Designing a Solver for Arithmetic Constraints to Support Education in Mathematics*

Ana Paula Tomás, Nelma Moreira, and Nuno Pereira

DCC-FC & LIACC
University of Porto, Portugal
{apt,nam,nfp}@dcc.fc.up.pt

Abstract. We present a conditional rewrite system for arithmetic and membership univariate constraints over real numbers, designed for computer assisted learning (CAL) in elementary math. Two fundamental principles guided the design of the proposed rewrite rules: *cognitive fidelity* (emulating steps students should take) and *correctness*, aiming that step-by-step solutions to problems look like ones carried out by students. In order to gain more flexibility to modify rules, add new ones and customize solvers, the rules are written in a specification language and then compiled to Prolog. The rewrite system is complete for a relevant subset of problems found in high-school math textbooks.

1 Introduction

To understand what people do when they do mathematics and write programs emulating that process is a continuous research topic in Artificial Intelligence, Automated Reasoning, and Symbolic Computation [5, 12]. Computer Mathematics is by now an established, although developing, subject. The challenge is to make the systems, including Computer Algebra systems and Proof Assistants, more (mathematician-)friendly [1].

Symbolic computation systems, like the commercial packages Maple and Mathematica, are widely used, though they can produce unexpected or wrong answers [1, 2, 7]. Nevertheless, in order to reduce the effort of writing solvers, some web-based learning environments and e-learning authoring tools support (unsafe) interaction with them [9, 14]. An attempt to build an educational system based on Maple that overcomes some of these problems of incorrectness is described in [11]. It required a good deal of programming. These packages were not developed specifically for education, which makes it difficult to get them generate step-by-step solutions that are *cognitive faithful* also. Indeed, when generating solutions,

* 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).

they seldom take the same steps as the students should take. In [2], a discussion about design criteria of software for mathematics education is given. These principles actually guided the design of *MathXpert*: a (commercial) menu-based system for algebra, trigonometry and calculus [3], implementing a over a thousand of operations, and that may run also in automatic mode.

AGILMAT – *Automatic Generation of Interactive Drills for Mathematics Learning* (`www.ncc.up.pt/AGILMAT/`) – aims at the design and implementation of a system to automatically create and solve math exercises, continuing research work reported in [13]. This work led to developing a prototype, called DEMOMATH, that also yields one-line solutions for some exercises. Its solver is fairly ad-hoc, and cannot be easily adapted to present step-by-step solutions with pedagogic interest, which motivated our current work. We propose a conditional rewrite system for arithmetic and membership univariate constraints over real numbers. To gain flexibility, the rules are written in a specification language and then compiled to Prolog.

In the next section we recall basic notions of real-valued functions and give examples of problems we want to automate. In Section 3 we introduce our representation for problems and constraints and show how to convert membership to arithmetic constraints, and reciprocally. Section 4 is devoted to the presentation of the proposed rewriting system, which was designed to be complete for the problems that can be solved by analyzing the sign variation of functions created by DEMOMATH.

2 Some Mathematical Background and Examples

We start with some notions about real-valued functions. $\mathbb{R}$ 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 the function f, and its image (a.k.a., range) is $f(\mathcal{D}_f) = \{f(x) : x \in \mathcal{D}_f\}$. We represent the restriction of f to $D \subseteq \mathcal{D}_f$ by $f|_D$ and the inverse function by f^{-1}, if it exists. If f is strictly monotonic over D, then $f|_D$ is invertible. A function f is even iff $f(x) = f(-x)$ for all $x \in \mathbb{R}$ and odd iff $f(x) = -f(-x)$, for all $x \in \mathbb{R}$. Table 1 shows the basic functions studied in math at high school, if we exclude the trigonometric functions and generic polynomial functions $pol_{a_n,\ldots,a_0} : x \mapsto \sum_{i=0}^{n} a_i x^i$.

Composition, sum, difference, product and quotient of functions are represented by $\circ$, $+$, $-$, $\times$ and $/$. We have $\mathcal{D}_{f\circ g} = \mathcal{D}_g \cap \{x : g(x) \in \mathcal{D}_f\}$, $\mathcal{D}_{f\odot g} = \mathcal{D}_f \cap \mathcal{D}_g$ for $\odot \in \{+, -, \times\}$ and $\mathcal{D}_{f/g} = \mathcal{D}_f \cap \mathcal{D}_g \setminus \{x : g(x) = 0\}$.

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.

A piecewise function f is of form $(f_i, D_i)_{i=1}^n$, with $n \geq 2$, being $f(x)$ given by $f_i(x)$ if $x \in D_i$. Table 1 contains properties that students learn and so solvers must also refer to them. This knowledge base, may be extended or reduced if appropriate.

Drills and practice We now give some examples of exercises we are interested in automating. The first ones are from a high school math textbook (grade 11). To create the two last ones and get their solution we have used DEMOMATH (available at `www.ncc.up.pt/~zp/demomath`).

- Find the domain of $f(x) = \frac{\sqrt[3]{1-x^2}}{4-x+\sqrt{x+2}}$.
- Express $g(x) = |x-1| + |x+1| + x$ without using the absolute value function. Solve $g(x) < |x+3|$.
- Study the sign variation of $-\sqrt{\frac{2|2x^2+2x|}{-3x-1}} - 1$ for $x \in \mathbb{R}$. (Solution: Negative in $]-\infty, -\frac{1}{3}] \cup \{0\}$ (the domain of the expression)).
- Solve $\left(2\ \left|-x^2 - 2\,x + 1\right|\right)^2 \left(2\,x^4 - \ x^2 - 3\right) \neq 0$ for $x \in \mathbb{R}$. (Solution: $]-\infty, \infty[\setminus \{-1-\sqrt{2}, -\frac{1}{2}\sqrt{6}, -1+\sqrt{2}, \frac{1}{2}\sqrt{6}\}$)

In DEMOMATH, the representation of the last constraint corresponds to $((pow_2 \circ pol_{2,0} \circ abs \circ pol_{-1,-2,1}) \times (pol_{2,-1,-3} \circ pow_2))(x) \neq 0$ for $x \in \mathbb{R}$.

Generic constraints involving $f \odot g$, for $\odot \in \{+, -, \times, /\}$ are often hard to solve (or undecidable). Hence, if we want to guarantee that there exists a solving procedure for the problem, we have to restrict to decidable subsets. Currently, the expressions DEMOMATH may create are described by the grammar given in [13]. Students may compute domains, zeros and study sign variation of functions defined by them provided they can solve linear or quadratic equations, equations of the form $aX^n + b = 0$, $a\sqrt[n]{X}+b = 0$, $X^n \pm Y^n = 0$, $\sqrt[n]{X} \pm \sqrt[n]{Y} = 0$, for $n \geq 2$, or $X/Y \pm Z/T = 0$, with $degree(XT) \leq 2$ and $degree(YZ) \leq 2$, and get rid of absolute values.

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. We define *atomic* and *complex* constraints as follows.

Definition 1. *The* atomic arithmetic constraints *are either of the form* $f(x) \lhd g(x)$ *and* $f(x) \lhd k$ *with* $\lhd \in \{=, \neq, >, <, \leq, \geq\}$, f *and* g *are real valued functions on reals and* k *is a ground arithmetic-term. The* atomic membership constraints *are of form* $f(x) \lhd S$ *with* $\lhd \in \{\in, \notin\}$ *and* S *is a ground set-term. The conjunction and disjunction of a finite number of constraints in the variable* x *is* a (complex) constraint $C(x)$.

We often write C instead of $C(x)$, omitting the variable, which shall cause no confusion, since we will address only problems that involve a unique variable. We use $\lhd^{-1}$ to denote the inverse of the binary relation $\lhd$, for $\lhd \in \{=, \neq, \leq, \geq, >, <\}$. Clearly, $\leq^{-1}$ is $\geq$, $<^{-1}$ is $>$, and $=^{-1}$ is $=$ and $\neq^{-1}$ is $\neq$.

Definition 2. *We inductively define* the domain of constraint C *(denoted by* $\mathcal{D}_C$*) by* $\mathcal{D}_{f(x) \lhd g(x)} = \mathcal{D}_f \cap \mathcal{D}_g$, $\mathcal{D}_{f(x) \lhd k} = \mathcal{D}_{f(x) \lhd S} = \mathcal{D}_f$, $\mathcal{D}_{\wedge_{i=1}^n C_i} = \cap_{i=1}^n \mathcal{D}_{C_i}$ *and* $\mathcal{D}_{\vee_{i=1}^n C_i} = \cup_{i=1}^n \mathcal{D}_{C_i}$.

By this definition, e.g., the domain of $(\sqrt{x} = 5 \vee 6x < -7)$ is $\mathbb{R}$ but the domain of $(\sqrt{x} = 5 \wedge 6x < -7)$ is $\mathbb{R}_0^+$. We now go through our representation for *problems*.

Definition 3. *The* problem P *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* $\langle id(x) \in D, x, D \rangle$ *and* D *is then called* the solution set *of the problem. (For short, we shall write* $\langle x \in D, x, D \rangle$ *instead.) We use* `sols`(P) *to denote the solution set of* P.

3.1 Membership versus Arithmetic Constraints

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

Definition 4. *A set is in a* standard 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 a subset of* $\mathbb{R}$ *that may be written in standard form.*

Although the *constraining sets* do not fully represent all subsets of $\mathbb{R}$, they cater for the most frequent types of sets that occur in math drills, if trigonometry is excluded. Nevertheless, we are considering extensions to be able to deal with trigonometric functions also.

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 standard form. (Here, we use French notation, writing $[-3,-1[$ instead of $[-3,-1)$, for example.)

This standard form is like a picture of the set represented in the real axis. We now introduce *the reduced normal form* which gives a more compact *arithmetic* representation of each constraining set, being thus relevant for CAL. The reduced normal form is unique.

Definition 5. *A constraining set is* in reduced normal form *(*`rnf`*, for short) iff it is given in one of the following forms:* $\mathbb{R}$*,* $\emptyset$*, a finite non-empty set,* $\cup_{i=1}^{n} S_i$*,* $\mathbb{R} \setminus S_{n+1}$*,* $(\cup_{i=1}^{n} S_i) \setminus S_{n+1}$*,* $((\cup_{i=1}^{n} S_i) \setminus S_{n+1}) \cup S_{n+2}$*, 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. For instance, $\texttt{rnf}([-3,-1[\cup\{2,17\}\cup[8,11[\cup]11,14[)\setminus\{10\}) = (([-3,-1[\cup[8,14[) \setminus \{10,11\}) \cup \{2,17\}$.

Definition 6. *Let* $\mathcal{S}^k_{\lhd}$ *denote the set* $\{x \in \mathbb{R} : x \lhd k\}$*, for* $k \in \mathbb{R}$ *and* $\lhd \in \{=, \neq, >, <, \leq, \geq\}$*.*

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

To help transform membership constraints into arithmetic constraints we introduce τ_1 that writes sets given in reduced normal form in terms of $\mathcal{S}^k_{\lhd}$'s, for suitable k's and $\lhd$'s and is defined as follows.

Definition 7. *The map τ_1 acts on* `rnf`*-sets and it is given by:* $\tau_1(\mathbb{R}) = \mathbb{R}$, $\tau_1(\emptyset) = \emptyset$, $\tau_1(\{a_1, \ldots, a_n\}) = \cup_{i=1}^n \mathcal{S}_{=}^{a_i}$, $\tau_1(\,[a,b]\,) = \mathcal{S}_{\geq}^{a} \cap \mathcal{S}_{\leq}^{b}$, $\tau_1(\,[a,b[\,) = \mathcal{S}_{\geq}^{a} \cap \mathcal{S}_{<}^{b}$, $\tau_1(\,]a,b]\,) = \mathcal{S}_{>}^{a} \cap \mathcal{S}_{\leq}^{b}$, $\tau_1(\,]a,b[\,) = \mathcal{S}_{>}^{a} \cap \mathcal{S}_{<}^{b}$, $\tau_1(\,[a,+\infty[\,) = \mathcal{S}_{\geq}^{a}$, $\tau_1(\,]-\infty,a]\,) = \mathcal{S}_{\leq}^{a}$, $\tau_1(\,]a,+\infty[\,) = \mathcal{S}_{>}^{a}$, $\tau_1(\,]-\infty,a[\,) = \mathcal{S}_{<}^{a}$, *for* $a, b \in \mathbb{R}$, *and, finally,* $\tau_1(\mathbb{R}\setminus\{a_1 \ldots, a_n\})) = \cap_{i=1}^n \mathcal{S}_{\neq}^{a_i}$, $\tau_1(A\setminus\{a_1 \ldots, a_n\}) = \tau_1(A) \cap (\cap_{i=1}^n \mathcal{S}_{\neq}^{a_i})$, *for* $A \neq \mathbb{R}$, *and* $\tau_1(\cup_{i=1}^n A_i) = \cup_{i=1}^n \tau_1(A_i)$.

This syntactic transformation of a `rnf`-set S is quite convenient to convert $f(x) \in S$ into arithmetic constraints by τ_2, for $\emptyset \neq S \neq \mathbb{R}$.

Definition 8. *The transformation τ_2 acts on membership constraints $f(x) \in S$, for S presented in terms of $\mathcal{S}_{\lessdot}^{k}$'s, being inductively given by:* $\tau_2(f(x) \in \mathbb{R}) = (f(x) \in \mathbb{R})$, $\tau_2(f(x) \in \emptyset) = (f(x) \in \emptyset)$, $\tau_2(f(x) \in \mathcal{S}_{\lessdot}^{k}) = (f(x) \lessdot k)$, $\tau_2(f(x) \in \cup_{i=1}^n S_i) = (\vee_{i=1}^n \tau_2(f(x) \in S_i))$ *and* $\tau_2(f(x) \in \cap_{i=1}^n S_i) = (\wedge_{i=1}^n \tau_2(f(x) \in S_i))$.

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 standard form [13]. Union, intersection and set difference are translated by `cup`, `cap` and `setminus`. Some symbolic representations were introduced for $\mathcal{S}_{\lessdot}^{k}$, e.g., `s(real)`, `s([])`, `s(K,eq)`, `s(K,lt)`, `s(K,leq)`. Exact arithmetic for a subset of $\mathbb{R}$ is supported also by a module defined for DEMOMATH, that uses CLP($\mathbb{Q}$) for some computations [8], since we are extending this prototype to implement core modules of AGILMAT system (exercise generators and solvers).

For every given constraining set S (s.t. $\emptyset \neq S \neq \mathbb{R}$) and function f, 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 $f(x) \in S$. Because we consider $x \in S$ simpler than $\Gamma(x \in S)$, we introduce yet another transformation $\tilde{\Gamma}$ defining it by $\tilde{\Gamma}(id(x) \in S) = (id(x) \in \texttt{rnf}(S))$ and $\tilde{\Gamma}(f(x) \in S) = \Gamma(f(x) \in S)$, for $f \neq id$.

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

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

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

Let us suppose that f is $rad_3 \circ pol_{2,-7}$, i.e., $f(x) = \sqrt[3]{2x-7}$. For solving the problem $\langle f(x) \in S, x, \mathbb{R}\rangle$, students transform the membership constraint to arithmetic constraints. Our solver will do exactly the same thing. Although $f(x) \in S$ is also trivially equivalent to $((f(x) \geq -3 \wedge f(x) < -1) \vee (f(x) \geq 8 \wedge f(x) < 11) \vee f(x) > 11)$, the proposed form is simpler and we think it is pedagogically relevant to adopt it instead.

Each atomic constraint of form $C = (f(x) \trianglelefteqslant k)$ or $C = (f(x) \trianglelefteqslant S)$, is equivalent to $(f(x) \in \texttt{ctrSet}(C))$, for $\texttt{ctrSet}(C)$ given by $\texttt{ctrSet}(f(x) \in S) = S$, $\texttt{ctrSet}(f(x) \notin S) = \mathbb{R} \setminus S$ and $\texttt{ctrSet}(f(x) \trianglelefteqslant k) = \mathcal{S}^k_{\trianglelefteqslant}$.

We introduce a partial function $\texttt{nf}$ that writes some constraints to a standard form: $\texttt{nf}(f(x) \trianglelefteqslant \beta) = (\tilde{\Gamma}(f(x) \in \texttt{ctrSet}(f(x) \trianglelefteqslant \beta)))$ for ground β and $\texttt{nf}(\bigotimes_{i\in I}(f(x) \trianglelefteqslant_i \beta_i)) = (\tilde{\Gamma}(f(x) \in \tilde{\bigotimes}_{i\in I}\texttt{ctrSet}(f(x) \trianglelefteqslant_i \beta_i)))$ for ground β_i. Here $\tilde{\vee} = \cup$ and $\tilde{\wedge} = \cap$.

4 Solving Problems

To achieve *cognitive faithful* solvers [2], we cannot manipulate problems and constraints in an arbitrary way. Each of the *rewrite rules* we propose uses some extra mathematical knowledge, e.g. about functions behavior, and, if applicable, transforms a problem into an equivalent one, under some specific conditions. For example, we introduced BoundRange whose main goal is to check whether an atomic constraint is valid or inconsistent based on functions range. BoundRange states that: *for any generic functions f and g, with $f \neq id$, and any ground set-term or arithmetic-term β, if $D \subseteq \mathcal{D}_{f\circ g}$ and $\mathcal{E}$ is such that $f(\mathcal{D}_f) \subseteq \mathcal{E}$ then*

$\langle (f \circ g)(x) \trianglelefteqslant \beta, x, D\rangle \rightarrow \langle x \in D, x, D\rangle$ **if** $\texttt{ctrSet}((f \circ g)(x) \trianglelefteqslant \beta) \supseteq \mathcal{E}$;
$\langle (f \circ g)(x) \trianglelefteqslant \beta, x, D\rangle \rightarrow \langle x \in \emptyset, x, \emptyset\rangle$ **if** $\texttt{ctrSet}((f \circ g)(x) \trianglelefteqslant \beta) \cap \mathcal{E} = \emptyset$;
$\langle (f \circ g)(x) \trianglelefteqslant \beta, x, D\rangle \rightarrow \langle \tilde{\Gamma}((f \circ g)(x) \in S \cap \mathcal{E}), x, D\rangle$
if $\trianglelefteqslant \notin \{=, \neq\}$, $\emptyset \neq S \cap \mathcal{E} \neq S$ *and* $S \not\supseteq \mathcal{E}$, *where* $S = \texttt{ctrSet}((f \circ g)(x) \trianglelefteqslant \beta)$.

We see that rewrite rules look like $P \rightarrow P'$ *if condition* although some preconditions were stated in a global head. Nevertheless, they could be moved to the *condition* part of each branch. It is not completely clear from this kind of mathematical representation which is the operational reading of rules. Moreover, implicit meta-knowledge should be made explicit in order to be able to explain solution steps. Because of that, we developed a language for a lower level specification of rewrite rules. The corresponding low level formulation of BoundRange looks as follows. Relevant conditions for writing explanations are annotated with (#).

```
BoundRange(P)
  begin
    is_atomic(P:ctr), is_ground(P:ctr:rhs),
    subseteq(func_dom(P:ctr:lhs:func),P:dom),
    (#)P:ctr:lhs:func =? F ∘ G,  !F =? id,
    E := (#)boundImage(F,func_dom(F)),
    S := (#)ctrSet(P:ctr)
    if (#)supseteq(S,E), (#)note("valid %", P:ctr)
      rewrite_to sfprob(P:var,inset,P:dom)
    elif (#)seteq(S cap E,s([])), (#)note("inconsistent %", P:ctr)
      rewrite_to sfprob(P:var,inset,s([]))
    else !inlist(P:ctr:op,[eq,neq]),
      (#)note("necessarily %", ctr(P:ctr:lhs:func,P:var,inset,E cap S)),
      !seteq((#)rnf(E cap S),S) rewrite_to
      prob(tgm(ctr(P:ctr:lhs:func,P:var,inset,E cap S)),P:var,P:dom)
    endif
  end
```

A compiler is being developed for this language so that we gain also more flexibility to modify rules, add new ones and customize solvers to different users or curricula. Each rule is compiled to a Prolog predicate. The `if`-block is translated to an auxiliary predicate, whose clauses correspond to the branches of the `if`-block. A single branch may succeed. We now describe this language formally, omitting reference to rule annotations.

4.1 A Specification Language for Rewrite Rules

The application of a rule to a problem will be another problem if the conditions of the rule are satisfied. Otherwise, the application of the rule fails. The specification language is a functional language with implicit types. Primitive (data)types are `boolean`, `real`, `set`, `function`, `constraint` and `problem`, that correspond trivially to the objects defined in the previous sections, and, also, `op_r` and `op_e` for relational operators and expression operators, respectively. All built-in constructs are typed and every rule definition must be type checked. In Table 2 a fragment of the syntax of the rule specification language is presented. The definition of a rule consists of a name, a parameter (of type `problem`) and a sequence of (atomic) conditions followed either by a nested `if_block` or by a **rewrite_to** `exp`, where `exp` corresponds to the resulting problem, if no condition is *false*. An `if` block allows for nested conditions, and must have at least two branches each one ended by **rewrite_to**. An atomic condition (of type `boolean`), or its negation (`!`), will allow the specification and the verification of mathematical knowledge as relations between functions, sets and

```
rule        ::= rule_name "(" var ")" begin conditions body end
body        ::= rewrite_to exp | if_block
if_block    ::= if conditions body (elif conditions body)*
                else conditions? body endif
conditions  ::= condition ("," condition)*
condition   ::= atomic_cond | "!" atomic_cond
atomic_cond ::= predicate | rel | let | matching | inlist
predicate   ::= bfunc "(" exp_1 "," ... "," exp_n ")"  n ≥ 1
rel         ::= rel "(" var "," exp "," exp ")" | exp op_r exp
let         ::= var ("," var)* := exp
matching    ::= exp =? exp
inlist      ::= inlist "(" exp "," "[" listobj "]" ")"
listobj     ::= atom ("," atom)*
exp         ::= var | atom | efunc "(" exp_1 "," ... "," exp_n ")"
                | var:sel_1:...:sel_n | exp op_e exp
                | apply "(" exp "," exp ")"  n ≥ 1
atom        ::= problem | constraint | set | function | real
                | op_r | op_e
```

Table 2. A Grammar for the Rules Specification Language.

real numbers; equality of problems or constraints; properties of functions; transformations and computations, etc. The syntactic constructs are:

`predicate` Any function that yields a truth value. Built-in predicates include tests for (build-in) types, e.g `is_atomic` for constraints, `in_sf` that tests if a problem is in solved form; properties of functions, e.g. `decreasing`, etc.

`rel` Infix notation for the usual relational operations on sets, etc; in **rel** the first argument must be a variable of type `op_r`.

`let` Binds variable identifiers to values (the condition is *false*, if the values cannot be computed)

`matching` Matching of two expressions (of the same type), with the left-hand side ground; is *false* if the matching is not possible.

`inlist` Tests if an object belongs to a set, e.g. **inlist**(X,[`inset`,`notin`]).

Finally, expressions include, for each type, variable identifiers (beginning with uppercase letters), constants (as defined in the previous sections), selectors (e.g. P`:var`), constructors (e.g ctr(`id`,P`:var`,`inset`,`s([])`)), infix operations and function application. The **apply** constructor is used only to evaluate real valued functions. We can have meta functions as `rewriting` that calls the *system solver* for rewriting sub-problems.

Solver execution Besides defining the rewrite rules, we need to specify how they are applied for solving problems. For that we use the notion of

strategy [4, 6]. A trivial strategy will be to try to apply all available rules until either a solved form or an upper bound on the number of steps (rule applications) is reached. For that we may have the following trivial solver (implemented in `Prolog`):

```
rewrite(P,P,_,N,NMax) :- stop(P,N,NMax),!.
rewrite(P,P1,S,N,NMax) :- rewrite_one(P,P0,S),
        N1 is N+1, rewrite(P0,P1,S,N1,NMax).
rewrite_one(P,P1,S):- choose_one_rule(R,S), apply_rule(R,P,P1).
stop(P,N,NMax):- in_sf(P); N > NMax.
```

4.2 Cognitive faithful rewriting rules

We now present the rewrite rules we propose, that contribute to the novelty of this work. As usual, $\Rightarrow, \Rightarrow^+$ and $\Rightarrow^\star$ stand for the (one-step) rewriting relation, its transitive irreflexive closure (one or more rewrite steps) and its transitive reflexive closure (zero or more steps). The constraint C is valid in D if $\langle C, x, D\rangle \Rightarrow^\star \langle id(x) \in D, x, D\rangle$ and inconsistent in D if $\langle C, x, D\rangle \Rightarrow^\star \langle id(x) \in \emptyset, x, \emptyset\rangle$. We shall write x instead of $id(x)$ and assume that D is in `rnf`-form. This property is kept invariant by the rewrite rules and therefore, if $P \Rightarrow^\star \langle x \in D', x, D'\rangle$ and $P \Rightarrow^\star \langle x \in D'', x, D''\rangle$ then $D' = D''$. If nothing else is said, f, g, h are generic functions (including id). First we present ReduceProbDomain, that says that solutions must be in $D \cap \mathcal{D}_C$. Then, we give four rules for tackling complex constraints and the rules for atomic constraints, omitting BoundRange.

(ReduceProbDomain)

$\langle C, x, \emptyset\rangle \rightarrow \langle x \in \emptyset, x, \emptyset\rangle$ **if** $C \neq (x \in \emptyset)$.
$\langle C, x, D\rangle \rightarrow \langle C, x, \mathtt{rnf}(D \cap \mathcal{D}_C)\rangle$ **if** $D \neq \emptyset$ and $\mathcal{D}_C \cap D \neq D$.

(SplitConstraints) To rewrite several top level conjuncts (or disjuncts). *Let* I_1 *and* I_2 *be finite sets,* $|I_1| \geq 2$ *and* $I_2 \neq \emptyset$. *If* $\bigotimes \in \{\vee, \wedge\}$

$\langle \bigotimes_{i\in I_1\cup I_2} C_i, x, D\rangle \rightarrow \langle (x \in S_a) \bigotimes \bigotimes_{i\in I_2} C_i, x, D\rangle$
  if $\langle \bigotimes_{i\in I_1} C_i, x, D\rangle \Rightarrow^+ \langle x \in S_a, x, S_a\rangle$.
$\langle \bigotimes_{i\in I_1\cup I_2} C_i, x, D\rangle \rightarrow \langle C'_a \bigotimes \bigotimes_{i\in I_2} C_i, x, D\rangle$
  if $\langle \bigotimes_{i\in I_1} C_i, x, D\rangle \Rightarrow^+ \langle C'_a, x, D\rangle$ and $C'_a \neq (x \in D)$.
$\langle \bigotimes_{i\in I_1\cup I_2} C_i, x, D\rangle \rightarrow \langle (x \in S_a \wedge C'_a) \bigotimes \bigotimes_{i\in I_2} C_i, x, D\rangle$
  if $\langle \bigotimes_{i\in I_1} C_i, x, D\rangle \Rightarrow^+ \langle C'_a, x, S_a\rangle$, $C'_a \neq (x \in S_a)$ and $D \neq S_a$.

(AggregateNormalize) To rewrite several top level atomic constraints $f(x) \trianglelefteq_i \beta_i$ to a simpler form (may detect inconsistency/validity).

$\langle \bigotimes_{i\in I_1\cup I_2} C_i, x, D\rangle \rightarrow \langle \bigotimes_{i\in I_1} C_i \bigotimes \mathtt{nf}(\bigotimes_{i\in I_2} C_i), x, D\rangle$
  if $C_i = (f(x) \trianglelefteq_i \beta_i)$, β_i is ground, for $i \in I_2$, $|I_2| \geq 2$ and $\mathtt{nf}(\bigotimes_{i\in I_2} C_i) \neq \bigotimes_{i\in I_2} C_i$.

(Conjunctive) To rewrite a single conjunct at top level.

$\langle \wedge_{i=1}^{n} C_i, x, D\rangle \to \langle \wedge_{i=1}^{j-1} C_i \wedge C'_j \wedge \wedge_{i=j+1}^{n} C_i, x, D\rangle$
  if $\langle C_j, x, D\rangle \Rightarrow^{+} \langle C'_j, x, D\rangle$ and $C'_j \neq (x \in D)$.
$\langle \wedge_{i=1}^{n} C_i, x, D\rangle \to \langle \wedge_{i=1}^{j-1} C_i \wedge C'_j \wedge \wedge_{i=j+1}^{n} C_i, x, D'\rangle$
  if $\langle C_j, x, D\rangle \Rightarrow^{+} \langle C'_j, x, D'\rangle$, $C'_j \neq (x \in D')$ and $D' \neq D$
$\langle \wedge_{i=1}^{n} C_i, x, D\rangle \to \langle x \in \emptyset, x, \emptyset\rangle$ **if** $\langle C_j, x, D\rangle \Rightarrow^{+} \langle x \in \emptyset, x, \emptyset\rangle$
$\langle \wedge_{i=1}^{n} C_i, x, D\rangle \to \langle \wedge_{i=1, i\neq j}^{n} C_i, x, D\rangle$ **if** $\langle C_j, x, D\rangle \Rightarrow^{\star} \langle x \in D, x, D\rangle$
$\langle \wedge_{i=1}^{n} C_i, x, D\rangle \to \langle \wedge_{i=1, i\neq j}^{n} C_i, x, D'\rangle$ **if** $\langle C_j, x, D\rangle \Rightarrow^{+} \langle x \in D', x, D'\rangle$, $D \neq D' \neq \emptyset$

(Disjunctive) To rewrite a single disjunct at top level.

$\langle \vee_{i=1}^{n} C_i, x, D\rangle \to \langle \vee_{i=1}^{j-1} C_i \vee (x \in D' \wedge C'_j) \vee \vee_{i=j+1}^{n} C_i, x, D\rangle$
  if $\langle C_j, x, D\rangle \Rightarrow^{+} \langle C'_j, x, D'\rangle$, $C'_j \neq (x \in D')$, $D' \neq D$ and $C_j \neq (x \in D' \wedge C'_j)$.
$\langle \vee_{i=1}^{n} C_i, x, D\rangle \to \langle \vee_{i=1}^{j-1} C_i \vee C'_j \vee \vee_{i=j+1}^{n} C_i, x, D\rangle$
  if $\langle C_j, x, D\rangle \Rightarrow^{+} \langle C'_j, x, D\rangle$, and $C'_j \neq (x \in D)$.
$\langle \vee_{i=1}^{n} C_i, x, D\rangle \to \langle \vee_{i=1, i\neq j}^{n} C_i, x, D\rangle$ **if** $\langle C_j, x, D\rangle \Rightarrow^{+} \langle x \in \emptyset, x, \emptyset\rangle$.
$\langle \vee_{i=1}^{n} C_i, x, D\rangle \to \langle x \in D, x, D\rangle$ **if** $\langle C_j, x, D\rangle \Rightarrow^{\star} \langle x \in D, x, D\rangle$.
$\langle \vee_{i=1}^{n} C_i, x, D\rangle \to \langle x \in \mathtt{rnf}(D' \cup D''), x, \mathtt{rnf}(D' \cup D'')\rangle$
  if $\langle C_j, x, D\rangle \Rightarrow^{\star} \langle x \in D', x, D'\rangle$, $\emptyset \neq D' \neq D$
  and $\langle \vee_{i=1, i\neq j}^{n} C_i, x, \mathtt{rnf}(D \setminus D')\rangle \Rightarrow^{\star} \langle x \in D'', x, D''\rangle$.

(ArithNormalize) To convert a membership constraint to an arithmetic constraint if the latter is simpler.

$\langle f(x) \trianglelefteqslant S, x, D\rangle \to \langle \mathtt{nf}(f(x) \trianglelefteqslant S), x, D\rangle$
  if $f \neq id$, $\trianglelefteqslant \in \{\in, \notin\}$ and $\mathtt{nf}(f(x) \trianglelefteqslant S) \neq (f(x) \trianglelefteqslant S)$.

(DefRealValuedFunc)

$\langle f(x) \in \emptyset, x, D\rangle \to \langle x \in \emptyset, x, \emptyset\rangle$ **if** $f \neq id$.
$\langle f(x) \notin \emptyset, x, D\rangle \to \langle x \in D, x, D\rangle$ **if** $f \neq id$ and $D \subseteq \mathcal{D}_f$.
$\langle f(x) \in \mathbb{R}, x, D\rangle \to \langle x \in D, x, D\rangle$ **if** $f \neq id$ and $D \subseteq \mathcal{D}_f$.
$\langle f(x) \notin \mathbb{R}, x, D\rangle \to \langle x \in \emptyset, x, \emptyset\rangle$ **if** $f \neq id$.

(DomainAtomConstr)

$\langle C, x, D\rangle \to \langle x \in \mathtt{rnf}(D \cap \mathtt{ctrSet}(C)), x, \mathtt{rnf}(D \cap \mathtt{ctrSet}(C))\rangle$
  if $C = (x \trianglelefteqslant \beta)$ and $C \neq (x \in D)$.

(ConstantFunc)

$\langle c_k(x) \trianglelefteqslant f(x), x, D\rangle \to \langle f(x) \trianglelefteqslant^{-1} k, x, D\rangle$ **if** $f \neq id$.
$\langle f(x) \trianglelefteqslant c_k(x), x, D\rangle \to \langle f(x) \trianglelefteqslant k, x, D\rangle$.
$\langle c_k(x) \trianglelefteqslant \beta, x, D\rangle \to \langle x \in D, x, D\rangle$ **if** β is ground and $k \trianglelefteqslant \beta$
$\langle c_k(x) \trianglelefteqslant \beta, x, D\rangle \to \langle x \in \emptyset, x, \emptyset\rangle$ **if** β is ground and $k \not\trianglelefteqslant \beta$.

The next rule applies to all functions f that are strictly monotonic, which include $pol_{a,b}$, p_k, pow_{2n+1}, rad_n, log_a and exp_a. For all the mentioned ones except exp_a and rad_n for n even, $f(\mathcal{D}_f) = \mathbb{R}$, so that the

condition $k \in f(\mathcal{D}_f)$ imposes no restriction in such cases. Clearly, f^{-1} must be computable or given in a look-up table. This is true for all these functions.

(StrictMonotonic) *If $D \subseteq \mathcal{D}_{f\circ g}$, $f \neq id$ is strictly monotonic and $k \in f(\mathcal{D}_f)$ then*

$\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle g(x) \trianglelefteqslant f^{-1}(k), x, D\rangle$ **if** $\trianglelefteqslant \in \{=, \neq\}$.
$\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle g(x) \trianglelefteqslant f^{-1}(k), x, D\rangle$ **if** $\trianglelefteqslant \notin \{=, \neq\}$ and f is strictly increasing in $\mathcal{D}_f$.
$\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle g(x) \trianglelefteqslant^{-1} f^{-1}(k), x, D\rangle$ **if** $\trianglelefteqslant \notin \{=, \neq\}$ and f is strictly decreasing in $\mathcal{D}_f$.

Functions f that are symmetric w.r.t. to a vertical line $x = a$ (i.e., $f(a+\delta) = f(a-\delta)$, for all δ such that $a+\delta \in \mathcal{D}_f$) and strictly monotonic in $\mathcal{D}_f \cap \mathcal{S}^a_{\geq}$ make another relevant class.

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

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

$\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle \tilde{\Gamma}(g(x) \in \mathcal{B}^{k'-a}_{a,\trianglelefteqslant}), x, D\rangle$ **if** $\trianglelefteqslant \in \{=, \neq\}$.
$\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle \tilde{\Gamma}(g(x) \in \mathcal{B}^{k'-a}_{a,\trianglelefteqslant}), x, D\rangle$ **if** $f|_{\mathcal{D}_f^{\geq a}}$ is increasing, $\trianglelefteqslant \notin \{=, \neq\}$.
$\langle (f \circ g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle \tilde{\Gamma}(g(x) \in \mathcal{B}^{k'-a}_{a,\trianglelefteqslant^{-1}}), x, D\rangle$ **if** $f|_{\mathcal{D}_f^{\geq a}}$ is decreasing, $\trianglelefteqslant \notin \{=, \neq\}$.

The following five rules may be seen as simpler instances of StrictMonotonic and AxialSymMonotonicBranch.

(AffineTransf)

$\langle (pol_{a,b} \circ f)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle f(x) \trianglelefteqslant (k-b)/a, x, D\rangle$ **if** $k \in \mathbb{R}$ and $\trianglelefteqslant \in \{=, \neq\}$.
$\langle (pol_{a,b} \circ f)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle f(x) \trianglelefteqslant (k-b)/a, x, D\rangle$ **if** $a > 0$, $k \in \mathbb{R}$, $\trianglelefteqslant \notin \{=, \neq\}$.
$\langle (pol_{a,b} \circ f)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle f(x) \trianglelefteqslant^{-1} (k-b)/a, x, D\rangle$ **if** $a < 0$, $k \in \mathbb{R}$, $\trianglelefteqslant \notin \{=, \neq\}$.

(Power)

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

(AbsoluteValue)

$\langle (abs \circ f)(x) \trianglelefteqslant k, x, D\rangle \to \langle \tilde{\Gamma}(f(x) \in \mathcal{B}^k_{0,\trianglelefteqslant}), x, D\rangle$ **if** $k \in \mathbb{R}$ and $k \geq 0$.

(RADIX)

$\langle (rad_n \circ f)(x) \trianglelefteqslant k, x, D\rangle \to \langle f(x) \trianglelefteqslant k^n, x, D\rangle$ **if** n is odd.
$\langle (rad_n \circ f)(x) \trianglelefteqslant k, x, D\rangle \to \langle f(x) \trianglelefteqslant k^n, x, D\rangle$ **if** n is even, $k \geq 0$ and $D \subseteq \mathcal{D}_{rad_n \circ f}$.

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

$\langle (pol_{a,b,c} \circ f)(x) \trianglelefteqslant k, x, D\rangle \to \langle (pol_{a,b,c-k} \circ f)(x) \trianglelefteqslant 0, x, D\rangle$ **if** $0 \neq k \in \mathbb{R}$.
$\langle (pol_{a,b,c} \circ f)(x) \trianglelefteqslant 0, x, D\rangle \to \langle \tilde{\Gamma}(f(x) \in \mathcal{B}^{k'}_{-b/(2a),\trianglelefteqslant}), x, D\rangle$ **if** $\Delta \geq 0$ *and* $a > 0$.
$\langle (pol_{a,b,c} \circ f)(x) \trianglelefteqslant 0, x, D\rangle \to \langle \tilde{\Gamma}(f(x) \in \mathcal{B}^{k'}_{-b/(2a),\trianglelefteqslant^{-1}}), x, D\rangle$ **if** $\Delta \geq 0$ *and* $a < 0$.

We note that if $\Delta < 0$, BOUNDRANGE can be applied taking $\mathcal{E} = pol_{a,b,c}(\mathbb{R}) = [-\frac{\Delta}{4a}, +\infty[$ if $a > 0$ and $\mathcal{E} = pol_{a,b,c}(\mathbb{R}) =]-\infty, -\frac{\Delta}{4a}]$ if $a < 0$, although 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 $a < 0$.

(PIECEWISE) *If* f *is a piecewise function given by* $(f_i, D_i)_{i=1}^n$, *let* $I = \{i \mid D_i \cap D \neq \emptyset\}$. *Let* `substall(g,h,C)` *be a function that replaces all occurrences of subexpression* `g` *by* `h` *in a constraint* `C` *yielding a new constraint. Then*

$\langle C, x, D\rangle \to \langle \vee_{i=1}^n (\texttt{substall}(f \circ g, f_i \circ g, C) \wedge g(x) \in D_i), x, D\rangle$
if $f \circ g$ is a subexpression in C and $g \neq id$.
$\langle C, x, D\rangle \to \langle \vee_{i \in I} (\texttt{substall}(f, f_i, C) \wedge x \in D_i), x, D\rangle$
if f is a subexpression in C and $I \neq \emptyset$.
$\langle C, x, D\rangle \to \langle x \in \emptyset, x, \emptyset\rangle$ **if** f is a subexpression in C and $I = \emptyset$.

(PRODUCTBYCONSTANT) *If* $k' \in \mathbb{R}$

$\langle (c_k \times f)(x) \trianglelefteqslant k', x, D\rangle \to \langle f(x) \trianglelefteqslant k'/k, x, D\rangle$ **if** $0 < k \in \mathbb{R} \setminus \{1\}$.
$\langle (c_k \times f)(x) \trianglelefteqslant k', x, D\rangle \to \langle f(x) \trianglelefteqslant^{-1} k'/k, x, D\rangle$ **if** $0 > k \in \mathbb{R} \setminus \{-1\}$.
$\langle (c_{-1} \times f)(x) \trianglelefteqslant k', x, D\rangle \to \langle f(x) \trianglelefteqslant^{-1} -k', x, D\rangle$.

(DIFFSQUARE)

$\langle (pow_N \circ g - pow_M \circ h)(x) \trianglelefteqslant 0, x, D\rangle \to$
$\langle ((pow_{\frac{N}{2}} \circ g - pow_{\frac{M}{2}} \circ h) \times (pow_{\frac{N}{2}} \circ g + pow_{\frac{N}{2}} \circ h))(x) \trianglelefteqslant 0, x, D\rangle$
if N and M are even, $pow_N \circ g \neq pow_M \circ h$.

(NULLPRODUCT) *If* $D \subseteq \mathcal{D}_{f \odot g}$, $\odot \in \{\times, /\}$ *and* $f, g \neq c_k$ *then*

$\langle (f \odot g)(x) \trianglelefteqslant 0, x, D\rangle \to \langle (f(x) \trianglelefteqslant 0 \wedge g(x) \trianglelefteqslant 0) \vee (f(x) \trianglelefteqslant^{-1} 0 \wedge g(x) \trianglelefteqslant^{-1} 0, x, D\rangle$
if $\trianglelefteqslant \in \{\geq, >\}$
$\langle (f \odot g)(x) \trianglelefteqslant 0, x, D\rangle \to \langle (f(x) \trianglelefteqslant 0 \wedge g(x) \trianglelefteqslant^{-1} 0) \vee (f(x) \trianglelefteqslant^{-1} 0 \wedge g(x) \trianglelefteqslant 0, x, D\rangle$
if $\trianglelefteqslant \in \{\leq, <\}$
$\langle (f \times g)(x) = 0, x, D\rangle \to \langle f(x) = 0 \vee g(x) = 0, x, D\rangle$
$\langle (f \times g)(x) \neq 0, x, D\rangle \to \langle f(x) \neq 0 \wedge g(x) \neq 0, x, D\rangle$
$\langle (f/g)(x) \neq 0, x, D\rangle \to \langle f(x) \neq 0, x, D\rangle$
$\langle (f/g)(x) = 0, x, D\rangle \to \langle f(x) = 0, x, D\rangle$

(FactMonotonic) *If* $D \subseteq \mathcal{D}_{(f\circ g)(x)\trianglelefteqslant(f\circ h)(x)}$, $f \neq id$ *is strictly monotonic and* $g \neq h$,

$\langle (f \circ g)(x) \trianglelefteqslant (f \circ h)(x), x, D\rangle \rightarrow \langle g(x) \trianglelefteqslant h(x), x, D\rangle$ **if** f is strictly increasing.
$\langle (f \circ g)(x) \trianglelefteqslant (f \circ h)(x), x, D\rangle \rightarrow \langle g(x) \trianglelefteqslant^{-1} h(x), x, D\rangle$ **if** f is strictly decreasing.

(FactOdd)

$\langle (f \circ g)(x) \trianglelefteqslant ((-f) \circ h)(x), x, D\rangle \rightarrow \langle (f \circ g)(x) \trianglelefteqslant (f \circ (-h))(x), x, D\rangle$
if $f \neq id$ is an odd function and $g \neq h$.

(ToHomQuocient)

$\langle (f/g)(x) \trianglelefteqslant k, x, D\rangle \rightarrow \langle (f - c_k \times g)/g)(x) \trianglelefteqslant 0, x, D\rangle$ **if** $0 \neq k \in \mathbb{R}$.

(DiffMono) *If* f *is strictly monotonic*

$\langle (f \circ g - f \circ h)(x) \trianglelefteqslant 0, x, D\rangle \rightarrow \langle (f \circ g)(x) \trianglelefteqslant (f \circ h)(x), x, D\rangle$

(SignDiff)

$\langle f(x) \trianglelefteqslant g(x), x, D\rangle \rightarrow \langle (f - g)(x) \trianglelefteqslant 0, x, D\rangle$ **if** neither f nor g are constant.

(SumNull) *If* $f \neq g$ *and both* $f(\mathcal{D}_f)$ *and* $g(\mathcal{D}_g)$ *are in* $\mathbb{R}_0^-$ *or both* $f(\mathcal{D}_f)$ *and* $g(\mathcal{D}_g)$ *are in* $\mathbb{R}_0^+$, *then*

$\langle (f + g)(x) = 0, x, D\rangle \rightarrow \langle (f(x) = 0 \wedge g(x) = 0, x, D\rangle$
$\langle (f + g)(x) \neq 0, x, D\rangle \rightarrow \langle (f(x) \neq 0 \vee g(x) \neq 0, x, D\rangle$

(SquarePol)

$\langle (rad_2 \circ f - g)(x) \trianglelefteqslant 0, x, D\rangle \rightarrow \langle (rad_2 \circ f)(x) \trianglelefteqslant g(x), x, D\rangle$
$\langle (rad_2 \circ f)(x) \trianglelefteqslant g(x), x, D\rangle \rightarrow \langle (f(x) \trianglelefteqslant (pow_2 \circ g)(x) \wedge g(x) \geq 0, x, D\rangle$

We need the last rules to guarantee the solvers completeness for the expressions that Demomath creates. The solvers will not support user-defined expressions $f(x)$, unless they may be *recognized* by the system. Simple algebraic manipulations may be carried out to express f in terms of a different combination of primitive functions. A distinct term-rewriting system is introduced to perform trivial reductions ($c_1 \times f \equiv f$, $c_0 \times f \equiv c_0$, $c_{-1} \times f \equiv -f$, ...) operate polynomials, replace abs ($abs \equiv ((id, \mathbb{R}_0^+), (-id, \mathbb{R}^-))$), lift rad_n and pow_n ($(c_k \times pow_{2n+1}) \circ f \equiv pow_{2n+1} \circ (c\ \sqrt[2n+1]{k} \times f)$, $(c_k \times rad_{2n}) \circ f \equiv sign(k) rad_{2n} \circ (c_{|k|^n} \times f)$, ... where $sign(k)$ is the signal of k), reduce to a common denominator ($f_1/g_1 \pm f_2/g_2 \equiv (f_1 \times \tilde{g}_2 \pm f_2 \times \tilde{g}_1)/lcm(g_2, g_1)$, if $\tilde{g}_i = lcm(g_2, g_1)/g_i$, for $i = 1, 2$, and $lcm(g_2, g_1)$ is a common multiple of g_1 and g_2, possibly $g_2 \times g_1$), etc.

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 solvers based on the proposed rewrite rules set fulfils this requirement. It is quite generic, for it takes advantage of a declarative approach. The system is being implemented in Prolog. We are going to tackle rewrite rules annotations to provide explanations in natural language, following ideas of [10].

References

1. H. Barendregt. Towards an Interactive Mathematical Proof Language. in F. Kamareddine (ed.), *Thirty Five Years of Automath*, Kluwer (2003) 25-36.
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. Beeson. MathXpert: un logiciel pour aider les élèves à apprendre les mathématiques par l'action. *Sciences et Techniques Educatives* 9:1-2 (2002).
4. P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and M. Vittek. Elan: A logical framework based on computational systems. *Electr. Notes Theor. Comput. Sci.*, 4 (1996).
5. A. Bundy. *The Computer Modelling of Mathematical Reasoning*. A. Press (1983).
6. H. Cirstea, C. Kirchner, L. Liquori, and B. Wack. Rewrite strategies in the rewriting calculus. *Electr. Notes Theor. Comput. Sci.*, 86:4 (2003).
7. H. Gottliebsen, T. Kelsey, U. Martin. Hidden Verification for Computational Mathematics. *J. Symbolic Computation* 39 (2005) 539-567.
8. C. Holzbaur. OFAI clp(q,r) Manual, Edition 1.3.3. Austrian Research Institute for Artificial Intelligence, TR-95-09, Vienna (1995).
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. R. Ravaglia, T. Alper, M. Rozenfeld, P. Suppes. Successful Pedagogical Applications of Symbolic Computation. In N. Kajler (ed.), *Computer-Human Interaction in Symbolic Computation*, Texts and Monographs in Symbolic Computation, Springer-Verlag (1998).
12. A. Robinson, A. Voronkoy (Eds). *Handbook of Automated Reasoning*, Elsevier Science (2001).
13. 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.
14. G. Xiao. WIMS – An Interactive Mathematics Server, *Journal of Online Mathematics and its Applications* 1, MAA (2001). (`http://wims.unice.fr`)