
Scheme-Based Systematic Exploration of Natural Numbers

authors: Adrian Craciun, **Madalina Hodorog**

acraciun@ieat.ro, mhodorog@ieat.ro

Institute E-Austria Timisoara

SYNASC 2006

26th of September 2006

1 Overview

1. Introduction: context for the case study
2. Exploration model and examples
- 1.1 3. Implementation
4. Related work
5. Conclusion

2 1. Introduction: context for the case study

2.1 Notion of a theory, $\mathcal{T} = \langle \mathcal{L}, \mathcal{KB}, \mathcal{IR} \rangle$

\mathcal{L} - a first order predicate logic language with equality;

$\mathcal{L} = \langle \mathcal{P}, \mathcal{F}, \mathcal{C} \rangle$, where,

\mathcal{P} - predicates (including the binary "=" predicate);

\mathcal{F} - functions (including the unary "id" function);

\mathcal{C} - constants.

\mathcal{KB} - knowledge base (formulae).

\mathcal{IR} - inference mechanism consisting of:

- first order predicate logic calculus rules;
- rewriting rules;
- specific inference rules.

3 1. Introduction: context for the case study

3.1 Theory of natural numbers,
 $\mathcal{T}_N = \langle \mathcal{L}_N, \mathcal{KB}_N, \mathcal{IR}_N \rangle$
 $\mathcal{L}_N = \langle \langle \text{is-nat}, = \rangle, \langle +, \text{id} \rangle \rangle, \langle 0 \rangle$.

\mathcal{KB}_N : equality axioms and Peano axioms.

Axioms["Peano axioms",	
is-natural[0]	"nat:generati
\forall is-natural[x^+]	"nat:generati
is-natural[x]	
\forall $x^+ \neq 0$	"nat:uniquen
is-natural[x]	
\forall ($x^+ = y^+ \Rightarrow x = y$)	"nat:uniquen
is-natural[x, y]	
$(\mathcal{F}[0] \wedge \forall_{\text{is-natural}[x]} (\mathcal{F}[x] \Rightarrow \mathcal{F}[x^+])) \Rightarrow \forall_{\text{is-natural}[x]} \mathcal{F}[x]$	"induction pi
]	

Remark: "induction principle" axiom is an *axiom scheme* (outside first order logic). To use, it will be lifted to the level of inference.

$\mathcal{IR}_N =$

{	<i>structural induction</i> rule
	<i>general predicate logic inference</i> rules
	<i>equality inference</i> rules (rewriting, simplifica

4 1. Introduction - context for the case study

4.1 Knowledge Schemes

- higher order formulae that capture "interesting" mathematical knowledge;
- stored in libraries of schemes.

I. Knowledge schemes dependent on the theory

4.1.1 being developed (in our case: natural number theory)

$$\forall_{f,g,h} \left(\text{is-rec-nat-binary-fct-1r}[f, g, h] \Leftrightarrow \right. \\ \left. \forall_{\text{is-natural}[x,y]} ((f[x, 0] = g[x]) \wedge (f[x, y^+] = h[f[x, y]])) \right)$$

$$\forall_{f,g,h} \left(\text{is-rec-nat-binary-fct-1l}[f, g, h] \Leftrightarrow \right. \\ \left. \forall_{\text{is-natural}[x,y]} ((f[0, y] = g[y]) \wedge (f[x^+, y] = h[f[x, y]])) \right)$$

$$\forall_{f,g,h,\text{pred}} \left(\text{is-nat-rec-binary-rel-2}[f, g, h, \text{pred}] \Leftrightarrow \right. \\ \left. \forall_{\text{is-natural}[x,y,z]} ((f[x, 0] \Leftrightarrow g[x] = 0) \wedge \right. \\ \left. (f[x, y^+] \Leftrightarrow (\text{pred}[x, h[y]] \vee f[x, y]))) \right)$$

5 1. Introduction - context for the case study

5.0.1 II. Knowledge schemes independent of any theory

$$\forall_{p, \text{bin-op}} \left(\text{is-semigroup}[p, \text{bin-op}] \Leftrightarrow \forall_{p[x,y,z]} \left((p[\text{bin-op}[x, y]]) \wedge \right. \right. \\ \left. \left. ((\text{bin-op}[x, \text{bin-op}[y, z]]) = (\text{bin-op}[\text{bin-op}[x, y], z])) \right) \right)$$

$$\forall_{p, \text{bin-op}, \text{zero}} \left(\text{is-monoid}[p, \text{bin-op}, \text{zero}] \Leftrightarrow \right. \\ \left. \text{is-semigroup}[p, \text{bin-op}] \wedge \forall_{p[x]} (\text{bin-op}[x, \text{zero}] = x) \right)$$

$$\forall_{p, \text{bin-op}, \text{zero}, \text{inv}} \left(\text{is-group}[p, \text{bin-op}, \text{zero}, \text{inv}] \Leftrightarrow \right. \\ \left(\text{is-monoid}[p, \text{bin-op}, \text{zero}] \wedge \right. \\ \left. \left(\forall_{p[x]} \text{bin-op}[x, \text{inv}[x]] = \text{zero} \right) \right)$$

$$\forall_{p,r} \left(\text{is-preorder}[p, r] \Leftrightarrow \right. \\ \left. \forall_{p[x,y,z]} (r[x, x] \wedge ((r[x, y] \wedge r[y, z]) \Rightarrow (r[x, z]))) \right)$$

$$\forall_{p,r} \left(\text{is-partial-ordering}[p, r] \Leftrightarrow \right. \\ \left. \text{is-preorder}[p, r] \wedge \forall_{p[x,y,z]} ((r[x, y] \wedge r[y, x]) \Rightarrow (x = y)) \right)$$

6 2. Exploration model and examples

Exploration consistent with the model

6.0.1 proposed

by B. Buchberger - AISC 2004

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

7 2. Exploration model and examples

7.1 Exploration examples

7.1.1 Introducing a new notion

-search in the scheme library for a definition knowledge scheme that can be instantiated with the symbols of the language.

schAdditionFunction
$\forall_{f,g,h} \left(\text{is-rec-nat-binary-fct-1r}[f, g, h] \Leftrightarrow \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} ((f[x, 0] = g[x]) \wedge (f[x, y^+] = h[f[x, y]])) \right)$
<p>possibleSubsts :=</p> <p>{f → •[Plus], g → •[Identity], h → •[SuperPlus]},</p> <p>{f → •[Plus1], g → •[SuperPlus], h → •[SuperPlus]},</p> <p>{f → •[ProjLeft], g → •[Identity], h → •[Identity]},</p> <p>{f → •[ProjLeftPlus1], g → •[SuperPlus], h → •[Identity]}</p>
<p>newConcepts :=</p> <p>Map[UseScheme[is-rec-nat-binary-fct-1r, #] &, possibleSubsts]</p>
newConcepts

8 2. Exploration model and examples

8.0.1 Introducing propositions (I) - equivalent definitions

✧ search in the scheme library a potential equivalent definition scheme for the + function symbol.

schNewAdditionFunction
$\forall_{f,g,h} \left(\text{is-rec-nat-binary-fct-11}[f, g, h] \Leftrightarrow \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} ((f[0, y] = g[y]) \wedge (f[x^+, y] = h[f[x, y]])) \right)$
<p>possibleSubsts :=</p> <p>{f → •[Plus], g → •[Identity], h → •[SuperPlus]}, {f → •[Plus1], g → •[SuperPlus], h → •[SuperPlus]}, {f → •[ProjRight], g → •[Identity], h → •[Identity]}, {f → •[ProjRightPlus1], g → •[SuperPlus], h → •[Identity]}</p>
<p>newConcepts :=</p> <p>Map[UseScheme[is-rec-nat-binary-fct-11, #] &, possibleSubsts]</p>
newConcepts

✧ introduce the equivalent definition for the + function as a proposition in the theory.

<p>Proposition["nat.1.2:new simple recursion: 2IdentityTMSuperPlus", any[is-natural[x], is-natural[y]], $\begin{array}{l} 0 + y = y \\ x^+ + y = (x + y)^+ \end{array}$]</p>
--

ProofNewAddition

ProofNextNewAddition

9 2. Exploration model and examples

Introducing propositions (II) - semigroup, monoid, group: we look

9.0.1 at the algebraic knowledge schemes that can be instantiated with the + function symbol and the other symbols from the language.

9.0.2 Semigroup algebraic structure

✧ the first algebraic scheme instantiated with $\{p \rightarrow \bullet[\text{IsNatural}], \text{bin-op} \rightarrow \bullet[+]\}$ gives :

schSemigroup /. $\{p \rightarrow \bullet[\text{IsNatural}], \text{bin-op} \rightarrow \bullet[+]\}$

$$\{\text{is-semigroup}^{\text{TMIsNatural}, +} : \Leftrightarrow \bigvee_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y] \\ \text{is-natural}[z]}} (\text{is-natural}[x + y] \wedge (x + (y + z) = (x + y) + z))\}$$

Proposition["nat.1.2:associativity",
any[is-natural[x], is-natural[y], is-natural[z]],
(x + y) + z = x + (y + z)
]

✧ ProofAdditionAssociativity

9.0.3 Monoid algebraic structure

✧ the next scheme instantiated with $\{p \rightarrow \bullet[\text{IsNatural}], \text{bin-op} \rightarrow \bullet[+], \text{zero} \rightarrow \bullet[0]\}$ gives:

schMonoid /. $\{p \rightarrow \bullet[\text{IsNatural}], \text{bin-op} \rightarrow \bullet[+], \text{zero} \rightarrow \bullet[0]\}$

$$\{\text{is-monoid}^{\text{TMIsNatural}, +, 0} : \Leftrightarrow \text{is-semigroup}^{\text{TMIsNatural}, +} \bigwedge_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y] \\ \text{is-natural}[z]}} (x + 0 = x)\}$$

✧ Remark: all the propositions are already introduced in the theory!

1| 2. Exploration model and examples

10.0. Group algebraic structure

✧ the next algebraic scheme is the is-group scheme:

schGroup

$$\forall_{p, bin-op, zero, inv} \left(\text{is-group}[p, bin-op, zero, inv] :\Leftrightarrow \text{is-monoid}[p, bin-op, zero] \wedge \left(\forall_{p[x]} bin-op[x, inv[x]] = zero \right) \right)$$

✧ the possible substitutions in the scheme:

possibleSubsts := {{p → •[is-natural], bin-op → •[Plus], zero → •[0], inv → •[Identity]}, {p → •[is-natural], bin-op → •[Plus], zero → •[0], inv → •[SuperPlus]}}

✧ gives us the following:

Map[UseScheme[schGroup, #] &, possibleSubsts]

$$\left\{ \begin{array}{l} \text{is-group}[\text{TMIsNatural}, +, 0, \text{Identity}] :\Leftrightarrow \text{is-monoid}[\text{TMIsNatural}, +, 0] \wedge \left(\forall_{\text{is-natural}[x]} (x + x) = 0 \right), \\ \text{is-group}[\text{TMIsNatural}, +, 0, \text{TMSuperPlus}] :\Leftrightarrow \\ \text{is-monoid}[\text{TMIsNatural}, +, 0] \wedge \left(\forall_{\text{is-natural}[x]} (x + x^+) = 0 \right) \end{array} \right\}$$

✧ **Remark:** none of the above formula hold;

(the identity, the succesor functions are not inverses for the natural numbers).

✧ **introducing and solving a problem:**

- invent a new function symbol (Θ), such that *is-group*[*IsNatural*, +, 0, Θ];

(prove the formula

$$\forall_{\text{is-natural}} x + \Theta x = 0).$$

- apply the *lazy thinking method* to synthesize the Θ function;

- our goal introduces a contradiction \Rightarrow the problem has no solution;

✳ **conclusion:** there is no inverse function for the natural numbers.

1 2. Exploration model and examples

11.0. Multiplication Function Symbol.

- knowledge scheme used for inventing new notions:

schMultiplicationConstant

$$\forall_{f,h, const} \left(\text{is-rec-nat-binary-fct-0r}[f, const, h] :\Leftrightarrow \right.$$

$$\left. \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} ((f[x, 0] = const) \wedge (f[x, y^+] = h[x, f[x, y]])) \right)$$

- the substitution $\{f \rightarrow \bullet[\text{Times}], \text{const} \rightarrow \bullet[0], h \rightarrow \bullet[\text{Plus}]\}$ gives:

schMultiplicationConstant /.

$\{f \rightarrow \bullet[\text{Times}], \text{const} \rightarrow \bullet[0], h \rightarrow \bullet[\text{Plus}]\}$

$$\{\text{is-rec-nat-binary-fct-0r}[\bullet, 0, \text{Plus}] :\Leftrightarrow \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} ((x * 0 = 0) \wedge (x * y^+ = x + x * y))\}$$

Definition["nat.2.1: simple recursion: 3TMTimes",

any[is-natural[x], is-natural[y]],

$$\left. \begin{array}{l} x * 0 = 0 \\ x * y^+ = x * y + x \end{array} \right]$$

11.0.: Exponentiation Function Symbol.

- knowledge scheme used for inventing new notions: *schMultiplicationConstant*.

- the substitution $\{f \rightarrow \bullet[\text{Power}], \text{const} \rightarrow \bullet[1], h \rightarrow \bullet[\text{Times}]\}$ gives:

schMultiplicationConstant /.

{f → •[Power], const → •[1], h → •[Times]}

$$\{ \text{is-rec-nat-binary-fct-0r}[\wedge, 1, *] : \Leftrightarrow \left. \begin{array}{l} \forall_{\text{is-natural}[x]} ((x^0 = 1) \wedge (x^{y^+} = x * x^y)) \\ \text{is-natural}[y] \end{array} \right\}$$

Definition["exponentiation.2.2", any[is-natural[x], is-natural[y]],

$$\left. \begin{array}{l} x^0 = 1 \\ x^{y^+} = x^y * x \end{array} \right]$$

11.0.: Strict Less Than Equal Relation Symbol.

- knowledge scheme used for inventing new notions:

schRelationSymbol

$$\left. \begin{array}{l} \forall_{f,g,h,pred} \text{is-nat-rec-binary-rel-2}[f, g, h, pred] : \Leftrightarrow \\ \\ \forall_{\substack{\text{IsNatural}[x] \\ \text{IsNatural}[y] \\ \text{IsNatural}[z]}} ((f[x, 0] \Leftrightarrow g[x] = 0) \wedge (f[x, y^+] \Leftrightarrow pred[x, h[y]] \vee f[x, y])) \end{array} \right\}$$

- the substitution {f→•[<], g→•[SuperPlus], h→•[Identity], pred→•[Equal]} gives:

schRelationSymbol /.

{f → •[<], g → •[SuperPlus], h → •[Identity], pred → •[Equal]}

$$\{ \text{is-nat-rec-binary-rel-2}[\text{<}, \text{TMSuperPlus}, \text{Identity}, \text{=}] : \Leftrightarrow \left. \begin{array}{l} \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y] \\ \text{is-natural}[z]}} ((x < 0 \Leftrightarrow x^+ = 0) \wedge (x < y^+ \Leftrightarrow (x = y) \vee x < y)) \end{array} \right\}$$

Definition["relation symbol.3.2: TMSuperPlusIdentity",
any[is-natural[x], is-natural[y]],
 $(x < 0) \Leftrightarrow (x^+ = 0)$
 $(x < y^+) \Leftrightarrow ((x = y) \vee (x < y))$]

1: 2. Exploration model and examples

12.0. Introducing a new inference rule (Complete Induction Principle).

-introduce the proposition into the theory:

$$\text{Proposition} \left[\text{"Complete Induction"}, \right. \\ \left. \left(\forall_{\text{is-natural}[x]} \left(\forall_{\text{is-natural}[z]} (z < x \Rightarrow \mathcal{F}[z]) \Rightarrow (\mathcal{F}[x]) \right) \right) \Rightarrow \right. \\ \left. \forall_{\text{is-natural}[x]} \mathcal{F}[x] \right]$$

- the proof follows the proof from Manna and Waldinger 1985 book.

- the proof of the *Proposition(Complete Induction)* has the following form:

$$A_1 \Rightarrow G_1, \text{ where}$$

$$A_1 : \left(\forall_{\text{is-natural}[x]} \left(\forall_{\text{is-natural}[z]} (z < x \Rightarrow \mathcal{F}[z]) \Rightarrow (\mathcal{F}[x]) \right) \right) \text{ and} \\ G_1 : \forall_{\text{is-natural}[x]} \mathcal{F}[x]$$

- in order to prove this property, we prove an alternative property:

$$\left(\forall_{\text{is-natural}[y]} \left(\forall_{\text{is-natural}[z]} (z < y \Rightarrow \mathcal{F}[z]) \right) \right) \Rightarrow \forall_{\text{is-natural}[y]} \mathcal{G}[y] \quad \text{P :}$$

- our main goals are in this way the following:

$$A_1 \Rightarrow P \text{ (Complete Induction.2)}$$

$$P \Rightarrow G_1 \text{ (Complete Induction.1)}$$

1: 2. Exploration model and examples

Summary

- new induction principle allows
- new recursive schemes
- that can be used to introduce new concepts (see next slide)
- and their properties, which can be proved by the new induction principle.

1. 2. Exploration model and examples.

14.0. Quotient/Remainder Function Symbols.

- knowledge scheme used for inventing new concepts:

$$\text{schQuotRemFuncSymbol}$$

$$\forall_{f,g,h} \left(\text{is-nat-step-recl-fct-1-1}[f, g, h] \Leftrightarrow \right.$$

$$\left. \forall_{\text{is-natural}[x,y]} \left(f[x, y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[f[x - y, y]] & \Leftarrow \text{otherwise} \end{cases} \right) \right)$$

- original problem considered for solving:

$$\forall_{\text{is-natural}[x,y]} (x = y * \text{quot}[x, y])$$

- modified problem considered for solving:

$$\forall_{\text{is-natural}[x,y]} (x = y * \text{quot}[x, y] + \text{rem}[x, y])$$

- applying the lazy thinking method with the *schQuotRemFuncSymbol* knowledge scheme as candidate for both *quot* and *rem*

⇒ the quotient and remainder function symbols as solutions to the modified problem:

$$\text{Definition} \left[\text{"nat.5.1: simple recursion less: quotient"}, \right.$$

$$\text{any}[\text{is-natural}[x], \text{is-natural}[y]],$$

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

$$\text{Definition} \left[\text{"nat.5.2: simple recursion less: remainder"}, \right.$$

$$\text{any}[\text{is-natural}[x], \text{is-natural}[y]],$$

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

Divides Relation Symbol.

- knowledge scheme used for inventing new concepts:

schDividesRelation

$$\left[\forall_{r,p,q,const} \left(\text{is-nat-step-recr-rel-0-1-1}[r, p, q, const] : \Leftrightarrow \left(\forall_{\text{is-natural}[x,y]} \left(r[x, y] \Leftrightarrow \begin{cases} \text{const} & \Leftarrow y = 0 \\ p[x, y] & \Leftarrow x > y \\ q[r[x, y - x]] & \Leftarrow \text{otherwise} \end{cases} \right) \right) \right] \right]$$

schDividesRelation

- the substitution $\{r \rightarrow \bullet[|], p \rightarrow \bullet[\text{False}], q \rightarrow \bullet[\text{id}_B], \text{const} \rightarrow \bullet[\text{True}]\}$ gives:

schDividesRelation /.

$$\{r \rightarrow \bullet[|], p \rightarrow \bullet[\text{False}], q \rightarrow \bullet[\text{id}_B], \text{const} \rightarrow \bullet[\text{True}]\}$$

$$\forall_{|, \text{False}, \text{True}, \text{True}} \left(\text{is-nat-step-recr-rel-0-1-1}[|, \text{False}, \text{True}, \text{True}] : \Leftrightarrow \left(\forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} (x/y \Leftrightarrow \{ \text{True} \Leftarrow y = 0, \text{False}[x, y] \Leftarrow x > y, \text{True}[x/(y-x)] \Leftarrow \text{otherwise} \}) \right) \right)$$

Definition["nat.6.1: simple recursion less: new divides",

any[is-natural[x], is-natural[y]],

$$x \left| y \Leftrightarrow \begin{cases} \text{True} & \Leftarrow y = 0 \\ \text{False} & \Leftarrow x > y \\ x \mid (y - x) & \Leftarrow \text{otherwise} \end{cases} \right]$$

1: 3. Implementation

New Provers in the system

15.0.: ✓ The NNIP induction prover.

Purpose: implements the *induction* over natural numbers with few improvements.

Main New Features: we introduce the sort condition for the induction variable.

Used in : NatProver, NatProverPC, SimplerNatProverPC.

15.0.: ✓ The NNIPC induction prover.

Purpose: implements the *complete induction* over natural numbers.

Used in: NewNatProver.

15.: New Function

15.1.: ✓ UseScheme[*scheme,substitutions*] function.

Functionality: takes as arguments the knowledge scheme *scheme* and the list *substitutions* of all the possible combinations between the symbols from the language and generates the new concepts.

15.: Libraries for knowledge schemes

1| 4. Related work

16.0. HR

Methodology:

Examples

Model Checking
→

Mathematical Concepts $\xrightarrow{\text{Theorem Prover}}$ Proved Mathematical Concepts.

Resemblances:

✓ the use of the previously discovered theorems for proving.

Differences:

✓ old concepts $\xrightarrow{\text{Production Rules}}$ invention of new concepts.

✓ use of schemes.

16.0. MATHsAID system

Methodology: Initial Axioms $\xrightarrow{\text{Proving}}$ Consequences $\xrightarrow{\text{Filtering}}$ Interesting Mathematical Concepts and Theorems.

Resemblances:

✓ new theorems are build up in layers, rather than be discovered all at one time.

Differences:

✓ automated vs. semi-automated

✓ use of schemes

1' 5. Conclusion

17.0. Our work

- we reported on a case study in the scheme-based theory exploration: the natural numbers;
- we have shown that Buchberger's model is successfully applied using the *THEOREMA* system (compared to the textbook Manna, Waldinger).

17.0.: Exploration steps - what do we explore?

- addition, multiplication, the less than equal relation, exponentiation, subtraction, quotient, remainder;
- *next*: prime numbers (*one of the future goals*: the prime decomposition theorem).

17.0.: Exploration steps - how do we explore?

- new induction provers and prototype functions;
- an exploration round:
 - scheme retrieval, instantiation (introducing notions, propositions, problems), theory formulation - is done by hand, with limited system support (UseScheme);
 - proving done automatically with *Theorema*.

1: 5. Conclusion

18.0. Why this exploration?

- ☑ the knowledge schemes
 - represent condensed "interesting" mathematical knowledge.
 - help guide the exploration process.

- ☑ this approach (*algorithm schemes*)
 - provides good control to the user.
 - is more suitable to the exploration of more complex theories.
 - offer a methodology for theory exploration (with strong didactic value ☺).

- ☑ this system (*systematic use of knowledge schemes*)
 - should be used as a tool by mathematicians.
 - will be focused on problem solving (using the lazy thinking method).

1! 5. Conclusion

19.0.

Future work

- Implementation of tools:
 - to carry out exploration steps;
 - to query the libraries of schemes;
 - to query the language and the knowledge base.
- ... towards (semi-)automated theory exploration.