Systematic Exploration of Mathematical Theories

Mădălina Hodorog[1] Adrian Crăciun[2] Tudor Jebelean[1]

[1]Research Institute for Symbolic Computation
Johannes Kepler University Linz, Austria

[2]Institute eAustria Timișoara, Romania

mhodorog@risc.uni-linz.ac.at, acraciun@ieat.ro, tjebelean@risc.uni-linz.ac.at

Table of contents

Theory of Natural Numbers

- Language:
  - $\mathcal{L}_{\mathbb{N}} = \left\langle \langle \textit{is-nat}, = \rangle, \langle ^{+}, id \rangle, \langle 0 \rangle \right\rangle$,
    (predicates, functions, constants);
- Knowledge Base (KB):
  - equality axioms (lifted to inference);
  - Peano's axioms: generation and uniqueness axioms;
  - induction axiom (lifted to inference);
- Inference mechanism (inference rules):
  - rewriting lifted from equality axioms;
  - general predicate logic inference rules;
  - structural induction lifted from induction axiom;

Knowledge Schemes

- higher-order formulae ("interesting" mathematical knowledge);
- stored in libraries of schemes;
- used by instantiation (become first-order);

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

Knowledge Schemes

- higher-order formulae ("interesting" mathematical knowledge);
- stored in libraries of schemes;
- used by instantiation (become first-order);

- Example (Theory Dependent Schemes).

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

Knowledge Schemes

- Example (Theory Independent Schemes).

algebraic structures
is-semigroup
is-monoid
is-group
is-ring
is-unity-ring
is-integral-domain

relational structures
is-preorder
is-partial-ordering
is-total-partial-ordering
is-strict-partial-ordering
is-total-strict-partial-ordering

Knowledge Schemes

- Example (Theory Independent Schemes).

algebraic structures
is-semigroup
is-monoid
is-group
is-ring
is-unity-ring
is-integral-domain

relational structures
is-preorder
is-partial-ordering
is-total-partial-ordering
is-strict-partial-ordering
is-total-strict-partial-ordering

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

Knowledge Schemes

- Example (Theory Independent Schemes).

algebraic structures
is-semigroup
is-monoid
is-group
is-ring
is-unity-ring
is-integral-domain

relational structures
is-preorder
is-partial-ordering
is-total-partial-ordering
is-strict-partial-ordering
is-total-strict-partial-ordering

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

Scheme-Based Exploration Model (Buchberger)

Develop a theory in exploration rounds:

- introduce new notions (functions, predicates) with definition schemes;
- introduce and prove (or disprove) propositions about the notions with proposition schemes;
- introduce problems using algorithm schemes and solve them;
- introduce new inference rules by lifting knowledge or with inference schemes;

Scheme-Based Exploration Model (Buchberger)

Develop a theory in exploration rounds:

- introduce new notions (functions, predicates) with definition schemes;
- introduce and prove (or disprove) propositions about the notions with proposition schemes;
- introduce problems using algorithm schemes and solve them;
- introduce new inference rules by lifting knowledge or with inference schemes;

Example 1. Obtaining interesting functions

Introduce function symbols ($+$)

- $\underset{f,g,h}{\forall}(\textit{is-rec-nat-binary-fct-1r}[f,g,h] \Leftrightarrow \underset{\textit{is-nat}[x,y]}{\forall} \bigwedge \left\{ \begin{array}{l} f[x,0] = g[x] \\ f[x,y^{+}] = h[f[x,y]] \end{array} \right.)$,
- possible instantiations:
  $\{f \rightarrow \oplus, g \rightarrow id, h \rightarrow {}^{+}\}, \{f \rightarrow \boxplus, g \rightarrow {}^{+}, h \rightarrow {}^{+}\}$,
  $\{f \rightarrow \odot, g \rightarrow id, h \rightarrow id\}, \{f \rightarrow \boxdot, g \rightarrow {}^{+}, h \rightarrow id\}$
- for $\{f \rightarrow \oplus, g \rightarrow id, h \rightarrow {}^{+}\}$
- we obtain: $x \oplus 0 = x$
  $x \oplus y^{+} = (x \oplus y)^{+}$
- *Remark:* $\oplus$ is in fact $+$;

Example 1. Obtaining interesting functions

Introduce function symbols $(+)$

- $$\underset{f,g,h}{\forall}\left(\textit{is-rec-nat-binary-fct-1r}[f,g,h] \Leftrightarrow \underset{\textit{is-nat}[x,y]}{\forall} \bigwedge \left\{ \begin{array}{l} f[x,0] = g[x] \\ f[x,y^{+}] = h[f[x,y]] \end{array} \right. \right),$$
- possible instantiations:
  $\{f \to \oplus, g \to id, h \to {}^{+}\}, \{f \to \boxplus, g \to {}^{+}, h \to {}^{+}\},$
  $\{f \to \odot, g \to id, h \to id\}, \{f \to \boxdot, g \to {}^{+}, h \to id\}$

Example 1. Obtaining interesting functions

Introduce function symbols $(+)$

Properties	$+$	$\boxplus$	$\odot$	$\boxdot$
sort	$\surd$	$\surd$	$\surd$	$\surd$
associativity	$\surd$	$\times$	$\surd$	$\times$
commutativity	$\surd$		$\times$	
identity	$\surd$			
inverse	$\times$			

- introduce propositions:
  - *is-rec-nat-binary-fct-1l*$[+, id, {}^{+}]$ (an equivalent definition);
  - *is-semigroup*[*is-nat*, $+$] (the type and the associativity);
  - *is-monoid*[*is-nat*, $+, 0$];
  - *is-commutative-monoid*[*is-nat*, $+, 0$] (commutativity);
  - *is-group*[*is-nat*, $\oplus, 0, \ominus$] (use lazy thinking to synthesize $\ominus$, the problem has no solution);
  - which are automatically proven using the *Theorema* system;

Example 1. Obtaining interesting functions

Introduce function symbols (+)

Properties	$+$	$\boxplus$	$\odot$	$\boxdot$
sort	$\surd$	$\surd$	$\surd$	$\surd$
associativity	$\surd$	$\times$	$\surd$	$\times$
commutativity	$\surd$		$\times$	
identity	$\surd$			
inverse	$\times$			

- introduce propositions:
  - *is-rec-nat-binary-fct-1l*$[+, id, \ ^{+}]$ (an equivalent definition);
  - *is-semigroup*[*is-nat*, $+$] (the type and the associativity);
  - *is-monoid*[*is-nat*, $+, 0$];
  - *is-commutative-monoid*[*is-nat*, $+, 0$] (commutativity);
  - *is-group*[*is-nat*, $\oplus, 0, \ominus$] (use lazy thinking to synthesize $\ominus$, the problem has no solution);
  - which are automatically proven using the *Theorema* system;

Example 2. Inventing significant results

Problem Scheme: decomposition w.r.t $\otimes$

- $\underset{obj,p_1,p_2,p_3,\otimes}{\forall} FBDp[obj,p_1,p_2,p_3,\otimes] \Leftrightarrow \underset{\substack{obj[x,y] \\ p_1[x],p_2[y]}}{\forall} \; \underset{\substack{obj[z] \\ p_3[z]}}{\exists} x = y \otimes z.$
- for $\{obj \rightarrow \textit{is-nat}, p_1 \rightarrow true, p_2 \rightarrow true, p_3 \rightarrow true, \otimes \rightarrow *\}$:
  - $\underset{\textit{is-nat}[x,y]}{\forall} x = y * q[x,y]$,(1)
- for which we propose the solution algorithm:
  - $\underset{q,g,h}{\forall} \textit{is–nat–step–recl–fct–1–1}[q,g,h] \Leftrightarrow$
    $\underset{\substack{\textit{is–nat}[x,y] \\ y>0}}{\forall} q[x,y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[q[x-y,y]] & \Leftarrow \text{otherwise} \end{cases}$, (2)
- we prove (1) using (2).

Example 2. Inventing significant results

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

- first proof attempt: FAILURE!, but we change the problem:
  - $\underset{\substack{\textit{is-nat}[x] \\ \textit{is-positive}[y]}}{\forall} x = y * q[x, y]$
- second proof attempt: FAILURE!, but we change the problem:
  - $\underset{\substack{\textit{is-nat}[x] \\ \textit{is-positive}[y]}}{\forall} x = y * q[x, y] + r[x, y].$
- third proof attempt: summary of the proof.

Example 2. Inventing significant results

- Prove $\underset{\substack{\textit{is-nat}[x] \\ \textit{is-positive}[y]}}{\forall} x = y * q[x, y] + r[x, y]$, using:
  - $\textit{s1}[q, g, h] : \underset{\substack{\textit{is-nat}[x,y] \\ y>0}}{\forall} q[x, y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[q[x - y, y]] & \Leftarrow \text{otherwise} \end{cases},$
  - $\textit{s2}[r, k, t] : \underset{\substack{\textit{is-nat}[x,y] \\ y>0}}{\forall} r[x, y] = \begin{cases} k[x] & \Leftarrow x < y \\ t[r[x - y, y]] & \Leftarrow \text{otherwise} \end{cases},$
- ...details skipped, but in the proof we have to show:
- $y_0 * h[q[x_0 - y_0, y_0]] + t[r[x_0 - y_0, y_0]] =$
  $y_0 * (q[x_0 - y_0, y_0] + 1) + r[x_0 - y_0, y_0].$
- we generate the conjectures $h[x] = x + 1$, $t[y] = y$
- similarly we generate $g[x] = 0$, $k[x] = x$.

Example 2. Inventing significant results

- Prove $\underset{\substack{\textit{is-nat}[x] \\ \textit{is-positive}[y]}}{\forall} x = y * q[x, y] + r[x, y]$, using:
  - $\textit{s1}[q, g, h] : \underset{\substack{\textit{is–nat}[x,y] \\ y>0}}{\forall} q[x, y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[q[x-y, y]] & \Leftarrow \text{otherwise} \end{cases},$
  - $\textit{s2}[r, k, t] : \underset{\substack{\textit{is–nat}[x,y] \\ y>0}}{\forall} r[x, y] = \begin{cases} k[x] & \Leftarrow x < y \\ t[r[x-y, y]] & \Leftarrow \text{otherwise} \end{cases},$
- we obtain:
  - $g[x] = 0, h[x] = x + 1 \Rightarrow q[x, y] = \begin{cases} 0 & \Leftarrow x < y \\ q[x-y, y] + 1 & \Leftarrow \text{otherwise} \end{cases},$
  - $k[x] = x, t[y] = y \Rightarrow r[x, y] = \begin{cases} x & \Leftarrow x < y \\ r[x-y, y] & \Leftarrow \text{otherwise} \end{cases},$
- *Remark:*
  - $q \leftarrow$ *quotient function symbol*;
  - $r \leftarrow$ *remainder function symbol*

Summary

- Development of Natural Numbers
Functions	$^{+}$ id	$+$ $*$ $\wedge$	$^{-}$ $-$ $quot$ rem gcd
Predicates	*is-nat* $=$		$\leq$ $<$ *is-pos* $\mid$ $\triangleright$ is-prime
Constants	0	1	
Inference Rules	SI	CI	WFI
- Invention of Significant Results
  - Decomposition of natural numbers
  - $\Rightarrow$ quotient-remainder theorem
  - Decomposition w.r.t well-founded orderings.
  - $\Rightarrow$ prime decomposition theorem
- Further Work
  - apply the model for the exploration of the theory of integers

Summary

- Development of Natural Numbers
  -

Functions	$^{+}\ id$	$+\ *\ {}^{\wedge}$	$^{-}\ -\ quot\ rem\ gcd$
Predicates	*is-nat* $=$		$\leq\ <$ *is-pos* $\mid\ \triangleright$ is-prime
Constants	0	1	
Inference Rules	SI	CI	WFI

- Invention of Significant Results
  - Decomposition of natural numbers
  - $\Rightarrow$ quotient-remainder theorem
  - Decomposition w.r.t well-founded orderings.
  - $\Rightarrow$ prime decomposition theorem
- Further Work
  - apply the model for the exploration of the theory of integers

Implementation

In *Theorema*

- new provers:
  - NNIP (structural induction);
  - NNCIP (complete induction);
  - NNWFIP (well–founded induction w.r.t. $<, \triangleleft$);
- mathematical knowledge bases:
  - libraries of knowledge schemes;
  - exploration rounds in the theory of natural numbers (integers);
- additional tools:
  - the *UseScheme* function (invents new symbols);

Conclusion

- scheme-based exploration model is successfully applied;
- advantages of the model:
  - offer a metodology for theory exploration
    - ⇒ didactical and practical value.
  - leads to the invention of significant results;
  - significant potential for [semi]automated exploration/discovery of theories
    - ⇒ important for practical theorem proving and program verification.

Conclusion

- scheme-based exploration model is successfully applied;
- advantages of the model:
  - offer a metodology for theory exploration
    - $\Rightarrow$ didactical and practical value.
  - leads to the invention of significant results;
  - significant potential for [semi]automated exploration/discovery of theories
    - $\Rightarrow$ important for practical theorem proving and program verification.

Thank you for your attention.