The Quotient–Remainder Theorem for Natural Numbers: Discovery by Lazy Thinking*

Adrian Crăciun, Mădălina Hodorog
Institute e-Austria
Timişoara, Romania
{acraciun, mhodorog}@ieat.ro

Abstract

We present a case study in systematic theory exploration of natural numbers using the model recently proposed by Bruno Buchberger: the invention of the quotient-remainder decomposition theorem of natural numbers using the lazy thinking synthesis method.

We start from an initial knowledge base containing axioms and we develop the theory in exploration rounds using knowledge schemes, by introducing new notions, introducing properties about these notions, solving problems involving these notions.

This case study reflects the pedagogical value of the exploration model and of the lazy thinking method in particular, especially when supported by a system like *Theorema* (used in our case study), that allows a natural style similar with the one used by mathematicians.

1 Introduction to Systematic Theory Exploration

1.1 Systematic Exploration of Mathematical Theories

Recently, Bruno Buchberger proposed a new model for scheme-based systematic theory exploration (see [3], [4]). The model refers at developing a theory starting from an initial knowledge base containing axioms and then adding new concepts to it in exploration rounds. Each exploration round has the following steps:

- introduce new notions (function symbol, predicate symbol) using definition knowledge schemes.
- introduce and prove (or disprove) a proposition about a notion in the theory using proposition knowledge schemes.
- introduce problems involving a notion using algorithm schemes and solve them.
- introduce a new inference rule, by lifting knowledge or using inference schemes.

*Work supported by EU Marie Curie Project MERG-CT-2004-012718: SysteMaThEx

In our context, the knowledge schemes are higher order formulae that capture "interesting" mathematical knowledge. We store the knowledge schemes in libraries of schemes and we use them by instantiating them with symbols from the language of the theory. We formulate the knowledge schemes using higher order predicate logic and we assign them a unique name.

1.2 Context for the Case Study of Natural Numbers

We use the THEOREMA system to develop the knowledge base and all the tools that are useful in the exploration process.

1.2.1 Theory of Natural Numbers.

Language. To express the natural numbers theory, we use a first order predicate logic language with equality, see [8],[7]. For any theory, the language is a triple $\langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$, where $\mathcal{P}$, is the set of predicate symbols, $\mathcal{F}$ is the set of function symbols, and $\mathcal{C}$ is the set of constant symbols. In particular, for the natural numbers theory, the language is constucted from: the *is-nat* predicate symbol (which characterizes the natural numbers), the equality predicate symbol ($=$), the *superplus* function symbol ($^+$, that is the succesor function), the *identity* (id) function symbol, and the *zero* (0) constant symbol.

Knowledge Base. The knowledge base for the natural numbers theory consists of the equality axioms (formulated for any theory) and the Peano axioms (the generation of zero and of succesor axiom, the uniqueness of zero and of succesor axiom and the induction principle axiom). We must add, that the induction principle axiom and the equality axioms are axiom schemes (outside first order logic). To use, they will be lifted to the level of inference. Here we exemplify the Peano axioms for natural numbers:

$$\begin{array}{l} \texttt{Axioms}[\text{“generation”}, \texttt{any}[\textit{is-nat}[x]], \\ \quad \begin{array}{ll} \textit{is-nat}[0] & \text{“gen. zero”} \\ \textit{is-nat}[x^+] & \text{“gen. succ.”} \end{array}], \end{array}$$

$$\begin{array}{l} \texttt{Axioms}[\text{“uniqueness”}, \texttt{any}[\textit{is-nat}[x, y]], \\ \quad \begin{array}{ll} x^+ \neq 0 & \text{“zero”} \\ (x^+ = y^+) \Leftrightarrow (x = y) & \text{“succ.”} \end{array}], \end{array}$$

$$\begin{array}{l} \texttt{Axioms}[\text{“induction principle”}, \\ \quad (\mathfrak{F}[0] \wedge \underset{\textit{is-nat}[x]}{\forall} (\mathfrak{F}[x] \Rightarrow \mathfrak{F}[x^+]) \Rightarrow \underset{\textit{is-nat}[x]}{\forall} \mathfrak{F}[x]]. \end{array}$$

Inference Mechanism. The inference mechanism consists of the general predicate logic inference rules, the equality inference rules (such as: rewriting, simplification) and a specific rule for the natural numbers theory, that is the structural induction rule.

1.2.2 Knowledge schemes.

In our context, the knowledge schemes are higher order formulae that capture "interesting" mathematical knowledge. We store the knowledge schemes in libraries of schemes and we use them by instantiating them with symbols from

the language of the theory. We formulate the knowledge schemes using higher order predicate logic and we assign them a unique name.

Example 1 (Independent schemes: algebraic structures) Some examples of independent schemes are, for instance those denoting algebraic structures, such as:

$$\underset{p,op}{\forall}(\textit{is-semigroup}[p, op] \Leftrightarrow \underset{p[x,y,z]}{\forall} \bigwedge \left\{ \begin{array}{l} p[op[x, y]] \\ op[x, op[y, z]] = op[op[x, y], z] \end{array} \right.) \ ,$$

where p (standing for unary "sort" predicates, that denote domains), op (standing for binary functions) are higher order variables and *is-semigroup* is a special higher order constant (name, unique identifier of the schemes).

Example 2 (Independent schemes: relation structures) Other independent schemes are relation structures, such as *is-preorder*.

$$\underset{p,r}{\forall}(\textit{is-preorder}[p, r] \Leftrightarrow \underset{p[x,y,z]}{\forall} \bigwedge \left\{ \begin{array}{l} r[x, x] \\ (r[x, y] \wedge r[y, z]) \Rightarrow r[x, z] \end{array} \right.) \ ,$$

Example 3 (Theory-dependent schemes) Consider now the theory of natural numbers, as introduced in Subsubsection 1.2. The following is a natural numbers knowledge scheme, that captures the idea of a binary function defined recursively in terms of unary functions:

$$\underset{f,g,h}{\forall}(\textit{is-rec-nat-binary-fct-1r}[f, g, h] \Leftrightarrow \underset{is\text{-}nat[x,y]}{\forall} \bigwedge \left\{ \begin{array}{l} f[x, 0] = g[x] \\ f[x, y^+] = h[f[x, y]] \end{array} \right.) \ ,$$

The recursive structure of the above scheme reflects the inductive structure of natural numbers. In fact, a whole range of schemes can be constructed using the inductive structure of a particular domain.

Schemes are added to libraries based on experience (interestingness) and structure. While some of them are abstractions of mathematical ideas that were chrystalized over years and years of doing mathematics (or writing programs), some of them, dependent on recursive theories, can be generated systematically based on the structure of the theory. The two types of schemes are not necessarily disjoint.

Remark. For more information on the development of the natural numbers theory using the scheme-based exploration model, see [5, 6].

2 Lazy Thinking Synthesis Method

Problems are situation where a *solution* is desired. They can be introduced in the theory by instantiations of *problem schemes*. A method for solving problems in the context of scheme-based systematic theory exploration, synthesis by *lazy thinking*, was proposed by Bruno Buchberger, see [1, 2]. It consists of the following:

- a solution for the problem is proposed by selection of an *algorithm scheme* available,

- this is instantiated with new symbols (not present in the language) and a correctness proof is set up,
- the proof will very likely fail, due to the new symbols,
- the failed proof is analysed and conjectures on the new symbols are formulated, that allow the proof to get over the failure,
- the conjectures are specifications (problems) of the new symbols, and either concepts that verify the specifications can be retrieved from the knowledge, or the synthesis process is repeated.

3 Free Decomposition Problem

Free Decomposition Problem Scheme.

The knowledge scheme for binary free decomposition with pivot problem is:

$$\underset{obj,p_1,p_2,p_3,\otimes}{\forall} FBDp[obj,p_1,p_2,p_3,\otimes] \Leftrightarrow \underset{\substack{obj[x,y]\\ p_1[x]\\ p_2[y]}}{\forall} \underset{\substack{obj[z]\\ p_3[z]}}{\exists} x = y \otimes z,$$

that is for any *obj* x with the input condition p_1, and for any *obj* y with the input condition p_2 find a decomposition of x w.r.t. the compatible operation $\otimes$ involving y and the *obj* z with the input condition p_3.

Free Decomposition of Natural Numbers.

Instantiation of the $FBDp[obj,p_1,p_2,p_3,\otimes]$ problem scheme generates the decomposition of natural numbers:

$$FBDp[\textit{is-nat},\textit{true},\textit{is-positive},\textit{true},*] \Leftrightarrow \underset{\substack{\textit{is-nat}[x,y]\\ \textit{is-positive}[y]}}{\forall} \underset{\textit{is-nat}[z]}{\exists} x = y * z,$$

In the right formula of this scheme instantiation, we apply the skolemization rule to eliminate the existential quantifier, and we obtain the problem:

$$\underset{\substack{\textit{is-nat}[x,y]\\ \textit{is-positive}[y]}}{\forall} x = y * q[x,y],$$

where q is a new function symbol expressed in terms of the x and y objects. We try to solve this problem. The proof does not work for one of the cases from the knowledge scheme. Analyzing the failure of this proof, we realize that a modification of the original problem will avoid this failure:

$$\underset{\substack{\textit{is-nat}[x,y]\\ \textit{is-positive}[y]}}{\forall} x = y * q[x,y] + r[x,y].$$

In the exploration process, the *is-positive* predicate symbol was introduced according to the following definition:

$$\begin{array}{c}\texttt{Definition}[\text{“is-positive predicate symbol”},\\ \texttt{any}[\textit{is-nat}[x]],\\ \textit{is-positive}[x] \Leftrightarrow x \neq 0.],\end{array}$$

Solving the Free Decomposition Problem of Natural Numbers.

We want to solve the decomposition problem of natural numbers, that is synthesize the q function symbol from the formula:

$$\underset{is\text{-}nat[x,y]}{\forall} x = y * q[x,y], \text{ (1).}$$

using the following knowledge scheme:

$$\underset{f,g,h}{\forall}(is\text{–}nat\text{–}step\text{–}recl\text{–}fct\text{–}1\text{–}1[f,g,h] \Leftrightarrow$$
$$\underset{\substack{is\text{–}nat[x,y]\\ y>0}}{\forall}(q[x,y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[q[x-y,y]] & \Leftarrow \text{otherwise} \end{cases}),$$

where q, g and h are new function symbols with the requierement that they are functions over the domain denoted by *is-nat* predicate symbol.

Proof.

Take y_0 arbitrary but fixed such that *is-nat*$[y_0]$.

Show $\underset{is\text{-}nat[x]}{\forall} x = y_0 * q[x, y_0]$.

Case $y_0 = 0$:

Show $\underset{is\text{-}nat[x]}{\forall} x = 0$, which is in contradiction with the axiom of natural numbers. We obtain the first conjecture, according to which we take y_0 such that *is-positive*$[y_0]$.

Case $y_0 \neq 0$: Show $\underset{is\text{-}nat[x]}{\forall}(y_0 \neq 0) \Rightarrow x = y_0 * q[x, y_0]$ (2) .

We prove (2) by complete induction w.r.t. strict less equal ($<$).

Take x_0 arbitrary but fixed such that *is-nat*$[x_0]$.

Assume $\underset{is\text{-}nat[x]}{\forall} x < x_0 \Rightarrow x = y_0 * q[x, y_0]$ (3).

Show $x_0 = y_0 * q[x_0, y_0]$.

Case $x_0 < y_0$:

Using the $is\text{–}nat\text{–}step\text{–}recl\text{–}fct\text{–}1\text{–}1[f,g,h]$ scheme in this case:

$$q[x_0, y_0] = g[x_0].$$

So we have to show $x_0 = y_0 * g[x_0]$. (4)

From the properties of natural numbers:

$$x_0 < y_0 \Rightarrow \underset{is\text{-}positive[m]}{\forall} x_0 < m * y_0. \text{ (5)}$$

Instantiating in (5) $m \leftarrow g[x_0]$ we obtain:

$$x_0 < y_0 * g[x_0]. \text{ (6)}$$

Thus we fail to prove formula (4), which is in contradiction with (6).

Analyzing the failure, we observe that $x_0 = y_0 * g[x_0]$ holds if $g[x_0] = 0$.

Furthermore we can generalize the statement: $x_0 = y_0 * g[x_0] + x_0$ if $g[x_0] = 0$.

More general: $x_0 = y_0 * g[x_0] + k[x_0]$, and even:

$$x_0 = y_0 * q[x_0, y_0] + r[x_0, y_0].$$

So, we modify the original problem (1) which has to be solved in the following way:

$$\underset{\substack{is\text{-}nat[x]\\ is\text{-}positive[y]}}{\forall} x = y * q[x,y] + r[x,y].$$

4 Quotient-Remainder Theorem

Inventing the Quotient-Remainder Theorem.

We have seen that we try to solve the original problem (1), but we fail. Analizing the failure, we realize that a modification of the original problem will avoid this failure:

$$\underset{\substack{is\text{-}nat[x]\\ is\text{-}positive[y]}}{\forall} x = y * q[x,y] + r[x,y].$$

Assuming this, we want to solve this modified problem, that is synthesize the q, r function symbols using the lazy thinking method with the following knowledge schemes:

$$\underset{f,g,h}{\forall}(is\text{–}nat\text{–}step\text{–}recl\text{–}fct\text{–}1\text{–}1[q,g,h] \Leftrightarrow$$
$$\underset{\substack{is\text{–}nat[x,y]\\ y>0}}{\forall}(q[x,y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[q[x-y,y]] & \Leftarrow \text{otherwise} \end{cases}),$$

$$\underset{r,k,t}{\forall}(is\text{–}nat\text{–}step\text{–}recl\text{–}fct\text{–}new\text{–}1\text{–}1[r,k,t] \Leftrightarrow$$
$$\underset{\substack{is\text{–}nat[x,y]\\ y>0}}{\forall}(r[x,y] = \begin{cases} k[x] & \Leftarrow x < y \\ t[r[x-y,y]] & \Leftarrow \text{otherwise} \end{cases}),$$

where q, r, g, h, k, t are new function symbols with the requierement that they are functions over the domain denoted by the *is-nat* predicate symbol.

Proof.

Take y_0 arbitrary but fixed such that *is-nat*$[x_0]$, *is-positive*$[y_0]$.

Show $\underset{is\text{-}nat[x]}{\forall} x = y_0 * q[x,y_0] + r[x,y_0]$. (7).

We prove formula (7) by complete induction w.r.t. $<$ predicate symbol.

Take x_0 arbitrary but fixed such that *is-nat*$[x_0]$.

Assume $\underset{\substack{is\text{-}nat[x]\forall\\ x<x_0}}{x} = y_0 * q[x,y_0] + r[x,y_0]$. (8)

Show $x_0 = y_0 + q[x_0,y_0] + r[x_0,y_0]$. (9)

Case $x_0 < y_0$:

From the properties of natural numbers:

$$x_0 < y_0 \Rightarrow \underset{is\text{-}positive[m]}{\forall} x_0 < m * y_0. \text{ (10)}$$

Using the *is–nat–step–recl–fct–1–1* and *is–nat–step–recl–fct–new–1–1* knowledge schemes in this case, we have:

$$q[x_0,y_0] = g[x_0] \wedge r[x_0,y_0] = k[x_0]$$

Replacing this formulae in (9), we have to show:

$$x_0 = y_0 * g[x_0] + k[x_0]. \quad (11)$$

From $x_0 < y_0$, using formula (10), we obtain:

$$x_0 < y_0 * g[x_0], \text{ where } \textit{is-nat}[g[x_0]].$$

Moreover

$$x_0 < y_0 * g[x_0] + k[x_0], \text{ where } \textit{is-nat}[g[x_0]].$$

We fail to prove formula (11).
We observe that formula (11) holds if $g[x_0] = 0 \wedge k[x_0] = x_0$.
We generate the following conjectures:

Conjecture 1: $k[x] = x \wedge$

Conjecture 2: $g[x] = 0.$

which guarantee that formula (11) holds.
Case $x_0 \not< y_0$:
Using the *is–nat–step–recl–fct–1–1* and *is–nat–step–recl–fct–new–1–1* knowledge schemes in this case, we have:

$$q[x_0, y_0] = h[q[x_0 - y_0, y_0]] \wedge r[x_0, y_0] = t[r[x_0 - y_0, y_0]]$$

Replacing this formulae in (9), we have to show:

$$x_0 = y_0 * h[q[x_0 - y_0, y_0]] + t[r[x_0 - y_0, y_0]]. \quad (13)$$

Instantiating $x \leftarrow x_0 - y_0$ in formula (8) (because $x_0 - y_0 < x_0$ from the properties of natural numbers) we obtain:

$$x_0 - y_0 = y_0 * q[x_0 - y_0, y_0] + r[x_0 - y_0, y_0]$$

which implies:

$$x_0 = y_0 * q[x_0 - y_0, y_0] + r[x_0 - y_0, y_0] + y_0. \quad (14)$$

From (13) and (14), we have to show:

$$y_0 * h[q[x_0 - y_0, y_0]] + t[r[x_0 - y_0, y_0]] = y_0 * q[x_0 - y_0, y_0] + r[x_0 - y_0, y_0] + y_0. \quad (15)$$

Formula (15) is equivalent with:

$$y_0 * h[q[x_0 - y_0, y_0]] + t[r[x_0 - y_0, y_0]] = y_0 * (q[x_0 - y_0, y_0] + 1) + r[x_0 - y_0, y_0]. \quad (16)$$

We fail to prove formula (16).
We generate the following conjectures:

Conjecture 3: $h[x] = x + 1 \wedge$

Conjecture 4: $t[y] = y,$

which allow us to continue with the proof.

Replacing $q[x_0 - y_0, y_0] \leftarrow x, r[x_0 - y_0, y_0] \leftarrow y$ in formula (16), we obtain:

$$y_0 * h[x] + t[y] = y_0 * (x + 1) + y, \quad (17)$$

which holds based on the generated conjectures.

As solutions to our problem, we obtain the definitions for the q and r function symbols.

The definition for the q function symbol is in fact the definition for the well-known *quotient* (*quot*) function symbol:

$$quot[x, y] = \begin{cases} 0 & \Leftarrow x < y \\ quot[x - y, y] + 1 & \Leftarrow \text{otherwise} \end{cases},$$

generated using the conjectures 1 and 3 from the proof, and though replacing $q \leftarrow quot, g \leftarrow 0, h \leftarrow {}^{+}$ in *is–nat–step–recl–fct–1–1*$[quot, g, k]$ knowledge scheme.

The definition for the r function symbol represents in fact the definition for the well-known *remainder* (*rem*) function symbol:

$$rem[x, y] = \begin{cases} x & \Leftarrow x < y \\ rem[x - y, y] \Leftarrow \text{otherwise} & \end{cases},$$

generated using the conjectures 2 and 4 from the proof, and though replacing $r \leftarrow rem, k \leftarrow id, t \leftarrow id$ in *is–nat–step–recl–fct–1–1*$[r, k, t]$ knowledge scheme.

Uniqueness of the Quotient-Remainder Theorem.

We have seen that the binary free decomposition with pivot problem scheme allows us to invent the definitions for the well-known *quotient* and *remainder* function symbols. We also invent the following theorem:

$$\underset{\substack{is\text{-}nat[x] \\ is\text{-}positive[y]}}{\forall} x = y * quot[x, y] + rem[x, y]. \ (\triangle)$$

The theorem ($\triangle$) does not have unique solutions. We give the following counterexample to illustrate this statement: for $x \leftarrow 7, y \leftarrow 2$, we find the following solutions:

$$7 = 2 * 1 + 5, \text{ so } quot[7, 2] = 1, rem[7, 2] = 5,$$

$$7 = 2 * 3 + 1, \text{ so } quot[7, 2] = 3, rem[7, 2] = 1.$$

There is one particular case in which the theorem ($\triangle$) have unique solutions: the case in which $x < y$. According to the definitions of *quot*,*rem*, in this case:

$$quot[x, y] = 0, rem[x, y] = x.$$

So $x = y * 0 + x \Rightarrow x = 0 + x \Rightarrow x = x$, which holds.

We proof the following property of the theorem ($\triangle$):

$$\underset{\substack{is\text{-}nat[x] \\ is\text{-}positive[y]}}{\forall} rem[x, y] < y.$$

Proof.

Take y_0 arbitrary but fixed such that *is-positive*$[y_0]$.

We have to show $\underset{is\text{-}nat[x]}{\forall} rem[x, y_0] < y_0$. (18)

We prove formula (18) by complete induction w.r.t the $<$ predicate symbol.

Take x_0 arbitrary but fixed such that *is-nat*$[x_0]$.

Assume $\underset{x<x_0}{\forall} rem[x, y_0] < y_0$. (19)

Show $rem[x_0, y_0] < y_0$. (20)

Case $x_0 < y_0$:

By definition of the *rem* function symbol in this case, $rem[x_0, y_0] = x_0 < y_0$, so formula (20) holds.

Case $x_0 \not< y_0$:

By definition of the *rem* function symbol in this case,

$$rem[x_0, y_0] = rem[x_0 - y_0, y_0]. \text{ (21)}.$$

From (20) and (21), we have to show $rem[x_0 - y_0, y_0] < y_0$. (22)

From the properties of $<$ predicate symbol, we know

$$\underset{\substack{is\text{-}nat[x]\\ is\text{-}positive[y]}}{\forall} x - y < x \text{ (23)}.$$

Instantiating $x \leftarrow x_0, y \leftarrow y_0$ in (23) as *is-nat*$[x_0]$, *is-positive*$[y_0]$, we obtain:

$$x_0 - y_0 < x_0. \text{ (24)}$$

Based on formula (24), instantiating $x \leftarrow x_0 - y_0$ in (19) we obtain:

$$rem[x_0 - y_0, y_0] < y_0,$$

so formula (22) is proved.

This property will guarantee that the solutions to the problem ($\triangle$) are unique. Thus, we have prove the following theorem:

$$\underset{\substack{is\text{-}nat[x]\\ is\text{-}positive[y]}}{\forall} x = y * quot[x, y] + rem[x, y] \wedge rem[x, y] < y,$$

where *rem*, *quot* are the quotient and the remainder function symbols defined above. Actually, this theorem represents the *quotient-remainder* theorem.

5 Conclusion

We present part of a case study in systematic theory exploration of natural numbers, using a scheme based exploration model recently proposed by Bruno Buchberger: the discovery of the quotient-remainder decomposition of natural numbers.

We start from the well-known axioms of natural numbers and develop the theory in exploration rounds, by introducing new notions (using definition schemes), exploring their properties (using proposition schemes), solving problems involving the notions (introduced by problem schemes).

The particular problem we focus on is: for two natural numbers, x, y find another, z, such that $x = yz$ (x can be decomposed in the product of y and z). To solve the problem we apply *lazy thinking*, i.e. we try to invent an algorithm A to solve the problem $x = yA[x, y]$, by proposing an algorithmic idea (scheme) for A, and proving that it solves the problem.

As expected, this problem cannot be solved, and we show how the analysis of the failure leads to the quotient-remainder decomposition of natural numbers, i.e. for any natural numbers x, y,y positive, $x = yq[x, y] + r[x, y]$. By lazy thinking, we discover the appropriate algorithms q and r.

This case study reflects the pedagogical value of the exploration model and of the lazy thinking method in particular, especially when supported by a system like *Theorema* (used in our case study), that allows natural style (close to textbook mathematics) input and output. We emphasize the role of using accumulated mathematical experience (through knowledge schemes), learning from failure, expanding the available inference machinery (adding new inference rules), experimentation.

References

[1] Buchberger, B.: Algorithm Invention and Verification by Lazy Thinking. In D. Petcu, V. Negru, D. Zaharie, and T. Jebelean, editors, *Proceedings of SYNASC 2003, 5th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing Timisoara*, pages 2-26, Timisoara, Romania, 1-4 October 2003. Copyright: Mirton Publisher.

[2] Buchberger, B., Crăciun, A.: Algorithm Synthesis by Lazy Thinking: Examples and Implementation in Theorema. In F. Kamareddine, editor, *Electronic Notes in Theoretical Computer Science*, volume **93**, pages 24-59, 18 February 2004. Proc. of the Mathematical Knowledge Management Workshop, Edinburgh, Nov. 25, 2003.

[3] Buchberger, B.: Algorithm–Supported Mathematical Theory Exploration: A Personal View and Strategy. Lecture Notes in Artificial Intelligence, Springer, 7th Conference on Artificial Intelligence and Symbolic Computation(Research Institute for Symbolic Computation, Hagenberg, Austria) (Proceedings of AISC 2004):16, September.

[4] Buchberger, B., Crăciun A., Jebelean T., Kovacs L., Kutsia T., Nakagawa K., Piroi F., Popov N., Robu J., Rosenkranz M., Windsteiger W.: Theorema: Towards Computer-Aided Mathematical Theory Exploration. Journal of Applied Logic, pages 470-504, 2006.

[5] Hodorog, M.: Scheme–based Systematic Exploration of Mathematical Theories. Case study: The Natural Numbers. Technical Report no.07-06, I-Eat, 2006.

[6] Hodorog, M., Crăciun, A.: Scheme-Based Systematic Exploration of Natural Numbers. *Proceedings of SYNASC 2006, 8th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing Timisoara, Romania (2006)*, pages 23-34, Timisoara, Romania, September 26-29 2006. IEEE Computer Society Press.

[7] Manna, Z., Wadinger, R.: The Logical Basis for Computer Programming, Volume I, Deductive Reasoning. Addison-Wesley Publishing Co, SUA (1985)

[8] Shoenfield, J. R.: Mathematical Logic. Addison-Wesley Publishing Co, SUA (1967)