# 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, $\mathcal{T}=\langle\mathcal{L}, \mathcal{K}\mathrm{B}, \mathcal{I}\mathcal{R}\rangle$

$\mathcal{L}$ - 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.

$\mathcal{K}\mathrm{B}$ - knowledge base (formulae).

$\mathcal{I}\mathcal{R}$ - 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, $\mathcal{T}_{\mathcal{N}} = < \mathcal{L}_{\mathcal{N}} , \mathcal{KB}_{\mathcal{N}} , \mathcal{IR}_{\mathcal{N}} >$

$$\mathcal{L}_{\mathcal{N}} = \langle\langle \text{is-nat}, =\rangle, \langle ^{+} , \text{id} \rangle\rangle, \langle 0\rangle\rangle.$$

$\mathcal{KB}_{\mathcal{N}}$: equality axioms and Peano axioms.

Axioms["Peano axioms",

is-natural[0] "nat:generati

$\underset{\text{is-natural}[x]}{\forall}$ is-natural[$x^{+}$] "nat:generati

$\underset{\text{is-natural}[x]}{\forall}$ $x^{+} \neq 0$ "nat:uniquen

$\underset{\text{is-natural}[x,y]}{\forall}$ $(x^{+} = y^{+} \Rightarrow x = y)$ "nat:uniquen

$\left(\mathcal{F}[0] \bigwedge \underset{\text{is-natural}[x]}{\forall} (\mathcal{F}[x] \Rightarrow \mathcal{F}[x^{+}])\right) \Rightarrow \underset{\text{is-natural}[x]}{\forall} \mathcal{F}[x]$ "induction pı

]

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

$\mathcal{IR}_{\mathcal{N}}$=

- ***structural induction*** rule
- ***general predicate logic inference*** rules
- ***equality inference*** 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.

### 4.1.1 I. Knowledge schemes dependent on the theory being developed (in our case: natural number theory)

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

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

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

# 5 1. Introduction - context for the case study

### 5.0.1 II. Knowledge schemes independent of any theory

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

$$\underset{p,bin\text{-}op,zero}{\forall}\left(\text{is-monoid}[p, \text{bin-op}, \text{zero}] \Leftrightarrow \text{is-semigroup}[p, \text{bin-op}] \bigwedge \underset{p[x]}{\forall}(\text{bin-op}[x, \text{zero}] = x)\right)$$

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

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

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

## 6 2. Exploration model and examples

### 6.0.1 Exploration consistent with the model 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
```

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

```
possibleSubsts :=
  {{f → •[Plus], g → •[Identity], h → •[SuperPlus]},
    {f → •[Plus1], g → •[SuperPlus], h → •[SuperPlus]},
    {f → •[ProjLeft], g → •[Identity], h → •[Identity]},
    {f → •[ProjLeftPlus1], g → •[SuperPlus], h → •[Identity]}}
```

```
newConcepts :=
  Map[UseScheme[is-rec-nat-binary-fct-1r, #] &, possibleSubsts]
```

```
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
```

$$\underset{f,g,h}{\forall}\left(\text{is-rec-nat-binary-fct-11}[f,g,h] :\Leftrightarrow \underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]}}{\forall}\left((f[0,y]=g[y]) \wedge \left(f\left[x^{+},y\right]=h[f[x,y]]\right)\right)\right)$$

```
possibleSubsts :=
  {{f → •[Plus], g → •[Identity], h → •[SuperPlus]},
    {f → •[Plus1], g → •[SuperPlus], h → •[SuperPlus]},
    {f → •[ProjRight], g → •[Identity], h → •[Identity]},
    {f → •[ProjRightPlus1], g → •[SuperPlus], h → •[Identity]}}
```

```
newConcepts :=
  Map[UseScheme[is-rec-nat-binary-fct-11, #] &, possibleSubsts]
```

```
newConcepts
```

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

Proposition["nat.1.2:new simple recursion: 2Identity™SuperPlus",
  any[is-natural[x], is-natural[y]],

$$\left.\begin{array}{r} 0+y=y \\ x^{+}+y=(x+y)^{+} \end{array}\right]$$

ProofNewAddition

ProofNextNewAddition

# 9 2. Exploration model and examples

**9.0.1** **<u>Introducing propositions (II) - semigroup, monoid, group:</u> we look at the algebraic knowledge schemes that can be instantiated with the + function symbol and the other symbols from the language.**

### 9.0.2 <u>Semigroup algebraic structure</u>

❀ the first algebraic scheme instantiated with {p→•[IsNatural], bin-op→•[+]} gives :

schSemigroup /. {p → •[IsNatural], bin-op → •[+]}

$$\left\{\text{is-semigroup}[{}^{\text{TM}}\text{IsNatural}, +] \;:\Leftrightarrow \underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]\\ \text{is-natural}[z]}}{\forall} \left(\text{is-natural}[x+y] \wedge (x+(y+z) = (x+y)+z)\right)\right\}$$

```
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 <u>Monoid algebraic structure</u>

❀ the next scheme instantiated with {p→•[IsNatural], bin-op→•[+], zero→•[0]} gives:

schMonoid /. {p → •[IsNatural], bin-op → •[+], zero → •[0]}

$$\left\{\text{is-monoid}[{}^{\text{TM}}\text{IsNatural}, +, 0] \;:\Leftrightarrow\; \text{is-semigroup}[{}^{\text{TM}}\text{IsNatural}, +] \bigwedge \underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]\\ \text{is-natural}[z]}}{\forall} (x+0 = x)\right\}$$

❀ 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

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

❁ the possible substantiations 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]

$$\{\text{is-group}[{}^{TM}\text{IsNatural}, +, 0, \text{Identity}] :\Leftrightarrow \text{is-monoid}[{}^{TM}\text{IsNatural}, +, 0] \bigwedge \left(\underset{\text{is-natural}[x]}{\forall} (x + x) = 0\right),$$
$$\text{is-group}[{}^{TM}\text{IsNatural}, +, 0, {}^{TM}\text{SuperPlus}] :\Leftrightarrow \text{is-monoid}[{}^{TM}\text{IsNatural}, +, 0] \bigwedge \left(\underset{\text{is-natural}[x]}{\forall} (x + x^{+}) = 0\right)\}$$

❁ 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 (⊖), such that *is-group[· IsNatural,+,0, ⊖];*

(prove the formula $\underset{\text{is-natural}}{\forall} x + \ominus x = 0$).

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

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

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

# 1 2. Exploration model and examples

## 11.0.1 Multiplication Function Symbol.

- knowledge scheme used for inventing new notions:

schMultiplicationConstant

$$\underset{f,h,const}{\forall}\left(\text{is-rec-nat-binary-fct-0r}[f, const, h] :\Leftrightarrow \underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]}}{\forall}\left((f[x,0]=const)\wedge\left(f\left[x,y^{+}\right]=h[x,f[x,y]]\right)\right)\right)$$

- the substitution {f→•[Times], const→•[0], h→•[Plus]} gives:

schMultiplicationConstant /.
{f → •[Times], const → •[0], h → •[Plus]}

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

Definition["nat.2.1: simple recursion: 3™Times",
any[is-natural[x], is-natural[y]],
$x*0=0$
$x*y^{+}=x*y+x$]

## 11.0.2 Exponentiation Function Symbol.

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

- the substitution {f→•[Power], const→•[1], h→•[Times]} gives:

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

{is-rec-nat-binary-fct-0r[^, 1, ∗] :⇔ $\underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]}}{\forall}\left((x^0 = 1) \bigwedge (x^{y^+} = x * x^y)\right)$}

Definition["exponentiation.2.2", any[is-natural[x], is-natural[y]],
$x^0 = 1$
$x^{y^+} = x^y * x$]

### 11.0.3 Strict Less Than Equal Relation Symbol.

- knowledge scheme used for inventing new notions:

schRelationSymbol

$\underset{f,g,h,pred}{\forall}\Big($ is-nat-rec-binary-rel-2[$f$, $g$, $h$, $pred$] :⇔

$\underset{\substack{\text{IsNatural}[x]\\ \text{IsNatural}[y]\\ \text{IsNatural}[z]}}{\forall}\left((f[x, 0] \Leftrightarrow g[x] = 0) \wedge \left(f[x, y^+] \Leftrightarrow pred[x, h[y]] \vee f[x, y]\right)\right)\Big)$

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

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

{is-nat-rec-binary-rel-2[ < , ™SuperPlus, Identity, =] :⇔

$\underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]\\ \text{is-natural}[z]}}{\forall}\left((x < 0 \Leftrightarrow x^+ = 0) \wedge \left(x < y^+ \Leftrightarrow (x = y) \vee x < y\right)\right)$}

Definition["relation symbol.3.2: ™SuperPlusIdentity",
any[is-natural[x], is-natural[y]],
$(x < 0) \Leftrightarrow (x^{+} = 0)$
$(x < y^{+}) \Leftrightarrow ((x = y) \bigvee (x < y))$]

# 1: 2. Exploration model and examples

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

-introduce the proposition into the theory:

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

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

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

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

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

$$G_1 : \underset{\text{is-natural}[x]}{\forall} \mathcal{F}[x]$$

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

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

- our main goals are in this way the following:

$A_1 \Rightarrow P$ (Complete Induction.2)

$P \Rightarrow G_1$ (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. <u>Quotient/Remainder Function Symbols.</u>

- knowledge scheme used for inventing new concepts:

schQuotRemFunctSymbol

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

- original problem considered for solving:

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

- modified problem considered for solving:

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

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

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

Definition["nat.5.1: simple recursion less: quotient",

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

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

Definition["nat.5.2: simple recursion less: remainder",

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

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

**Divides Relation Symbol.**

- knowledge scheme used for inventing new concepts:

schDividesRelation

$$\left[\underset{r,p,q,const}{\forall}\left((\text{is-nat-step-recr-rel-0-1-1}[r, p, q, const]) :\Leftrightarrow\right.\right.$$

$$\left.\left.\left(\underset{\text{is-natural}[x,y]}{\forall}\left(r[x, y] \Leftrightarrow \begin{cases} const & \Leftarrow\ y = 0 \\ p[x, y] & \Leftarrow\ x > y \\ q[r[x, y - x]] & \Leftarrow\ \text{otherwise} \end{cases}\right)\right)\right)\right]$$

schDividesRelation

- the substitution {r→•[|], p→•[False], $q \to$ •[$id_B$], const→ •[True]} gives:

schDividesRelation/.
{r → •[/], p → •[False], q → •[$id_B$], const → •[True]}

$$\underset{/,False,True,True}{\forall}\left(\text{is-nat-step-recr-rel-0-1-1}[/, False, True, True] :\Leftrightarrow\right.$$

$$\left.\underset{\substack{\text{is-natural}[x]\\ \text{is-natural}[y]}}{\forall}(x / y \Leftrightarrow \|True \Leftarrow y = 0, False[x, y] \Leftarrow x > y, True[x / (y - x)] \Leftarrow \text{otherwise}\|)\right)$$

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

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

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

]

## 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

## 16. Related work

### 16.0.1 HR

Methodology: Examples $\xrightarrow{\text{Model Checking}}$ 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.

### 16.0.2 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

# 17. Conclusion

## 17.0.1 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).

## 17.0.2 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).

## 17.0.3 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 querry the language and the knowledge base.

  ... towards (semi-)automated theory exploration.