ h)(x) \trianglelefteq 0, x, D\rangle}{\langle g(x) - h(x) \trianglelefteq 0, x, D\rangle} \text{ if } f \text{ increases}$$

$$\frac{\langle (f \circ g)(x) - (f \circ h)(x) \trianglelefteq 0, x, D\rangle}{\langle g(x) - h(x) \trianglelefteq^{-1} 0, x, D\rangle} \text{ if } f \text{ decreases}$$

Rule 11 *If $f \neq id$ is an odd function*

$$\frac{\langle (f \circ g)(x) + (f \circ h)(x) \trianglelefteq 0, x, D\rangle}{\langle (f \circ g)(x) - (f \circ (-h))(x) \trianglelefteq 0, x, D\rangle}$$

Rule 12 *(*ProductByConstant*)*

$$\frac{\langle (c_k \times f)(x) \trianglelefteq k', x, D\rangle}{\langle f(x) \trianglelefteq k'/k, x, D\rangle} \text{ if } k > 0.$$

$$\frac{\langle (c_k \times f)(x) \trianglelefteq k', x, D\rangle}{\langle f(x) \trianglelefteq^{-1} k'/k, x, D\rangle} \text{ if } k < 0.$$

$$\frac{\langle (\mathtt{0} \times f)(x) \trianglelefteq k', x, D\rangle}{\langle \mathtt{0}(x) \trianglelefteq k', x, D\rangle}$$

Rule 13 *(*ReduceToSameDenominator*)*

$$\frac{\langle (f/g)(x) \trianglelefteq k, x, D\rangle}{\langle ((f - c_k \times g)/g)(x) \trianglelefteq 0, x, D\rangle} \text{ if } k \neq 0.$$

Rule 14 *(*NullProduct*) If $\odot \in \{\times, /\}$ and $\trianglelefteq \in \{\geq, >\}$ then*

$$\frac{\langle (f \odot g)(x) \trianglelefteq 0, x, D\rangle}{\langle (f(x) \trianglelefteq 0 \wedge g(x) \trianglelefteq 0) \vee (f(x) \trianglelefteq^{-1} 0 \wedge g(x) \trianglelefteq^{-1} 0), x, D\rangle}$$

and if $\trianglelefteq \in \{\leq, <\}$ then

$$\frac{\langle (f \odot g)(x) \trianglelefteq 0, x, D\rangle}{\langle (f(x) \trianglelefteq 0 \wedge g(x) \trianglelefteq^{-1} 0) \vee (f(x) \trianglelefteq^{-1} 0 \wedge g(x) \trianglelefteq 0), x, D\rangle}$$

and

$$\frac{\langle (f \times g)(x) = 0, x, D\rangle}{\langle f(x) = 0 \vee g(x) = 0, x, D\rangle}$$

$$\frac{\langle (f \times g)(x) \neq 0, x, D\rangle}{\langle f(x) \neq 0 \wedge g(x) \neq 0, x, D\rangle}$$

$$\frac{\langle (f/g)(x) = 0, x, D\rangle}{\langle f(x) = 0, x, D\rangle}$$

$$\frac{\langle (f/g)(x) \neq 0, x, D\rangle}{\langle f(x) \neq 0, x, D\rangle}$$

We note that in $\langle (f/g)(x) \trianglelefteq 0, x, D\rangle$, we assumed before that D is contained in $\mathcal{D}_{f/g}$, which justifies the correction of the above rules.

Also, to unify the expressions that are obtained by the solver with the head of the presented rules, the solver does, in some situations, some arithmetic simplications.

In order to apply the rewriting rules, we have also to perform some controlled algebraic manipulations. The idea is to limit the number of times an expression may be rewritten to guarantee termination. The solver needs rules translating the properties of the operations (commutative, associative, distributive, ...). It also needs special rules. In particular, to get rid of *abs* and to lift pow_n and rad_n

$$abs \equiv ((id, \mathbb{R}_0^+), (-id, \mathbb{R}^-))$$

$$(c_k \times pow_{2n+1}) \circ f \equiv pow_{2n+1} \circ ((rad_{2n+1} \circ c_k) \times f)$$

$$(c_k \times rad_{2n+1}) \circ f \equiv rad_{2n+1} \circ ((pow_{2n+1} \circ c_k) \times f)$$

$$(c_k \times rad_{2n}) \circ f \equiv sign(k) rad_{2n} \circ ((pow_{2n} \circ abs \circ c_k) \times f)$$

$$(c_k \times pow_{2n}) \circ f \equiv sign(k) pow_{2n} \circ ((rad_{2n} \circ abs \circ c_k) \times f)$$

with $sign(k)$ representing the signal of k. Moreover, we need to manipulate polynomials addition and multiplication,

$$pol_{a_n,\ldots,a_0} + pol_{b_m,\ldots,b_0} \equiv pol_{a_n,\ldots,a_{m-1},a_m+b_m,\ldots a_0+b_0}$$

where $n \geq m$ and

$$pol_{a_n,\ldots,a_0} \equiv pol_{a_j,\ldots,a_0}$$

if $a_j \neq 0$ and $a_i = 0$, for all $i > j > 0$,

$$k \times pol_{a_n,\ldots,a_0} \equiv pol_{ka_n,\ldots,ka_0},$$

and we also need to define the product of two polynomials.

5 Conclusions

Applications of CAL to math education require a careful analysis of procedures that students usually apply to solve math drills to design generic solvers with pedagogic relevance. We claim that the solver proposed in this paper fulfils this requirement. At the same time, it is quite generic, for it takes advantage of a declarative approach. The solver is being implemented in Prolog. We are going also to annotate the rewriting rules to provide explanations of the exercises in natural language, following ideas of [10].

References

[1] K. R. Apt. *Principles of constraint programming.* Cambridge Univ. Press, (2003).

[2] M. Beeson. Design Principles of Mathpert: Software to support education in algebra and calculus. In N. Kajler (ed.), *Computer-Human Interaction in Symbolic Computation*, Texts and Monographs in Symbolic Computation, Springer-Verlag (1998), 89-115.

[3] M. Carlsson, G. Ottosson, B. Carlson. An open-ended finite domain constraint solver. In H. Glaser, P. Hartel, H. Kuchen (Eds), *Programming Languages: Implementations, Logics and Programs (PLILP'97)*, Lecture Notes in Computer Science **1292**, Springer-Verlag (1997) 191-206.

[4] FP6-Project "LeActiveMath": Language-Enhanced User-Adaptive, Interactive eLearning for Mathematics, 2004/2006. (`http://www.dfki.de/leactivemath/`)

[5] C. Holzbaur. OFAI clp(q,r) Manual, Edition 1.3.3. Austrian Research Institute for Artificial Intelligence, TR-95-09, Vienna (1995).

[6] R. Isidro. *SA^3C – A system for computer assisted assessement and learning.* MSc thesis, University of Aveiro, Portugal (2004). (in Portuguese)

[7] IST-Project "MoWGLI": Mathematics On the Web: Get it by Logic and Interfaces, 2002/2005. (`http://www.mowgli.cs.unibo.it/`)

[8] K. Marriott, P. Stuckey. Programming with Constraints – An Introduction. MIT Press (1998).

[9] E. Melis, E. Andrés, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, C. Ullrich. ActiveMath: A Generic and Adaptive Web-Based Learning Environment, *International Journal of Artificial Intelligence in Education* **12(4)** (2001) 385–407. (`http://www.activemath.org/`)

[10] A. Ranta. Grammatical Framework: A Type-Theoretical Grammar Formalism. *Journal of Functional Programming* **14(2)** (2004) 145–189.

[11] A. Robinson, A. Voronkoy (Eds). *Handbook of Automated Reasoning*, Vol I, Elsevier Science (2001).

[12] SICStus Prolog User Manual (Release 3.12.0), SICS, Sweden (2004).
`http://www.sics.se/sicstus/`

[13] The YAP Prolog System, release 4.4.4, 2004. (`http://www.ncc.up.pt/~vsc/YAP`)

[14] A. P. Tomás, J. P. Leal. A CLP-Based Tool for Computer Aided Generation and Solving of Maths Exercises. In V. Dahl, P. Wadler (Eds), *Practical Aspects of Declarative Languages, 5th Int. Symposium PADL 2003*, Lecture Notes in Computer Science **2562**, Springer-Verlag (2003) 223–240.

[15] G. Xiao. WIMS – An Interactive Mathematics Server, *Journal of Online Mathematics and its Applications* **1**, MAA (2001). (`http://wims.unice.fr`)

[16] G. Xiao. On Public-Questions Tests. Univ. Nice Sophia-Antipolis, France (2004).