A Note on Formalising Undefined Terms in Real Analysis

Claudio Sacerdoti Coen[1,2] Enrico Zoli[1,3]

Department of Computer Science
University of Bologna
Bologna, Italy

Abstract

To adopt proof assistants based on logic of total functions in education, one major problem is that of representing partial functions. In particular, one wishes to capture undefinedness in a rigorous way, while staying as close as possible to traditional mathematical practice. In this paper, we propose to represent potentially undefined terms with partial setoids on lifted terms, and to understand book equalities occurring in equality chains as special directed relations that hold only under assumptions of definedness for some terms. We also employ a suitable notion of strict morphism to fully automate propagation of these directed relations in strict contexts.

Keywords: undefinedness, strictness, proof assistant, Real Analysis, education, Matita

1 Introduction

DAMA (Dimostrazione Assistita per la Matematica e l'Apprendimento) is a recently started locally funded project of the University of Bologna aimed at the development of a self-assessment tool for students of a first Real Analysis course. The tool will be able to check the correctness of proofs and provide hints or counterexamples. It should also force the student to prove every necessary side condition in computational exercises.

The tool, still under development, will be based on the core of the Matita interactive theorem prover [1], also under development by the research group of Prof. Asperti at the University of Bologna. Matita is based on the Calculus of (Co)Induction Constructions, a logic of total functions. Since partial functions play a major role in Real Analysis, the first problem we have to address is that of the representation of partial functions.

[1] Partially supported by the strategic project DAMA (Dimostrazione Assistita per la Matematica e l'Apprendimento) of the University of Bologna.

[2] sacerdot@cs.unibo.it

[3] zoli@math.unifi.it

Several papers have already been devoted to the subject. Müller and Slind in [6] provide a good comparison of the techniques proposed and give many useful references. However, in didactics we cannot adopt those solutions that are far from the usual mathematical practice. The latter, called in [4] the *traditional approach to undefinedness*, can be summarized as: 1) atomic terms (variables and constants) are always defined; 2) a function application is undefined either when the argument or the function is undefined or when the function is undefined on the argument; 3) formulas are always defined, and are false when undefined terms occur within them.

Among the proposals close to the usual mathematical practice, Farmer's [4] is certainly very interesting, but it requires an ad-hoc logic with undefinedness. A similar remark applies to other proposals requiring ad-hoc logics, e.g. [7]. Since we do not plan to change the logic of Matita, we cannot adopt these solutions. In this paper, we therefore encode partial functions as total functions on lifted terms. To hide the details of this formalisation from the student, we exploit an automation tactic that deal with partial setoids [8]. Indeed, lifted terms naturally form a partial setoid, when equipped with the partial equivalence relation that coincides with equality on defined terms, and is false otherwise.

The main contribution of the paper is the observation that there is more than reasoning on partial setoids in the traditional handling of equality chains over potentially undefined terms. Indeed, in equality chains some steps also reduce the proof of definedness of one side to that of the other. Hence we need to extend automation to fully capture this form of reasoning.

We claim that in practice our approach is suitable for didactics. However, it is still to be completely automated and tested extensively on students.

To be more concrete, consider the following running example as it usually appears in textbooks.

Theorem 1.1 *Suppose that x is a real number with $|x| < 1$. Then $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$.*

Proof. $\sum_{i=0}^{n} x^i = \frac{1-x^{n+1}}{1-x}$ by an easy induction over n. Thus

$$\begin{aligned}
\sum_{n=0}^{\infty} x^n &= \lim_{n\to\infty} \sum_{i=0}^{n} x^i \\
&= \lim_{n\to\infty} \frac{1-x^{n+1}}{1-x} \\
&= \frac{\lim_{n\to\infty}(1-x^{n+1})}{1-x} \\
&= \frac{1-\lim_{n\to\infty} x^{n+1}}{1-x} \\
&= \frac{1}{1-x}.
\end{aligned}$$

□

In our opinion, this exercise is interesting for it consists in two distinct parts: the first is a simple proof by induction that exercises proving skills; the second part is an easy computational proof where the main difficulty (completely hidden in the text) is to check that every expression is well-defined. Note that in the given proof

no rewriting step is explicitly justified at all. Therefore, in our judgement, it is too lax for an exercise solution, since we expect students to show at least where the hypotheses are employed.

The first question about this exercise we should pose ourselves concerns our teaching goals when proposing it to the student. We identify some alternatives in Sect. 2, and give a rigorous account of the justification of each proof step. Formalisation in a logic of total functions is discussed in Sect. 3. In the same section we also discuss how to provide automation to hide the intricacies of the formalization, allowing natural manipulation of formulas. Automation is achieved by coupling the ideas in [8] with the notion of *strict morphism*, i.e., a function that preserves equalities and undefinedness. Conclusions follow in Sect. 4.

2 Teaching goals

In principle, proof assistants can become a useful aid for self-assessment of proving skills. However, the gap between the usual pen&paper mathematical practice and the formalised mathematics these systems understand is significant and difficult to fill. Two phenomena yield this gap: the lack of absolute rigour in the usual manipulation of formulas; the limitations of formal systems, which render formalisation difficult and formalised proofs obscure. We postpone discussion on the second phenomenon to Sect. 3.

The first phenomenon is already evident in our running example where we manipulate limits of sequences before proving convergence of the sequences. Similarly, it often happens to compute derivatives of functions before proving their differentiability. Admittedly, an excess of rigour may lead to drown the essence of a proof in pedantic details. However, we argue that a first Real Analysis course for students in Mathematics or Computer Science should both frame rigorously their minds — our first teaching goal — and teach effective high-level formulas manipulation — our second teaching goal. The two teaching goals may require different sets of exercises (or different approaches to them).

For instance, our running example could be proposed as an exercise twice. The first time, just after the introduction of the "tends to" relation, where no partial operator is employed and each limit must explicitly be shown (or assumed) to exist. The expected solution to the exercise would be the following:

Theorem 2.1
Suppose that x is a real number with $|x| < 1$. Then $\sum_{i=0}^{n} x^i \xrightarrow[n\to\infty]{} \frac{1}{1-x}$.

Proof.
We prove $\sum_{i=0}^{n} x^i = \frac{1-x^{n+1}}{1-x}$ by induction over n.
...
Moreover $x^{n+1} \xrightarrow[n\to\infty]{} 0$ since $|x| < 1$.
Hence $1 - x^{n+1} \xrightarrow[n\to\infty]{} 1$.
Hence $\frac{1-x^{n+1}}{1-x} \xrightarrow[n\to\infty]{} \frac{1}{1-x}$ as $1 - x \neq 0$.
Hence the thesis. □

Note that, with respect to Theorem 1.1, the equality chain has been replaced

with a chain of implications. Moreover, the two chains proceed in opposite order. The chosen order appears more natural in this context, because we need to rigorously prove the existence of limits before computing with them. The two proofs are different in an even more radical sense: in the original proof we can see formulas, such as $\frac{1-\lim_{n\to\infty} x^{n+1}}{1-x}$, that have no exact counterpart in the new proof. Note also the need to explicitly prove denominators to be different from 0, because of the lack of undefined terms (which is consistent with the choice of avoiding the partial limit operator).

It is possible to follow the original proof a little more closely by adopting the original order of the chain. However, the solution looks rather artificial:

Proof.
...
We have $1 - x \neq 0$ since $|x| < 1$.
We need to prove $\frac{1-x^{n+1}}{1-x} \xrightarrow[n\to\infty]{} \frac{1}{1-x}$.
It is sufficient to prove $1 - x^{n+1} \xrightarrow[n\to\infty]{} 1$.
It is sufficient to prove $x^{n+1} \xrightarrow[n\to\infty]{} 0$.
The thesis follows from $|x| < 1$. □

To reach our second teaching goal, we could propose the same exercise a second time just after the introduction of the limit operator. This time the emphasis is on exercising the students on effective limit computation, where (usually) as many side conditions as possible are avoided. Thus, we look for a solution quite adherent to that of Theorem 1.1. In particular, we admit the existence of undefined terms to be handled according to the traditional approach discussed in the introduction.

In Theorem 1.1, the definedness of each expression containing a partial operator (a limit or a division) is not justified. Consistently with our first teaching goal, we would like to avoid such a lax approach, but we look for lightweight justifications. We propose to make sense of the proof by assuming that, at each step in the equation chain, ground for the definedness of the left hand side is derived by the assumption that the right hand side will be proved to be defined later on. This property does not always hold in general, unless side conditions are required. E.g., if $\frac{ax}{a}$ is defined, then so is x and $x = \frac{ax}{a}$; but the definedness of x yields neither that of $\frac{ax}{a}$ nor the equality $x = \frac{ax}{a}$. The latter property holds when the additional hypothesis $a \neq 0$ is assumed. Thus, replacing $\frac{ax}{x}$ with x in an equality chain or the other way around yields different side conditions; this suggests that, when terms can possibly be undefined, rewriting chains are not symmetric. As far as we know, this phenomenon has not extensively been studied in the literature. We propose to understand it rigorously by introducing two oriented relations: $\overset{\triangleright}{=}$ and its dual $\overset{\triangleleft}{=}$.

Definition 2.2
$x \overset{\triangleright}{=} y$ when x is defined if y is, and in this case they are equal.
$x \overset{\triangleleft}{=} y$ when $y \overset{\triangleright}{=} x$ or equivalently when y is defined if x is, and in this case they are equal.

These relations are oriented variants of Farmer's *quasi equality* relation: x is quasi equal to y when x and y are either both undefined or equal. Quasi equality

can be reduced to our definitions: x is quasi equal to y if $x \stackrel{\rhd}{=} y$ and $x \stackrel{\lhd}{=} y$. Therefore we call our relations *directed quasi equalities*.

Our running example can now be rigorously justified using $\stackrel{\rhd}{=}$:

Proof.
...

$$\begin{aligned}
\sum_{n=0}^{\infty} x^n &\stackrel{\text{def}}{=} \lim_{n\to\infty} \sum_{i=0}^{n} x^i \\
&\stackrel{\rhd}{=} \lim_{n\to\infty} \frac{1-x^{n+1}}{1-x} \\
&\stackrel{\rhd}{=} \frac{\lim_{n\to\infty}(1-x^{n+1})}{1-x} && (1)\\
&\stackrel{\rhd}{=} \frac{1-\lim_{n\to\infty} x^{n+1}}{1-x} && (2)\\
&\stackrel{\rhd}{=} \frac{1}{1-x} \text{ since } |x|<1 && (3)\\
&\quad \text{which is defined since } |x|<1.
\end{aligned}$$

□

It is to be noted that, since the existence of each expression is reduced by $\stackrel{\rhd}{=}$ to the definedness of $\frac{1}{1-x}$, in the final step we have to show this term to be defined in order to conclude $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$. Without this justification, the final conclusion of the equality chain would become (the weaker) $\sum_{n=0}^{\infty} x^n \stackrel{\rhd}{=} \frac{1}{1-x}$.

In general, given an equality chain

$$\ldots \stackrel{\rhd}{=} M_1 \stackrel{\lhd}{=} E_1 \stackrel{\lhd}{=} \ldots \stackrel{\lhd}{=} E_n \stackrel{\lhd}{=} T_1 \stackrel{\rhd}{=} E_{n+1} \stackrel{\rhd}{=} \ldots \stackrel{\rhd}{=} E_m \stackrel{\rhd}{=} M_2 \stackrel{\lhd}{=} E_{m+1} \stackrel{\lhd}{=} \ldots,$$

to conclude that all terms in the chain are equal (and thus defined), we have to prove that every term M_i is defined. This done, we recursively obtain that any other term in the equality chain is defined and, ultimately, that all terms are equal.

More formally, an equality chain fragment $E_1 \ \mathcal{R} \ E_2 \ \mathcal{R}' \ E_3$ must be always understood as an application of an ad-hoc generalized transitivity principle $\forall x, y, z. x \ \mathcal{R} \ y \wedge y \ \mathcal{R}' \ z \Rightarrow x \ \mathcal{S} \ z$, where the relation S depends on R and R' and is usually the strongest of them. The following generalized transitivity principles hold for our directed quasi equalities:

Lemma 2.3 *For x, y, z potentially undefined terms*

$$x \stackrel{\triangleright}{=} y \wedge y \stackrel{\triangleright}{=} z \Rightarrow x \stackrel{\triangleright}{=} z$$
$$x \stackrel{\triangleleft}{=} y \wedge y \stackrel{\triangleleft}{=} z \Rightarrow x \stackrel{\triangleleft}{=} z$$
$$x = y \wedge y \stackrel{\triangleleft}{=} z \Rightarrow x = z$$
$$x \stackrel{\triangleright}{=} y \wedge y = z \Rightarrow x = z$$
$$x \stackrel{\triangleright}{=} y \wedge y \stackrel{\triangleleft}{=} z \Rightarrow x = z \quad \textit{when } y \textit{ is defined}$$
$$x \stackrel{\triangleleft}{=} y \wedge y \stackrel{\triangleright}{=} z \Rightarrow x = z \quad \textit{when } x \textit{ and } z \textit{ are defined}$$

The proof of this lemma is trivial (just recall that in the traditional approach two terms are equal only when they are both defined and have the same value).

As another example that employs an equality chain, let us look at the corresponding proof obtained by reversing the direction of the chain:

Proof.
...
Since $|x| < 1$ the expression $\frac{1}{1-x}$ is defined. Hence

$$\begin{aligned}
\frac{1}{1-x} &\stackrel{\triangleleft}{=} \frac{1 - \lim_{n\to\infty} x^{n+1}}{1-x} \text{ since } |x| < 1 \\
&\stackrel{\triangleleft}{=} \frac{\lim_{n\to\infty}(1 - x^{n+1})}{1-x} \\
&\stackrel{\triangleleft}{=} \lim_{n\to\infty} \frac{1 - x^{n+1}}{1-x} \\
&\stackrel{\triangleleft}{=} \lim_{n\to\infty} \sum_{i=0}^{n} x^i \\
&\stackrel{\text{def}}{=} \sum_{n=0}^{\infty} x^n
\end{aligned}$$

□

This time, the expression that is directly shown to be defined is the leftmost end of the chain.

Observe that the corresponding rewriting steps in the direct and backward proof are justified by exactly the same lemma, since $x \stackrel{\triangleright}{=} y$ is equivalent to $y \stackrel{\triangleleft}{=} x$ (because of the symmetry of the equality). For instance, both (1) $\stackrel{\triangleright}{=}$(2) and (2) $\stackrel{\triangleleft}{=}$(1) are justified by

Lemma 2.4 *For all c and all real sequences $f(n)$,*

$$\lim_{n\to\infty}(c - f(n)) \stackrel{\triangleright}{=} c - \lim_{n\to\infty} f(n)$$

The lemma used to justify (2) $\stackrel{\triangleright}{=}$(3) as well as (3) $\stackrel{\triangleleft}{=}$(2) is

Lemma 2.5 *If $|x| < 1$, then $\lim_{n\to\infty} x^n \stackrel{\triangleright}{=} 0$.*

Note that the hypothesis $|x| < 1$ is to be explicitly justified in the equality chain by a side proof. Incidentally, the following corresponding "dual" lemma (that could justify (2) $\stackrel{\lhd}{=}$(3) or (3) $\stackrel{\rhd}{=}$(2)) has no extra hypotheses, as the definedness of $\lim_{n\to\infty} x^n$ already gives $|x| < 1$.

Lemma 2.6 $0 \stackrel{\rhd}{=} \lim_{n\to\infty} x^n$.

Finally, we would like to spend a few additional words on teaching goals. So far, we have recognized two teaching goals in adopting proof assistants and types in mathematical education. The first one is the emphasis on mathematical rigour, so as to augment both the comprehension of the role played by axioms and hypotheses in inferences and the awareness of common pitfalls in mathematical reasoning. The second one is the enhancement of skills in mathematical practice, where rigour is not forced. In both cases it is important for a didactic tool to hide all the difficulties arising from the formalisation process. In particular, we expect the system to provide enough automation to completely hide from the student the side proofs arising from technical side conditions. These are justifications we do not expect to see on paper, independently from the teaching goal.

We can also identify a third alternative goal, i.e., teaching (mathematical) formalisation techniques, which are largely independent from the mathematical subject being formalized, and definitely outside standard curricula in Mathematics and Computer Science. For this reason we will not address this subject any longer.

The exercise dealing with the "tends to" relation can be straightforwardly formalised in a logic of total functions, provided that division requires either a proof that the quotient is not 0 or a formalisation via alternative techniques such as, e.g., Σ-types. The one that deals with the limit operator requires a preliminary encoding of undefined terms and operators, and a substantial amount of automation to hide the intricacies of this encoding. This is the subject of the next section.

3 Formalisation

Plenty of functions and operators in elementary calculus are partial. The following are examples of undefined expressions: division by zero; supremum of an unbounded set; limit of an oscillating sequence; value of a non-converging series; derivative of a discontinuous function; definite integral of a non-integrable function.

The usual "lax" mathematical practice manipulates partial expressions in a way that is often difficult to understand rigorously. In Sect. 1 we have partially discussed the issue by sticking to what Farmer calls "the traditional approach to undefinedness", enriched by two new directed quasi equality relations $\stackrel{\rhd}{=}$ and $\stackrel{\lhd}{=}$ that replace book equality in chains of equations. However, we have kept the description at an informal level. We now address the problem of capturing these ideas in a formal system.

In the unlikely case that our system is based on a logic with undefinedness [4] or a three value logic [7], the formalisation proceeds very smoothly: the logic is already aware of undefined terms, equality is already the proposed relation, $\stackrel{\rhd}{=}$ and $\stackrel{\lhd}{=}$ can be directly defined and appropriate replacement principles are likely to be

derivable.

When the logic is a logic of totally defined terms, we need a way to represent undefinedness and we need to change the equality (and every other predicate) to behave correctly over undefined terms. Since equality is replaced by a coarser relation, it is no longer true that $x = y \Rightarrow C[x] \iff C[y]$ for each context C. In order to keep term manipulation simple, the system is now in charge of automatically proving the side conditions that make the previous implication true for a particular C. In [8], the first author has given the Coq proof assistant this explicit support in the case of generalized setoids, including partial setoids. Partial setoids can be exploited in our context, since the well-known representation of potentially undefined terms of type T as elements of the lifted type $T_\perp \stackrel{\text{def}}{=} T \oplus \{\perp\}$ gives rise to a partial setoid.

Even when partial setoids automation is exploited, the user is still requested to prove many side conditions that are implicitly discharged in the mathematical practice, and that we plan to avoid basing our equality chains on $\stackrel{\rhd}{=}$ and $\stackrel{\lhd}{=}$. Hence, we now provide automation for $\stackrel{\rhd}{=}$ and $\stackrel{\lhd}{=}$. Let us recall the notion of partial setoid and the associated morphisms.

A *partial setoid* is a couple $(T, \simeq_T)$, where T is a type and $\simeq_T$ is a symmetric and transitive relation over T. An element x of T is said to be *proper* when $x \simeq_T x$. We will also write $x \downarrow^T$ for $x \simeq_T x$. We drop the subscript/superscript T when the partial equality relation is clear from the context.

A *morphism* of partial setoids $(S, \simeq_S), (T, \simeq_T)$ is a function $f : S \to T$ such that $f(x) \simeq_T f(y)$ whenever $x \simeq_S y$. We will write $f : (S, \simeq_S) \Rightarrow (T, \simeq_T)$ for f being a morphism between $(S, \simeq_S)$ and $(T, \simeq_T)$. With a little notational abuse, we will say that f has type $(S, \simeq_S) \Rightarrow (T, \simeq_T)$.

If $(S, \simeq_S)$ and $(T, \simeq_T)$ are partial setoids, so is $(S, \simeq_S) \times (T, \simeq_T) \stackrel{\text{def}}{=} (S \times T, \simeq_S \times \simeq_T)$ (the cartesian product equipped with the partial equivalence relation induced by $\simeq_S$ and $\simeq_T$ acting componentwise). The type **Prop** of propositions together with coimplication forms the (total) setoid (**Prop**, $\iff$).

Alternative formulations of the notion of partial setoids are explored in [3]. The one we adopt is that of Bailey [2].

For each type T, we represent potentially undefined terms of T with elements of the lifted type $T_\perp$. We equip $T_\perp$ with a partial setoid structure by defining the partial equivalence relation $\simeq$ as $x \simeq y$ iff $x, y \neq \perp$ and $x = y$ as elements of T. According to this definition, proper elements are defined elements and we can read $x \downarrow$ as "x is defined". Moreover, from $x \simeq y$ we immediately know $x \downarrow$ and $y \downarrow$ since for no x, not even $\perp$, we have $\perp \simeq x$.

The reader acquainted with the work of Farmer should note that our notation is not consistent with [4]: he denotes our relation $\simeq$ with $=$, whereas we reserve the symbol $=$ for either Leibniz's or book equality; moreover, he uses $\simeq$ for quasi equality, whereas we only have special symbols for directed quasi equalities. Finally, in Farmer's logic with undefinedness all atoms are implicitly defined, while our variables are always quantified over $T_\perp$ for some T. Thus, according to the terminology used by Feferman in [5], Farmer's logic is a logic of definedness, while we are encoding a logic of existence.

The following facts over morphisms are exploited in manual and automatic proofs over setoids. The identity function over S is always a morphism of type $(S, \simeq) \Rightarrow (S, \simeq')$ whenever $\simeq$ and $\simeq'$ are partial equivalence relations over S such that the graph of $\simeq$ is included in the graph of $\simeq'$ (i.e. $x \simeq x'$ implies $x \simeq' x'$ for each x, x'). The constant function $f(x) \stackrel{\text{def}}{=} c$ of type $S \to T$ is a morphism of type $(S, \simeq_S) \Rightarrow (T, \simeq_T)$ iff $c \downarrow^T$. The function $\mathcal{C}_f(x) \stackrel{\text{def}}{=} f(x, x)$ is a morphism of type $(S, \simeq_S) \Rightarrow (T, \simeq_T)$ whenever f is a morphism of type $(S, \simeq_S) \times (S, \simeq_S) \Rightarrow (T, \simeq_T)$. Composition of morphisms is a morphism. Finally, any partial equivalence relation $\simeq$ over S is a morphism of type $(S, \simeq) \times (S, \simeq) \Rightarrow (\mathbf{Prop}, \iff)$, since it is symmetric and transitive (proof given in [8]).

Let us now consider how to formalise our running example in this setting. Subtraction between real numbers is represented as a morphism of type $(\mathbb{R}_\perp, \simeq) \Rightarrow (\mathbb{R}_\perp, \simeq)$; exponentiation as a morphism of type $(\mathbb{R}_\perp, \simeq) \Rightarrow (\mathbb{N}, =) \Rightarrow (\mathbb{R}_\perp, \simeq)$; the limit operator as a morphism of type $(\mathbb{N} \to \mathbb{R}_\perp, \simeq_\to)$, where $f \simeq_\to g$ when $f(n) \simeq g(n)$ for each n; finally, division is represented as a morphism of type $(\mathbb{R}_\perp, \simeq) \times (\mathbb{R}_\perp, \simeq_0) \Rightarrow (\mathbb{R}_\perp, \simeq)$ where $x \simeq_0 y$ iff $x \simeq y$ and $x \neq 0$. Note that the identity function over real numbers is a morphism from $(\mathbb{R}, \simeq_0)$ to $(\mathbb{R}, \simeq)$; hence, we can compose a morphism of type $(\mathbb{R}, \simeq) \Rightarrow (S, \simeq_S)$ with a morphism of type $(T, \simeq_T) \Rightarrow (\mathbb{R}, \simeq_0)$. Note also that it is impossible to prove division to be a morphism of type $(\mathbb{R}_\perp, \simeq) \times (\mathbb{R}_\perp, \simeq) \Rightarrow (\mathbb{R}_\perp, \simeq)$, as $1 \simeq 1$ and $0 \simeq 0$ but $1/0$ is $\perp$ and it is not true[4] that $\perp \simeq \perp$.

Consider the third equality in our running example:

$$\frac{\lim_{n\to\infty}(1 - x^{n+1})}{1 - x} \stackrel{\trianglerighteq}{=} \frac{1 - \lim_{n\to\infty} x^{n+1}}{1 - x}$$

and suppose that Lemma 2.4 has already been proved. To justify the equality we need to contextualise (an instance of) the lemma to the context $C(w) \stackrel{\text{def}}{=} \frac{w}{1-x}$. In other words, we need to prove

$$\lim_{n\to\infty}(1 - x^{n+1}) \stackrel{\trianglerighteq}{=} 1 - \lim_{n\to\infty} x^{n+1} \Rightarrow \frac{\lim_{n\to\infty}(1 - x^{n+1})}{1 - x} \stackrel{\trianglerighteq}{=} \frac{1 - \lim_{n\to\infty} x^{n+1}}{1 - x}$$

or, more generally, that $x \stackrel{\trianglerighteq}{=} y \Rightarrow (C(x) \Rightarrow C(y))$. Adopting the terminology of [8], we need to prove that $C(\cdot)$ is a covariant morphism from the partial and asymmetric setoid $(\mathbb{R}, \stackrel{\trianglerighteq}{=})$ to the asymmetric setoid $(\mathbf{Prop}, \Rightarrow)$. This can be done, even automatically, exploiting all the lemmas in [8]. However, to our knowledge Coq is the only system that implements automation over "generalised" setoids (i.e., setoids whose relation is not required to be reflexive, symmetrical or transitive). Moreover, when asymmetric setoids are employed, user-provided proofs that basic morphisms are indeed morphisms become more involved. Therefore, we now propose a simpler alternative automation technique that does not require asymmetric setoids.

To avoid the latter, we can expand the definition of $\stackrel{\trianglerighteq}{=}$ both in the lemma and

[4] We do not write $\perp \not\simeq \perp$ since that latter symbol is better understood as *book diversity*, i.e., $x \not\simeq y$ iff x and y are both defined and different. Thus, it is not true that $\perp \not\simeq \perp$.

in the equality to be justified; they become, respectively,

$$\forall c, f.(c - \lim_{n\to\infty}) \downarrow \Rightarrow \lim_{n\to\infty}(c - f(n)) \simeq c - \lim_{n\to\infty} f(n)$$

$$\frac{1 - \lim_{n\to\infty} x^{n+1}}{1-x} \downarrow \Rightarrow \frac{\lim_{n\to\infty}(1 - x^{n+1})}{1-x} \simeq \frac{1 - \lim_{n\to\infty} x^{n+1}}{1-x}$$

The equality can now be justified, thanks to the lemma and the definition of a morphism, by assuming (H) $\frac{1-\lim_{n\to\infty} x^{n+1}}{1-x} \downarrow$ and showing:

(i) $(1 - x) \downarrow$ and $(1 - \lim_{n\to\infty} x^{n+1}) \downarrow$. Both are consequences of the assumption (H). The second proof is used to discharge the hypothesis of the lemma.

(ii) $C(w) \stackrel{\text{def}}{=} \frac{w}{1-x}$ is a morphism of type $(\mathbb{R}_\perp, \simeq) \Rightarrow (\mathbb{R}_\perp, \simeq)$.
This is of course the case, for $id(w) \stackrel{\text{def}}{=} w$ is an identity morphism, $c(w) \stackrel{\text{def}}{=} 1 - x$ is a constant morphism returning the element $1 - x$ (which is defined because of (i)), division is a morphism, and $C(w)$ is a composition of division, $c(\cdot)$ and the identity morphism.

When implementing automation over partial setoids as explained in [8], (ii) is proved automatically by the system. Hence we only need to provide automation for (i). Now, (i) holds since we can prove that $\forall n, d.\frac{n}{d} \downarrow \Rightarrow n \downarrow \wedge d \downarrow$. In other words, if the application of division to some arguments yields a defined term, then the arguments are defined as well. This is a general property of morphisms and predicates in the traditional approach to undefinedness, and it can be recognized to be just *strictness* (see, for instance, [5]):

Definition 3.1
A morphism $f : (S_\perp, \simeq_S) \Rightarrow (T_\perp, \simeq_T)$ is *strict* if $f(x) \downarrow^T$ implies $x \downarrow^S$.
A morphism $P : (S_\perp, \simeq) \Rightarrow (\mathbf{Prop}, \iff)$ is strict if $P(x)$ implies $x \downarrow$.

The partial equivalence relation $\simeq_S$ over $S_\perp$ is strict for each type S. Moreover, all morphisms in our running example are strict. For instance, if $x - y$ is defined then both x and y are.

The following facts help showing that a predicate is strict and can be easily exploited in a tactic that proves syntactic composition of morphisms to be strict. The identity morphism is strict, whereas constant morphisms are not. Composition of strict morphisms yields a strict morphism. The morphism $f(x) \stackrel{\text{def}}{=} (g(x), h(x))$ is strict on x if (at least) one of $g(x)$ and $h(x)$ is. The morphism $P(x) \vee Q(x)$ on x is strict if $P(x)$ and $Q(x)$ are; $P(x) \wedge Q(x)$ is strict on x if $P(x)$ is strict or if $Q(x)$ is; $\exists y.P(x, y)$ is strict on x whenever $P(x, y)$ is strict on x for each y; moreover, $\forall y.P(x, y)$ is strict on x if there exists a y such that $P(x, y)$ is strict on x. There is no relation between the strictness of $P(x)$ and that of $\neg P(x)$. In particular, in the traditional approach to undefinedness all predicates over undefined terms are false, so that the negative form of a predicate $P(x)$ is not the logical negation of P, but the predicate "$x \downarrow \wedge \neg P(x)$". In turn, we obtain the following restriction over implication: the predicate $P(x) \Rightarrow Q(x)$ is strict on x if $Q(x)$ is strict on x and P is constant over x. The latter check is necessary, as the following counterexample shows: $x \simeq x \Rightarrow x \simeq x$ always holds, but it does not imply $x \downarrow$.

Similarly to what the first author did in [8], we can easily implement a semi-reflexive tactic to prove automatically when a morphism $P : (S, \simeq) \Rightarrow (\mathbf{Prop}, \iff)$, that is a syntactic composition of strict and non-strict morphisms, is strict in one argument. The tactic is based on the lemmas above.

Exploiting such a tactic, we can specialise [8] (or its restriction to symmetric and transitive setoids) to automatically prove, for a context P that is a suitable syntactic composition of strict and non strict morphisms,

$$P(x) \Rightarrow (y \stackrel{\rhd}{=} x) \Rightarrow P(y)$$

Coming back to our running example, we can now use automation to reduce, for instance, a proof of $\frac{\lim_{n\to\infty}(1-x^{n+1})}{1-x} \simeq \frac{1}{1-x}$ to a proof of $\frac{1-\lim_{n\to\infty} x^{n+1}}{1-x} \simeq \frac{1}{1-x}$ by means of a proof of $\lim_{n\to\infty}(1 - x^{n+1}) \stackrel{\rhd}{=} 1 - \lim_{n\to\infty} x^{n+1}$ and no other side condition.

For the sake of generality, we remark that strict morphisms are a special case of *co-guarded morphisms*.

Definition 3.2
A morphism $f : (S_\perp, \simeq_S) \Rightarrow (T_\perp, \simeq_T)$ is *Q-co-guarded* if $f(x) \downarrow^T \Rightarrow Q(x)$.

Strict morphisms are $\downarrow$-co-guarded morphisms. Note that for a strict morphism $P : (S, \simeq) \Rightarrow (\mathbf{Prop}, \iff)$ the condition $P(x) \downarrow \iff$ can be omitted since it is not informative.

For a Q-co-guarded strict morphism P over **Prop**, we can specialise [8] to automate proofs of

$$P(x) \Rightarrow (Q(x) \Rightarrow y \simeq x) \Rightarrow P(y)$$

So far, the only interesting class of co-guarded morphisms we currently know is that of strict morphisms.

To summarise, we suggest how to formalise chains of directed quasi equalities by automating "directed rewritings" justified by lemmas of the form $\forall \bar{x}, P(\bar{x}) \Rightarrow F(\bar{x}) \stackrel{\rhd}{=} F'(\bar{x})$, being $\bar{x}$ a vector of variables and the extra hypotheses $P(\cdot)$ justified as side conditions in the chain. Since $F(\bar{x}) \stackrel{\rhd}{=} F'(\bar{x})$ is equivalent to $F'(\bar{x}) \downarrow \Rightarrow F(\bar{x}) \simeq F'(\bar{x})$, a number of lemmas can exploit the hypothesis $F'(\bar{x}) \downarrow$ to drop many extra conditions such as the definedness of quantified variables occurring in $F'(\bar{x})$.

For instance, in our running examples all lemmas can be given in this form and only Lemma 2.5 requires an extra hypothesis. As a comparison, corresponding lemmas in Farmer's approach [4] would also miss the additional hypotheses, but for a different reason: since his logic is a logic of definedness, all quantified variables are implicitly assumed to be defined.

We finally add that we have not discussed two other simple and well-known approaches to undefined terms (not far away from the usual mathematical practice, but too lax according to our first teaching goal). Contrary to the traditional approach to undefinedness, in both approaches some dubious properties can hold even for undefined terms. The first approach extends every partial function and operation to the whole domain by assigning an arbitrary value outside the original

domain. For instance, $1/0$ becomes a perfectly valid real number, say 17, that enjoys the property $(1/0)/17 = 1$. The approach is often used in HOL, Isabelle and Mizar, and always in ACL2. The second approach extends axiomatically every partial function and operation to the whole domain, but this value remains unknown by prefixing each axiom of the theory with hypotheses that avoid characterising the partial functions outside their domain of definitions. For instance, it is true that $1/0$ is a perfectly valid number, but we can never get rid of the division since $(a/b) * b = a$ iff $b \neq 0$. However, all universal properties are enjoyed also by "undefined" terms: $0 * (1/0) = 0$ since $\forall x, 0 * x = 0$, $(1/0) = (1/0)$ as $\forall x, x = x$, etc. Even if these properties are unlikely to be noticed by the student, it may happen to unconsciously employ them, for instance omitting side conditions that are made unnecessary by the additional properties. Users of HOL, Isabelle and Mizar often mix the two approaches.

Coq, NuPRL and PVS, being based on dependent types, can use yet another approach: they can encode the domain of a partial function in its type, so that application to arguments outside the domain is not allowed by the type system. We have not considered this approach in the paper since it is not well suited for students for several reasons. For instance, applications of partial functions require a new argument that is the proof that the argument belongs to the domain of the function. First year students do not understand that derivations can be represented syntactically and used as arguments to functions; moreover, when a mathematical operator has a standard notation, there is no natural place where the new argument can be put. PVS avoids this problem by removing the extra arguments and replacing them with side proofs for domain conditions. Wiedijk and Zwanenburg in [9] apply the approach of PVS to obtain a first order logic with domain conditions, without dependent types. Looking at the examples in the paper it is pretty obvious that the user is faced with too many side conditions. We believe that their number can be highly reduced in equality chains exploiting directed quasi equalities.

4 Conclusions

The most important problem to be faced when adopting proof assistants and types in education is probably that of representating and manipulating undefined expressions. In the case that adopting a logic with undefinedness is not an option, we are led to represent partial functions and operators in logics of total functions. Many formalisations have been proposed in the literature, but most of them are of limited use in education. We suggest that types of potentially undefined terms can be suitably represented with partial setoids and that partial functions can be represented as morphisms over partial setoids. This suggestion is not new: support for term manipulation in the partial setoids setting has already been implemented for Coq in [8]. However, in this setting equality chains require too many side conditions to be proved.

While analysing an informal equality chain that deals with potentially undefined terms we have noticed that a number of side conditions can be dropped by reducing definedness of both equality arguments to that of just one of them (to be proved later on). We have captured this behavior by introducing two oriented quasi

equalities $\stackrel{\triangleright}{=}$ and $\stackrel{\triangleleft}{=}$ that can be exploited in equality chains. We have also proposed the automation of the propagation of these equalities by combining automation for partial setoids with the notion of a strict morphism. A strict morphism is a morphism P such that $P(x)$ implies x defined. Hence, for a given strict morphism P, the system can automatically prove $P(x) \Rightarrow (y \stackrel{\triangleright}{=} x) \Rightarrow P(y)$, which means that P is a contra-variant morphism over the partial asymmetric relation $\stackrel{\triangleright}{=}$ (compare with [8]).

Thanks to automation, we have shown how an informal proof that computes the limit of a convergent series could be formalised in a proof assistant like Matita, providing only the minimal amount of rigorous justifications we expect from clever students even in a pen&paper proof.

As a future work, we plan to complete the implementation in Matita of the proposed automation and to test the proposed approach on a significant set of exercises to be submitted to a good number of students. We also plan a deeper comparison with alternative approaches. In particular, directed quasi equalities and their combinations with strict morphisms are likely to be exploitable with similar benefits in other contexts.

References

[1] A. Asperti, C. Sacerdoti Coen, E. Tassi, S. Zacchiroli. "User Interaction with the Matita Proof Assistant", in Journal of Automated Reasoning, Special Issue on User Interfaces for Theorem Proving. To appear.

[2] A. Bailey. "Representing Algebra in LEGO", M. Phil. thesis, University of Edinburgh.

[3] G. Barthe, O. Pons, V. Capretta. "Setoids in type theory", Journal of Functional Programming, 13(2), pages 261-293, 2003.

[4] W.M. Farmer. "Formalizing Undefinedness Arising in Calculus", in Automated Reasoning, Lecture Notes in Computer Science (LNCS), 3097:475-489, 2004.

[5] S. Feferman, "Definedness", Erkenntnis 43 (1995) 295-320.

[6] O. Müller, K. Slind. "Treating partiality in a logic of total functions", the Computer Journal, 40:640–652, 1997.

[7] M. Kerber, M. Kohlhase. "A Mechanization of Strong Kleene Logic for Partial Functions", CADE 1994: 371-385.

[8] C. Sacerdoti Coen, "A Semi-reflexive Tactic for (Sub-)Equational Reasoning", in Types for Proofs and Programs International Workshop, TYPES 2004. Lecture Notes in Computer Science (LNCS), Vol. 3839, 99–115, 2006.

[9] F. Wiedijk, J. Zwanenburg, "First order logic with domain conditions", in Theorem Proving in Higher Order Logics, Proceedings of TPHOLs 2003, D. Basin and B. Wolff (eds.), Springer LNCS 2758, 221-237, 2003.