

Scheme-based systematic exploration of mathematical theories. Case study: The Natural Numbers.

Mădălina Hodorog
Institute e-Austria Timișoara, Romania
mhodorog@ieat.ro

September 5, 2006

Abstract

This technical report contains the development of the natural numbers theory using the model proposed by Bruno Buchberger for scheme-based systematic exploration of mathematical theories. The exploration contains several steps referring at: the invention of mathematical notions, the invention of equivalent definitions using knowledge schemes, the invention of structural propositions using algebraic schemes, the invention of propositions resulting from the interaction of the notions already introduced in the theory.

1 Introduction

One of the models proposed for developing a theory is the model proposed by Bruno Buchberger ([Buchberger B.,2004]), which refers to the systematic exploration of mathematical theories, model which is illustrated next in this paper.

The systematic exploration of mathematical theories refers at developing a theory by adding new concepts to it. The exploration contains several phases of exploration, each phase of exploration consisting of the following steps:

- invention of mathematical notions (i.e. definitions for functions) using recursive knowledge schemes.
- invention and verification of propositions concerning the mathematical notions. There can be distinguished three types of propositions:
 - invention and verification of propositions (i.e equivalent definitions) resulting from the exploration of the definitions already introduced in the theory using new recursive knowledge schemes.
 - invention and verification of structural propositions using algebraic structures schemes.
 - invention and verification of propositions resulting from the interaction of the notions already introduced in the theory using function symbol schemes.

For illustrating the systematic exploration of the natural numbers theory we use the THEOREMA system, and the model proposed by Bruno Buchberger (AISC 2004) for mathematical theories exploration. Developing the natural numbers theory using the model proposed by Bruno Buchberger, consists in the above steps of systematic exploration which will be thoroughly illustrated next in this paper.

2 Exploration Round 0

Notion of a theory To develop the theory we use the first order predicate logic language with equality as it is formal enough to express all mathematical notions. For the theory that we want to develop we have to define three main concepts:

- the language, that consists of several symbols used to construct the formulae of the theory.
- the knowledge base, that contains the axioms and all the definitions and propositions introduced in the theory.
- the inference mechanism, which allows us to prove the propositions from the theory.

Natural Numbers Theory The initial step of the exploration uses the following symbols for the initial language :

$$L[0] := \bullet \text{language}[\bullet \text{constants}[0], \bullet \text{functions}[\bullet \text{SuperPlus}], \bullet \text{Identity}], \\ \bullet \text{predicates}[\bullet \text{is-natural}], \bullet [=]].$$

The current knowledge base contains the axioms of equality, the axioms of generation and uniqueness of the natural numbers and the axiom of mathematical induction:

$$(\mathcal{F}[0] \wedge \forall_{\text{is-natural}[x]} (\mathcal{F}[x] \rightarrow \mathcal{F}[x^+])) \rightarrow \forall_{\text{is-natural}[x]} \mathcal{F}[x]$$

Remark. In fact, the axioms of equality and the axioms of mathematical induction are axiom schemes and they are formulated in higher order predicate logic language. They will not be used in this form, but they will be lifted to the inference level.

The inference rules consist from the induction rule, the general rules of predicate logic and the rules of equality (simplification, rewriting). The inference mechanism which is used for the defined theory consists in implementing and using different provers from the THEOREMA system.

In the next steps of exploration we use the theory developed in the previous steps. At first, new definitions and propositions are introduced using definition schemes and recursive schemes and proved until all the possible variants are automatically explored and then finally added to the theory. For proving the propositions we use the algorithmic provers from the THEOREMA system and even implement new provers when needed. In this way, the results obtained at

each step of exploration are used for expanding the current knowledge base to a new knowledge base ([Buchberger B.,2004]).

Next we give some examples of how these steps of exploration really work in practice.

3 Exploration Round 1: New Function Symbol (addition)

3.1 Current Language Used for Exploration

The language used in the first step of the exploration is identical to the initial language because no new mathematical notions have been introduced in the theory at this moment. Therefore, the language at this step of the exploration has the following form:

$$L[1] \leftarrow L[0] := \bullet language[\bullet constants[0], \\ \bullet functions[\bullet SuperPlus], \bullet Identity]], \\ \bullet predicates[\bullet is-natural], \bullet [=]].$$

3.2 New Definition Introduced in the Theory

At this point of the exploration we intend to invent new mathematical concepts using the knowledge schemes and all the notions already introduced in the theory. We proceed in the following way: at first we invent the new definition using the recursive knowledge schemes. In order to do this we follow some specific steps:

-writing the general recursive scheme for inventing the new function symbol (or the new operation):

$$schAdditionFunct := \bullet [\forall_{f,g,h} (is-rec-add-nat-binary-funct[f, g, h] \Leftrightarrow \\ \forall_{is-natural[x,y]} \wedge \left\{ \begin{array}{l} (f[x, 0] = g[x]) \\ (f[x, y^+] = h[f[x, y]]) \end{array} \right\})]$$

-generating all the possible function symbols (operations) that are the result of the different possible combinations between all the symbols already introduced in the language of the theory. In order to do this, we use a special function that we implemented, the function named *UseScheme[scheme,substitutions]*, which takes as arguments the general knowledge scheme *scheme* already written and the list *substitutions* of all the possible combinations between all the current notions from the language and generates all the possible functions that can be constructed:

$$possibleSubsts := \\ \{ \{ f \rightarrow \bullet Plus, g \rightarrow \bullet Identity, h \rightarrow \bullet SuperPlus \}, \\ \{ f \rightarrow \bullet Plus1, g \rightarrow \bullet SuperPlus, h \rightarrow \bullet SuperPlus \}, \\ \{ f \rightarrow \bullet ProjLeft, g \rightarrow \bullet Identity, h \rightarrow \bullet Identity \}, \\ \{ f \rightarrow \bullet ProjLeftPlus1, g \rightarrow \bullet SuperPlus, h \rightarrow \bullet Identity \} \}$$

$$newConcepts := Map[UseScheme[schAdditionFunct, \#] \&, possibleSubsts]$$

As a remark we must note that after executing these instructions the variable *newConcepts* contains a list which represents all the possible function symbols that can be generated and it has the following form in the THEOREMA system:

newConcepts

$$\left(\begin{array}{l} (is-rec-add-nat-binary-funct[+, Identity, SuperPlus] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (x + 0 = x) \\ (x + y^+ = (x + y)^+) \end{array} \right\} \\ is-natural[y] \end{array} \right) \\ \\ \left(\begin{array}{l} (is-rec-add-nat-binary-funct[Plus1, SuperPlus, SuperPlus] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (Plus1[x, 0] = x^+) \\ (Plus1[x, y^+] = Plus1[x, y]^+) \end{array} \right\} \\ is-natural[y] \end{array} \right) \\ \\ \left(\begin{array}{l} (is-rec-add-nat-binary-funct[ProjLeft, Identity, Identity] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (ProjLeft[x, 0] = x) \\ (ProjLeft[x, y^+] = ProjLeft[x, y]) \end{array} \right\} \\ is-natural[y] \end{array} \right) \\ \\ \left(\begin{array}{l} (is-rec-add-nat-binary-funct[ProjLeftPlus1, SuperPlus, Identity] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (ProjLeftPlus1[x, 0] = x^+) \\ (ProjLeftPlus1[x, y^+] = ProjLeftPlus1[x, y]) \end{array} \right\} \\ is-natural[y] \end{array} \right)$$

-obtaining the traditional addition function symbol. The result of the previous operation is a list of several new function symbols that are generated using the indicated notions from the *possibleSubsts* variable. From this list of new function symbols we extract the first function, which is in fact the traditional operation of addition:

$$Def Addition := newConcepts[1]$$

At this moment we can introduce, for the time being manually, the new invented definition in the theory, using the 'Definition' key word from the THEOREMA system. This key word together with all the other key words from the THEOREMA system (such as: 'Proposition', 'Theorem', etc) are called in fact 'label management tools' ([Buchberger B., 2004]).

Here is an example of how such a declaration should be made in the THEOREMA system:

$$\begin{array}{l} Definition["nat 1.2: simple recursion: 1IdentityTMSuperPlus"], \\ \quad any[is-natural[x], is-natural[y]], \\ \quad x + 0 = x \\ \quad x + y^+ = (x + y)^+ \\] \end{array}$$

3.3 New Propositions Introduced in the Theory

Next in our exploration, we can continue with introducing three types of propositions in the theory: propositions referring to equivalent definitions, structural propositions and propositions resulting from the interaction between all the notions from the theory.

3.3.1 Propositions - Equivalent Definitions

Each time a new category of propositions is introduced, the following steps must be followed. At first inventing the new proposition, which reduces to:

- writing the general definition knowledge scheme for inventing a new proposition:

$$schNewAdditionFunct := \bullet \left[\forall_{f,g,h} (is-rec-newadd-nat-binary-funct[f, g, h] \Leftrightarrow \right. \\ \left. \forall_{is-natural[x,y]} \wedge \left\{ \begin{array}{l} (f[0, y] = g[y]) \\ (f[x^+, y] = h[f[x, y]]) \end{array} \right\} \right]$$

-generating all the possible function symbols (operations) that are the result of the different possible combinations between all the symbols already introduced in the theory:

$$possibleSubsts := \\ \{ \{ f \rightarrow \bullet[Plus], g \rightarrow \bullet[Identity], h \rightarrow \bullet[SuperPlus] \}, \\ \{ f \rightarrow \bullet[NewPlus1], g \rightarrow \bullet[SuperPlus], h \rightarrow \bullet[SuperPlus] \}, \\ \{ f \rightarrow \bullet[NewProjLeft], g \rightarrow \bullet[Identity], h \rightarrow \bullet[Identity] \}, \\ \{ f \rightarrow \bullet[NewProjLeftPlus1], g \rightarrow \bullet[SuperPlus], h \rightarrow \bullet[Identity] \} \}$$

$$newConcepts := \\ Map[UseScheme[schNewAdditionFunct, \#]\&, possibleSubsts]$$

After executing these instructions the variable *newConcepts* contains a list which represents all the possible function symbols that can be generated and it has the following form in the THEOREMA system:

$$\left(\begin{array}{l} (is-rec-newadd-nat-binary-funct[+, Identity, SuperPlus] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (0 + y = y) \\ (x^+ + y = (x + y)^+) \end{array} \right\} \\ is-natural[y] \\ \\ (is-rec-newadd-nat-binary-funct[NewPlus1, SuperPlus, SuperPlus] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (NewPlus1[0, y] = y^+) \\ (NewPlus1[x^+, y] = NewPlus1[x, y]^+) \end{array} \right\} \\ is-natural[y] \\ \\ (is-rec-newadd-nat-binary-funct[NewProjLeft, Identity, Identity] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (NewProjLeft[0, y] = y) \\ (NewProjLeft[x^+, y] = NewProjLeft[x, y]) \end{array} \right\} \\ is-natural[y] \\ \\ (is-rec-newadd-nat-binary-funct[NewProjLeftPlus1, SuperPlus, Identity] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \left\{ \begin{array}{l} (NewProjLeftPlus1[0, y] = y^+) \\ (NewProjLeftPlus1[x^+, y] = NewProjLeftPlus1[x, y]) \end{array} \right\} \\ is-natural[y] \end{array} \right)$$

-obtaining a new addition operation (actually a new definition function symbol) which is the new proposition invented. From the previous generated list

of possible new function symbols, we extract the first invented function symbol, which represents the new proposition, equivalent definition for the addition function symbol:

DefNewAddition := newConcepts[1]

Secondly, we introduce the new invented proposition in the theory using the 'Proposition' key word from THEOREMA system:

*Proposition["nat 1.2: new simple recursion: 2IdentityTM SuperPlus"],
any[is-natural[x], is-natural[y]],
0 + y = y
x⁺ + y = (x + y)⁺
]*

Finally, we prove the new invented proposition using a new implemented induction prover from THEOREMA (which contains only the mathematical induction principle and the natural deduction rules), using the 'Prove' key word from THEOREMA system:

*Prove[Proposition["nat.1.2: new simple recursion:
2IdentityTM SuperPlus"]
, using → Theory["nat.1.2.0"], by → SimplerNatProverPC]*

The proof of this proposition is successful and it is being generated in a new Mathematica notebook in a very natural language closely related to the already known mathematical language used for proofs in general. For generating the proof, we can use other induction provers like: *NatProverPC* or *SimplerNatProverPC*. Both of these two provers lead us to a successful proof, but the second prover *SimplerNatProverPC* generates a shorter and more comprehensive proof.

3.3.2 Structural Propositions

We follow the same steps as exemplified previously for introducing the structural propositions. We will illustrate the main steps in inventing the propositions taking as an example the invention of the addition associativity proposition.

First of all, we introduce the new proposition in the theory:

-we write the general knowledge scheme, which in fact is an algebraic structure scheme and has the following form:

$$schSemigroup := \bullet [\forall_{p, bin-op} (is-monoid[p, bin-op] \Leftrightarrow \forall_{is-natural[x,y,z]} \bigwedge \left\{ \begin{array}{l} (p[bin-op[x, y]]) \\ (bin-op[x, bin-op[y, z]] = bin-op[bin-op[x, y], z]) \end{array} \right\})]$$

-we generate all the possible propositions that are the result of the different combinations between all the symbols already introduced in the language :

possibleSubsts := { {p → •[IsNatural], bin-op → •[+]} }

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

We can observe that the only possible result of the previous instruction is one single property, which is the required property for our defined theory: the associativity of the addition function symbol property. Now we can introduce the new proposition in the theory, using the key word 'Proposition' as follows:

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

and prove it using one of the induction algorithmic provers with the following construction:

```
Prove[Proposition["nat.1.2: associativity"],
      using → Theory["nat.1.2.1"], by → SimplerNatProverPC]
```

3.3.3 Propositions resulting from the interaction between the notions existing in the theory

We will give as an example of inventing this type of propositions the left cancellation property of the addition function symbol.

Inventing the proposition means:

-writing the general knowledge scheme, which in this case is a function symbol scheme:

$$schFunctSymbol := \bullet [\forall (funct-symbol[f] \Leftrightarrow \bigwedge_{is-natural[x,y,z]} \{ ((f[z,x] = f[z,y]) \Rightarrow (x = y)) \})]$$

-generating all the possible propositions that are the result of the different combinations between all the symbols already introduced in the language obtaining in this way the left cancellation property of addition:

$$possibleSubsts := \{ \{ f \rightarrow \bullet [+] \} \}$$

$$Map[UseScheme[schFunctSymbol, \#] \&, possibleSubsts]$$

Finally, we can introduce the left cancellation proposition of addition in the theory using the key word 'Proposition':

```
Proposition["nat 1.2: left cancellation of addition ",
            any[is-natural[x], is-natural[y], is-natural[z]],
            (z + x = z + y) ⇒ (x = y)
          ]
```

and prove it with the most appropriate induction algorithmic provers:

```
Prove[Proposition["nat.1.2: left cancellation of addition"],
      using → Theory["nat.1.2.4"], by → NatProver]
```

The proof is successful and it can be viewed in a new Mathematica notebook.

4 Exploration Round 2: New Function Symbol (multiplication)

4.1 Current Language Used for Exploration

Once a new symbol function (i.e. the addition function symbol) is introduced in the theory, the language used in the previous step of the exploration is expanded to a new language of the following form:

$$L[2] \leftarrow L[1] \cup \{+\} := \bullet language[\bullet constants[0], \\ \bullet functions[\bullet [SuperPlus], \bullet [Identity], \bullet [Plus]], \\ \bullet predicates[\bullet [is-natural], \bullet [=]]].$$

Next in our exploration, we will use the new expanded language, which is $L[2]$.

4.2 New Definition Introduced in the Theory

We begin again with the automatical invention of possible new mathematical definitions given the existing knowledge schemes and the new language of the theory.

To invent a new function symbol we proceed in the same manner as in the first step of the exploration:

-we write and use the following new recursive scheme:

$$schMultiplicationConst :=$$

$$\bullet [\quad \forall_{f, const, h} (is-rec-mult-nat-binary-const[f, const, h] \Leftrightarrow \\ \forall_{is-natural[x, y]} \wedge \left\{ \begin{array}{l} (f[x, 0] = const) \\ (f[x, y^+] = h[x, f[x, y]]) \end{array} \right\})]$$

-we generate all the possible function symbols that can be generated from the interaction between all the concepts introduced in the theory, using the same function *UseScheme*, which produces all the possible new function symbols:

$$possibleSubsts := \{ \{ f \rightarrow \bullet [Times], const \rightarrow \bullet [0], h \rightarrow \bullet [Plus] \} \}$$

$$newConcepts :=$$

$$Map[UseScheme[schMultiplicationConst, \#]\&, possibleSubsts]$$

As a remark we must note that after executing these instructions the variable *newConcepts* contains a list which represents all the possible function symbols that can be generated and it has the following form in the THEOREMA system:

newConcepts

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

$$\left\{ \begin{array}{l} (is-rec-mult-nat-binary-const[*], 0, +) \Leftrightarrow \\ \forall_{is-natural[x], is-natural[y]} \wedge \left\{ \begin{array}{l} (x * 0 = x) \\ (x * y^+ = x + x * y) \end{array} \right\} \end{array} \right\}$$

-we obtain the traditional multiplication function, by extracting from the previously generated list the symbol in which we are interested:

$$DefMultiplication := newConcepts[1]$$

After inventing the multiplication function symbol, we can easily introduced the definition of the new functions into the theory using the 'Definition' key word from the from THEOREMA system:

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

4.3 New Propositions Introduced in the Theory

As in the first step of the exploration, we present three types of propositions that can be introduced in the theory once we have already introduced several mathematical symbols: propositions-equivalent propositions, structural propositions and propositions resulting from the interaction of the symbols already introduced in the theory.

4.3.1 Propositions - Equivalent Definitions

For inventing the new proposition-equivalent definition for the multiplication function symbol, we follow the three ordinary steps:

-we write the general definition knowledge scheme for inventing a new proposition:

$$\begin{array}{l} schNewMultiplicationConst := \\ \bullet [\forall_{f, const, h} (is-rec-newmult-nat-binary-const[f, const, h] \Leftrightarrow \\ \forall_{is-natural[x, y]} \wedge \left\{ \begin{array}{l} (f[0, y] = const) \\ (f[x^+, y] = h[f[x, y], x]) \end{array} \right\})] \end{array}$$

-we generate all the possible function symbols(operations) that are the result of the different possible combinations between all the symbols already introduced in the language of the theory:

$$possibleSubsts := \{ \{ f \rightarrow \bullet[Times], const \rightarrow \bullet[0], h \rightarrow \bullet[Plus] \} \}$$

$$\begin{array}{l} newConcepts := \\ Map[UseScheme[schNewMultiplicationConst, \#] \&, possibleSubsts] \end{array}$$

-we obtain a new multiplication operation (actually a new definition function symbol) which is the new proposition invented:

newConcepts

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

Finally, we can introduce the new invented proposition in the language using the 'Proposition' key word from the THEOREMA system:

Proposition["nat.2.1: new simple recursion: 4TMTimes"],
any[*is-natural*[*x*], *is-natural*[*y*]],
 $0 * y = 0$
 $x^+ * y = x * y + x$
 $]$

In the end, we must prove the new introduced proposition in the theory using one of the implemented provers from the THEOREMA system which relies on the mathematical induction principle and whose name is the NatProver:

Prove[*Proposition*["nat.2.1: new simple recursion: 4TMTimes"],
, using \rightarrow *Theory*["nat.2.1.0"], *by* \rightarrow *NatProver*]

The proof of this proposition takes some time and it just proves the first part of the proposition, while the second part of the proposition cannot be proven. This is due to the provers from the THEOREMA system. In the attempt to solve this problem a new mathematical induction principle will be further introduced in the theory, the complete mathematical induction principle which will also be implemented in a new prover from the THEOREMA system.

4.3.2 Structural Propositions

Several structural propositions are invented and introduced at this point of the exploration as a result of the algebraic structures schemes. We will present the identity element property towards the multiplication function symbol. The existence of this proposition introduces a new constant symbol in the language of the theory. The identity element property towards the multiplication function symbol is introduced for verifying the monoid algebraic structure of the natural numbers versus the multiplication function symbol. This property is one of the three properties required for the monoid algebraic structure: the sort property, the associativity property and of course the identity element property.

In order to invent the identity element property towards the multiplication function symbol, we use the following algebraic structure scheme for the monoid algebraic structure:

$$\text{schMonoid} := \bullet \left[\begin{array}{l} \forall_{p, \text{bin-op}, \text{const}} (\text{is-monoid}[p, \text{bin-op}, \text{const}] \Leftrightarrow \\ \forall_{\text{is-natural}[x, y, z]} \wedge \left\{ \begin{array}{l} (p[\text{bin-op}[x, y]]) \\ (\text{bin-op}[x, \text{bin-op}[y, z]] = \text{bin-op}[\text{bin-op}[x, y], z]) \\ (\text{bin-op}[x, \text{const}] = x) \end{array} \right\} \end{array} \right]$$

Afterwards, we can introduce in the theory the required identity element property towards the multiplication function symbol using the key word 'Proposition' from the THEOREMA system in the following way:

$$\begin{array}{l} \textit{Proposition}["nat.2.1: identity element", \\ \textit{any}[is-natural[x]], \\ x * c = x \\] \end{array}$$

We can prove the above proposition using all the induction provers: the *NatProver*, *NatProverPC*, *SimplerNatProverPC*. The construction we need to use when generating the proof of the proposition is the following (for one of the algorithmic provers taken as an example):

$$\begin{array}{l} \textit{Prove}[\textit{Proposition}["nat.2.1: identity element", \\ , \textit{using} \rightarrow \textit{Theory}["nat.2.1.3"], \textit{by} \rightarrow \textit{NatProver}] \end{array}$$

When proving the new property (using the *NatProver* prove), we are looking for a unique constant with the given property (i.e. $x*c=x$). In the automatically generated proof the searched constant is proven to be *constant*= 0⁺ (during the proof this formula becomes a temporary goal of the proof). This constant is abbreviated with the constant symbol 1, the identity element property of the multiplication function symbol being rewritten in the following way:

$$\begin{array}{l} \textit{Proposition}["nat.2.1: renamed identity element", \\ \textit{any}[is-natural[x]], \\ x * 1 = x \\] \end{array}$$

In this case, the proof fails. One of the reasons for this, could be that there are many assumptions and propositions introduced in the theory, which are difficult to manage by the already implemented provers from the THEOREMA system.

4.3.3 Propositions resulting from the interaction between the notions existing in the theory

In order to underline all the necessary steps for introducing a new proposition we take as an example the introduction of the left distributivity of multiplication versus addition proposition into the theory. The proposition is the result between the interaction of the two function symbols, the addition and the multiplication function symbols, and the two predicate symbols, the *is-natural[x]* and *equal(=)* predicate symbols. We introduce directly the proposition into the theory, using the 'Proposition' key word from the THEOREMA system:

$$\begin{array}{l} \textit{Proposition}["nat 2.1 : left distributivity of multiplication versus addition"], \\ \textit{any}[is-natural[x], is-natural[y], is-natural[z]], \\ (x + y) * z = x * z + y * z \\] \end{array}$$

After introducing the left distributivity of multiplication versus the addition proposition into the theory, we prove it using the *NatProver* from the THEOREMA system which implements the principle of mathematical induction and the rules of natural deduction:

Prove[
Proposition["nat.2.1: left distributivity of multiplication versus addition"]
, using \rightarrow *Theory*["nat.2.1.6.3"], by \rightarrow *SimplerNatProverPC*]

Unfortunately, in this case (as in the previous), the proof generated when using the *SimplerNatProverPC* prover is unsuccessful. The same thing happens when we want to prove the propositions using all the others provers that implements the mathematical induction principle: *NatProverPC* or *NatProver*. Next in our exploration, we should construct new induction provers in the *THEOREMA* system which should bring us to a satisfying result.

5 Exploration Round 3: New Relation Symbol (lessThan, strictLessThan)

5.1 Current Language Used for Exploration

The language from the second step of exploration is expanded to a new language of the theory once the new multiplication function symbol and the constant symbol 1 are introduced in the theory. It has the following form:

$$L[3] \leftarrow L[2] \cup \{*\} := \bullet \text{language}[\bullet \text{constants}[0,1], \\ \bullet \text{functions}[\bullet \text{SuperPlus}], \bullet \text{Identity}], \\ \bullet \text{Plus}], \bullet \text{Times}], \\ \bullet \text{predicates}[\bullet \text{is-natural}], \bullet [=]]]$$

5.2 New Definition Introduced in the Theory

After inventing and introducing two important function symbols in the theory, the addition function symbol and the multiplication function symbol, we propose to invent some new relation symbols in the theory. These new relation symbols are searched because after establishing important operation about the natural numbers, we would like to establish an order between these natural numbers, so as to arrange them after a given order.

We begin again with the automatical invention of possible new mathematical definitions about the desired relation symbols given the existing knowledge schemes written for the relation symbols and the new language of the theory.

To invent a new relation symbol we proceed in the same manner as in the previous steps of the exploration:

-we write and use the following new recursive scheme for the relation symbols:

$$\text{schRelationSymbol} := \\ \bullet [\quad \forall_{f,g,h,pred} (\text{is-simple-rec-relation-binary-symbol}[f, g, h, pred] \Leftrightarrow \\ \forall_{\text{is-natural}[x,y,z]} \wedge \left\{ \begin{array}{l} (f[x, 0] \Leftrightarrow g[x] = 0) \\ (f[x, y^+] \Leftrightarrow (pred[x, h[y]] \vee f[x, y])) \end{array} \right\})]]$$

-we generate all the possible relation symbols that can be generated from the interaction between all the concepts introduced in the theory, using the same function *UseScheme*, which produces all the possible new relation symbols:

$$\begin{aligned}
possibleSubsts := & \{ \{ f \rightarrow \bullet[LessEqual], g \rightarrow \bullet[Identity], \\
& h \rightarrow \bullet[SuperPlus], pred \rightarrow \bullet[Equal] \}, \\
& \{ f \rightarrow \bullet[StrictLessEqualPositive], g \rightarrow \bullet[Identity], \\
& h \rightarrow \bullet[Identity], pred \rightarrow \bullet[Equal] \}, \\
& \{ f \rightarrow \bullet[Less], g \rightarrow \bullet[SuperPlus], \\
& h \rightarrow \bullet[Identity], pred \rightarrow \bullet[Equal] \}, \\
& \{ f \rightarrow \bullet[LessEqualPositive], g \rightarrow \bullet[SuperPlus], \\
& h \rightarrow \bullet[SuperPlus], pred \rightarrow \bullet[Equal] \}
\end{aligned}$$

$$newConcepts := Map[UseScheme[schRelationSymbol, \#]\&, possibleSubsts]$$

As a remark we must note that after executing these instructions the variable *newConcepts* contains a list which represents all the possible relation symbols that can be generated and it has the following form in the THEOREMA system:

newConcepts

$$\left(\begin{array}{l}
(is-simple-rec-relation-binary-symbol[\leq, Identity, SuperPlus, =] \Leftrightarrow \\
\forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} (x \leq 0 \Leftrightarrow x = 0) \\ (x \leq y^+ \Leftrightarrow ((x = y^+) \vee (x \leq y))) \end{array} \right\} \\
\\
(is-simple-rec-relation-binary-symbol[StrictLessEqualPositive, \\
Identity, Identity, =] \Leftrightarrow \\
\forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} (StrictLessEqualPositive[x, 0] \Leftrightarrow x = 0) \\ (StrictLessEqualPositive[x, y^+] \\ \Leftrightarrow ((x = y) \vee (StrictLessEqualPositive[x, y]))) \end{array} \right\} \\
\\
(is-simple-rec-relation-binary-symbol[<, SuperPlus, Identity, =] \Leftrightarrow \\
\forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} (x < 0 \Leftrightarrow x^+ = 0) \\ (x < y^+ \Leftrightarrow ((x = y) \vee (x < y))) \end{array} \right\} \\
\\
(is-simple-rec-relation-binary-symbol[LessEqualPositive, \\
SuperPlus, SuperPlus, =] \Leftrightarrow \\
\forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} (LessEqualPositive[x, 0] \Leftrightarrow x^+ = 0) \\ (LessEqualPositive[x, y^+] \\ \Leftrightarrow ((x = y^+) \vee (LessEqualPositive[x, y]))) \end{array} \right\}
\end{array} \right)$$

We must mention that between all these four possible relation symbols that are generated using the relation symbol knowledge scheme and the notions existing in the theory, we obtain the following traditional relation symbols: the first invented relation symbol is in fact the *LessThanEqual* relation symbol, the second invented relation symbol is the *StrictLessThanEqual* relation symbol only for positive numbers, the third invented relation symbol is the *Strict-LessThanEqual* relation symbol, and the last invented relation symbol is the *LessThanEqual* relation symbol for positive numbers.

-we obtain the traditional *LessThanEqual* relation symbol, by extracting from the previously generated list the relation symbol in which we are interested:

$$Def LessThan := newConcepts[1]$$

-in the same way we obtain the traditional *StrictLessThanEqual* relation symbol, by extracting from the generated relation symbol list *newConcepts* the second relation symbol invented which in fact represent exactly the desired *StrictLessThanEqual* relation symbol:

$$DefStrictLessThan := newConcepts[3]$$

After inventing the *LessThanEqual* and *StrictLessThanEqual* relation symbols, we can easily introduced the definitions for the two new invented relation symbols in the theory using the 'Definition' key word from the from THEOREMA system:

$$\begin{array}{l}
\text{Definition["relation symbol.3.1: Identity}^{TM}\text{SuperPlus"}, \\
\quad \text{any[is-natural[x], is-natural[y]],} \\
\quad (x \leq 0) \Leftrightarrow (x = 0) \\
\quad (x \leq y^+) \Leftrightarrow ((x = y^+) \vee (x \leq y)) \\
]
\end{array}$$

$$\begin{array}{l}
\text{Definition["relation symbol.3.2: }^{TM}\text{SuperPlusIdentity"}, \\
\quad \text{any[is-natural[x], is-natural[y]],} \\
\quad (x < 0) \Leftrightarrow (x^+ = 0) \\
\quad (x < y^+) \Leftrightarrow ((x = y) \vee (x < y)) \\
]
\end{array}$$

5.3 New Propositions Introduced in the Theory

5.3.1 Propositions - Equivalent Definitions

When studying the propositions - equivalent definitions for the *LessThanEqual* and *StrictLessThanEqual* relation symbols, we begin with writing new knowledge definition schemes for inventing new mathematical notions(in this case new definitions).

Unforcently writing these new knowledge definitions schemes cannot contribute to inventing new propositions which cannot be proven in our defined theory.In this way, at this point of the exploration, no new propositions can be introduced in the theory.

5.3.2 Structural Propositions

For introducing new structural propositions regarding the *LessThanEqual* and *StrictLessThanEqual* relation symbols, we follow the relation knowledge schemes in a hierarchical way. We try to verify each existing knowledge scheme in order to invent new propositions with respect to the *LessThanEqual* and *StrictLessThanEqual* relation symbols. We follow the same steps which have to be verified when trying to introduce any new proposition,just that some propositions can be introduced after all the steps have been corectly executed,while others propositions cannot.

We will give as an example the introduction of two structural propositions: the symmetry of the *LessThanEqual* relation symbol and the transitivity of the *StrictLessThanEqual* relation symbol.

For inventing the symmetry of the *LessThanEqual* relation symbol property, we follow the three regular steps:

-we write the general structural knowledge scheme for inventing the new proposition:

$$schPreorder := \bullet[\forall_r(is-preorder[r] \Leftrightarrow \forall_{is-natural[x,y,z]} \wedge \left\{ \begin{array}{l} (r[x,x]) \\ ((r[x,y]) \wedge (r[y,z])) \Rightarrow (r[x,z]) \end{array} \right\})]$$

-we generate all the possible relation symbols that are the result of the different possible combinations between all the symbols already introduced in the language of the theory:

$$possibleSubsts := \{ \{r \rightarrow \bullet[\leq]\}, \{ \{r \rightarrow \bullet[<]\} \}$$

$$Map[UseScheme[schPreorder, \#]\&, possibleSubsts]$$

The list of all the possible generated combinations has the following form:

$$\left\{ \begin{array}{l} (is-preorder[\leq] \Leftrightarrow \\ \forall_{is-natural[x], is-natural[y], is-natural[z]} \wedge \left\{ \begin{array}{l} (x \leq x) \\ (((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z)) \end{array} \right\}) \\ (is-preorder[<] \Leftrightarrow \\ \forall_{is-natural[x], is-natural[y], is-natural[z]} \wedge \left\{ \begin{array}{l} (x < x) \\ (((x < y) \wedge (y < z)) \Rightarrow (x < z)) \end{array} \right\}) \end{array} \right\}$$

We observe that among all the propositions generated in the list of possible generated relations, we obtain the proposition that we need, that is the reflexivity of the *LessThanEqual* relation symbol.

Finally, we can introduce the new invented proposition in the language using the key word from the THEOREMA system 'Proposition':

$$\begin{array}{l} Proposition["nat.3.1: reflexivity(\leq)", \\ \quad \quad \quad any[is-natural[x]], \\ \quad \quad \quad (x \leq x) \\] \end{array}$$

We try to prove the proposition using one of the provers that implements the mathematical induction principle. Unfortunately, no satisfying results are to be obtained and the causes seem to be the same as in the previous cases when no correct proof was generated.

$$\begin{array}{l} Prove[Proposition["nat.3.1: reflexivity(\leq)"], \\ using \rightarrow Theory["nat.3.1.0"], by \rightarrow SimplerNatProverPC] \end{array}$$

The same steps are to be followed when introducing the transitivity of the *StrictLessThanEqual* relation symbol property, except that we can use all the previous steps because the general structural knowledge scheme that is used for introducing the transitivity of the *StrictLessThanEqual* relation symbol proposition is the same as the knowledge scheme used in inventing the reflexivity of the *LessThanEqual* relation symbol.

Moreover, from the same list with all the possible combinations resulting from the *Preorder* knowledge scheme, we observe that we have already invented the needed transitivity proposition.

$$\left\{ \begin{array}{l} (is\text{-preorder}[\leq] \Leftrightarrow \\ \forall \\ is\text{-natural}[x], is\text{-natural}[y], is\text{-natural}[z] \wedge \left\{ \begin{array}{l} (x \leq x) \\ (((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z)) \end{array} \right\} \\ (is\text{-preorder}[\lt] \Leftrightarrow \\ \forall \\ is\text{-natural}[x], is\text{-natural}[y], is\text{-natural}[z] \wedge \left\{ \begin{array}{l} (x < x) \\ (((x < y) \wedge (y < z)) \Rightarrow (x < z)) \end{array} \right\} \end{array} \right\}$$

In this way, we can introduce the proposition into the theory, with the same key word from THEOREMA system, 'Proposition':

$$\begin{array}{l} Proposition["nat.3.2: transitivity(<)", \\ \quad any[is\text{-natural}[x], is\text{-natural}[y], is\text{-natural}[z]], \\ \quad ((x < y) \wedge (y < z)) \Rightarrow (x < z) \\] \end{array}$$

Finally, we try to prove the proposition using one of the algorithmic provers implemented in THEOREMA, but the proof is not successful:

$$\begin{array}{l} Prove[Proposition["nat.3.2: transitivity(<)"], \\ using \rightarrow Theory["nat.3.2.0"], by \rightarrow SimplerNatProverPC] \end{array}$$

5.3.3 Propositions resulting from the interaction between the notions existing in the theory

We illustrate two examples for introducing this type of propositions into the theory: the left addition of the *LessThanEqual* relation symbol proposition and the left zero of the *StrictLessThanEqual* relation symbol proposition.

The left addition of the *LessThanEqual* relation symbol proposition is invented as the interaction between the *Plus* function symbol, the *is-natural* and the *equal* predicate symbols and the *StrictLessThanEqual* relation symbol. We can introduce this proposition into the theory using the key word 'Proposition' from the THEOREMA system:

$$\begin{array}{l} Proposition["nat.3.1: left addition(\leq)", \\ \quad any[is\text{-natural}[x], is\text{-natural}[y]], \\ \quad ((x \leq y) \Leftrightarrow (\exists_{is\text{-natural}[z]} (x + z = y))) \\] \end{array}$$

We try to prove the proposition using the *SimplerNatProverPC* prover, but unfortunately the proof does not succeed:

$$\begin{array}{l} Prove[Proposition["nat.3.2: left zero(\leq)"], \\ using \rightarrow Theory["nat.3.2.6"], by \rightarrow SimplerNatProverPC] \end{array}$$

The second illustrated proposition, the left zero of the *StrictLessThanEqual* relation symbol proposition is discovered as the interaction between the *StrictLessThanEqual* relation symbol and the *is-natural* and the *zero* predicate symbols. We introduce the proposition in the following way:

$$\begin{array}{l} Proposition["nat.3.2: left zero(<)", \\ \quad any[is\text{-natural}[x]], \\ \quad (0 < x) \\] \end{array}$$

We try to prove the propositions using one of the provers but again we do not obtain a successful desired proof:

Prove[*Proposition*["nat.3.2: left zero(<)"],
using \rightarrow *Theory*["nat.3.2.6"], by \rightarrow *SimplerNatProverPC*]

6 Exploration Round 4: New Inference Mechanism

We have seen that several times in our proofs the results were not satisfying because the proofs automatically generated were not complete. This is due to the large amount of knowledge (or of the assumptions) which was introduced in the theory and which is used each time a proof is generated. Another reason for the unsuccessful proofs could also be the inference mechanism used in the implementation of the provers used to automatically generate the proofs. Because of this, we decide to implement a new inference rule for proving the different propositions from our theory. Moreover, using the *StrictLessThanEqual* (<) relation symbol from the developed theory, we can introduce and prove an alternative version of the induction principle, which is often more convenient to use: the complete induction principle for nonnegative integers.

In order to introduce the complete induction principle, we write the knowledge scheme so as to be familiar with the mechanism of the principle itself:

$$\left(\forall_{is-natural[x]} \left(\forall_{is-natural[x']} \left((x' < x \Rightarrow \mathcal{F}[x']) \Rightarrow (\mathcal{F}[x]) \right) \right) \right) \Rightarrow \forall_{is-natural[x]} \mathcal{F}[x]$$

We can now introduce the complete induction principle in the theory as a proposition which has to be proven using the traditional induction principle:

Proposition["Complete Induction",

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

]

The first attempt in proving the proposition called *CompleteInduction* is to use the already implemented induction provers from the `THEOREMA` system. Unfortunately, the proof is unsuccessful and so we are forced to adopt another approach in proving the *CompleteInduction* proposition: we try to prove an alternative goal obtained in the following way.

First of all, we observe that the *CompleteInduction* proposition has the following form $A_1 \Rightarrow G_1$, where

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

$$G_1 : \forall_{is-natural[x]} \mathcal{F}[x]$$

In order to prove this property, we prove an alternative property:

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

In this way, our main goals are the following:

$A_1 \Rightarrow P$ and

$P \Rightarrow G_1$

We begin with proving the first goal, which means that we have to prove the following proposition:

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

In order to prove this proposition using the induction principle we need the following two assumptions, implying the natural numbers properties referring to *successor* and *closure*:

Proposition[*"smallerThanSuccessor(<)"*, *any*[*is - natural*[x]],
 $x < x + 1$
]

Proposition[*"closure w.r.t +1"*, *any*[*is - natural*[x]],
 $is - natural[x + 1]$
]

Consequently, using these important assumptions we proof the *FirstProof-CompleteInduction* proposition by any implemented prover from the *THEOREMA* system which uses the rules of natural deduction for the high-order predicate logic as an inference mechanism:

Prove[*Proposition*[*" first complete induction "*],
using \rightarrow {*Proposition*[*"smallerThanSuccessor(<)"*],
Proposition[*" closure w.r.t +1"*]}, *by* \rightarrow *PC*]

Instead of the *PC* prover, we can also use the *PredicateProver* prover because they both use the same inference mechanism: the calculus of high-order predicate logic.

We continue with proving the second goal needed for proving the complete induction principle, which is formalized under the following proposition:

Proposition[*"second complete induction"*,
 $\left(\forall_{is-natural[x]} \left(\forall_{is-natural[z]} \left((z < x \Rightarrow \mathcal{F}[z]) \Rightarrow (\mathcal{F}[x]) \right) \right) \right) \Rightarrow$
 $\left(\forall_{is-natural[y]} \left(\forall_{is-natural[z]} (z < y \Rightarrow \mathcal{F}[z]) \right) \right)$
]

For proving this second goal, we need the following three important assumptions in the knowledge base, which are in fact two properties from the natural numbers theory:

Proposition[*"smallerThanSuccessor"*, *any*[x, y]],
 $(x < y + 1) \Rightarrow ((x < y) \vee (x = y))$
]

```

Proposition["right zero(<)", any[is - natural[x]],
  x ≠ 0
]

Proposition["new property(<)", any[is - natural[x], is - natural[y]],
  (x + 1 = y) ⇒ (x < y)
]

```

We also need to include among our assumptions the equality symmetry axiom, which has the following form:

```

Axiom["equality: symmetry", any[x, y],
  (x = y) ⇒ (y = x)
]

```

In order to prove the second proposed goal under the previous assumptions, we choose to use any implemented prover from the THEOREMA system which uses the induction principle as the inference mechanism:

```

Prove[Proposition["second complete induction "],
  using → {Axiom["equality : symmetry"], Proposition["rightzero(<)"],
  Proposition["smallerThanSucesor"], Proposition["closure w.r.t +1"],
  Proposition["new property(<)" ]}, by → NatProver]

```

However, the proof of this proposition fails. This is a real issue which must be solved in the development of the natural numbers theory. In order to advance in our exploration we must prove the new inference mechanism, which reduces at this moment to proving the *SecondCompleteInduction* proposition.

7 Implementation

We have introduced new provers in the system for proving the propositions from the theory being developed. The new implemented provers are the following: the **NNIP** induction prover and the **NNIPC** induction prover.

NNIP induction prover. This prover implements the induction over natural numbers with the new feature that we introduce the sort condition for the induction variable. This prover is used in: NatProver, NatProverPC, Simpler-NatProverPC.

NNIPC induction prover. This prover implements the complete induction over natural numbers and it is used in NewNatProver.

8 Improvements in the previous exploration rounds

8.1 Improvements in Exploration Round 2: New Function Symbols (exponentiation, predecessor, subtraction)

8.1.1 Exponentiation Function Symbol

Current Language Used for Exploration At this point of the exploration the language used has the following form:

$$\begin{aligned}
L[3] \leftarrow L[2] \cup \{*\} := & \bullet \text{language}[\bullet \text{constants}[0, 1], \\
& \bullet \text{functions}[\bullet \text{SuperPlus}, \bullet \text{Identity}], \\
& \bullet \text{Plus}, \bullet \text{Times}], \\
& \bullet \text{predicates}[\bullet \text{is-natural}, \bullet [=]]
\end{aligned}$$

New Definition Introduced in the Theory. We choose to introduce the exponentiation function symbol as a generalization for the multiplication function symbol that already exists in the theory. We follow the same steps as for the previous exploration rounds:

-we write the knowledge scheme used for inventing new function symbols:

$$\text{schExponentiationConst} :=$$

$$\begin{aligned}
& \bullet [\forall_{f, \text{const}, h} (\text{is-rec-exp-nat-binary-const}[f, \text{const}, h] \Leftrightarrow \\
& \forall_{\text{is-natural}[x, y]} \wedge \left\{ \begin{array}{l} (f[x, 0] = \text{const}) \\ (f[x, y^+] = h[f[x, y], x]) \end{array} \right\})]
\end{aligned}$$

-we generate all the possible function symbols that can be generated from the interaction between all the concepts introduced in the theory, using the implemented *UseScheme* function and the *schExponentiationConst* knowledge scheme:

$$\begin{aligned}
\text{possibleSubsts} := & \\
& \{ \{ f \rightarrow \bullet \text{Power}, \text{const} \rightarrow \bullet [1], h \rightarrow \bullet \text{Times} \}, \\
& \{ f \rightarrow \bullet \text{Power1}, \text{const} \rightarrow \bullet [1], h \rightarrow \bullet \text{Plus} \}, \\
& \{ f \rightarrow \bullet \text{Power2}, \text{const} \rightarrow \bullet [0], h \rightarrow \bullet \text{Times} \}, \\
& \{ f \rightarrow \bullet \text{Power3}, \text{const} \rightarrow \bullet [0], h \rightarrow \bullet \text{Plus} \} \}
\end{aligned}$$

$$\text{newConcepts} :=$$

$$\text{Map}[\text{UseScheme}[\text{schExponentiationConst}, \#] \&, \text{possibleSubsts}]$$

As a remark we must note that after executing these instructions the variable *newConcepts* contains a list which represents all the possible function symbols that can be generated and it has the following form in the THEOREMA system:

newConcepts

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

$$(is-rec-exp-nat-binary-const[Power1, 1, +] \Leftrightarrow \\ \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (Power1[x, 0] = 1) \\ (Power1[x, y^+] = Power1[x, y] + x) \end{array} \right.)$$

$$(is-rec-exp-nat-binary-const[Power2, 0, *] \Leftrightarrow \\ \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (Power2[x, 0] = 0) \\ (Power2[x, y^+] = Power2[x, y] * x) \end{array} \right.)$$

$$(is-rec-exp-nat-binary-const[Power3, 0, +] \Leftrightarrow \\ \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (Power3[x, 0] = 0) \\ (Power3[x, y^+] = Power3[x, y] + x) \end{array} \right.)$$

-we obtain the traditional exponentiation function symbol by extracting from the *newConcepts* list the symbol we are interested in:

$$DefExp := newConcepts[1]$$

After inventing the exponentiation function symbol, we can easily introduce the definition of this function symbol into the theory using the 'Definition' key word from the THEOREMA system:

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

New Propositions Introduced in the Theory. Because no new knowledge schemes for equivalent definitions can be written, there are no propositions-equivalent definitions concerning the exponentiation function symbol that can be introduced in the theory. Moreover, the exponentiation function symbol does not verify any algebraic structure so no structural propositions can be introduced in the theory. The only type of propositions that can be introduced are the propositions resulting from the interaction between all the notions that already exist in the theory.

Propositions resulting from the interaction between all the notions already introduced in the theory. The following propositions can be introduced in the theory as different interactions between all the notions that already exist in the language of the theory:

- the *sort* proposition of the exponentiation function symbol, as an interaction between the *is-natural* predicate symbol and the *exponentiation* function symbol.
- the *exponentiation one* proposition of the exponentiation function symbol, as an interaction between the *is-natural*, the *equal* predicate symbols, the *one (1)* constant symbol and the *exponentiation* function symbol.
- the *base zero* proposition of the exponentiation function symbol, as an interaction between the *is-natural*, the *equal* predicate symbols, the *zero (0)* constant symbol and the *exponentiation* function symbol.
- the *exponentiation plus* proposition of the exponentiation function symbol, as an interaction between the *is-natural*, the *equal* predicate symbols, the *plus*, the *times* function symbols and the *exponentiation* function symbol.
- the *exponentiation times* proposition of the exponentiation function symbol, as an interaction between the *is-natural*, the *equal* predicate symbols, the *times* function symbols and the *exponentiation* function symbol.

We will exemplify the introduction in the theory of the *exponentiation plus* proposition of the exponentiation function symbol. We introduce the proposition directly in the theory using the 'Definition' key word from the THEOREMA system:

$$\begin{array}{l} \text{Proposition["nat.2.2: exponentiation plus"],} \\ \quad \text{any[is-natural[x], is-natural[y]],} \\ \quad x^{y+z} = x^y * x^z \end{array}$$

After introducing the exponentiation plus proposition into the theory, we prove it using the *NatProver* prover from the THEOREMA system:

```
Prove[
Proposition["nat.2.2: exponentiation plus"],
using → {Axiom["nat:generation"],
  Definition["nat.1.2: simple recursion: 1IdentitySuperPlus"],
  Proposition["nat.1.2:new simple recursion:2IdentitySuperPlus"],
  Proposition["nat.2.1:simple recursion: 4Times"],
  Definition["nat.2.1: simple recursion: 3Times"], }
by → NatProver]
```

The proof is successful.

8.1.2 Predecessor Function Symbol.

Current Language Used for Exploration. The current language that we use for exploration has the following form:

$$\begin{array}{l} L[3] \leftarrow L[2] \cup \{\wedge\} := \bullet \text{language}[\bullet \text{constants}[0, 1], \\ \quad \bullet \text{functions}[\bullet [SuperPlus], \bullet [Identity], \\ \quad \quad \bullet [Plus], \bullet [Times], \bullet [\wedge]], \\ \quad \bullet \text{predicates}[\bullet [is-natural], \bullet [=]]] \end{array}$$

New Definition Introduced in the Theory. We introduce the predecessor function symbol in the theory because we need it when inventing the subtraction function symbol. We follow the same exploration steps as before:

-we write the definition knowledge scheme for introducing new function symbols in the theory:

schPredFunctSymbol :=

$$\bullet \left[\begin{array}{l} \forall_{f,g,h,const} (\text{pred-funct-symbol}[f, g, h, const] \Leftrightarrow \\ \forall_{is-natural[x]} \wedge \{ (f[g[x, const]] = h[x]) \} \end{array} \right]$$

-we generate all the possible combinations that can be constructed using the *schPredFunctSymbol* knowledge scheme and all the notions already introduced in the theory:

possibleSubsts :=

$$\begin{aligned} & \{ \{ f \rightarrow \bullet[\text{SuperMinus}], g \rightarrow \bullet[\text{Plus}], h \rightarrow \bullet[\text{Identity}], const \rightarrow \bullet[1] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus1}], g \rightarrow \bullet[\text{Plus}], h \rightarrow \bullet[\text{SuperPlus}], const \rightarrow \bullet[1] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus2}], g \rightarrow \bullet[\text{Times}], h \rightarrow \bullet[\text{Identity}], const \rightarrow \bullet[1] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus3}], g \rightarrow \bullet[\text{Times}], h \rightarrow \bullet[\text{SuperPlus}], const \rightarrow \bullet[1] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus4}], g \rightarrow \bullet[\text{Plus}], h \rightarrow \bullet[\text{Identity}], const \rightarrow \bullet[0] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus5}], g \rightarrow \bullet[\text{Plus}], h \rightarrow \bullet[\text{SuperPlus}], const \rightarrow \bullet[0] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus6}], g \rightarrow \bullet[\text{Times}], h \rightarrow \bullet[\text{Identity}], const \rightarrow \bullet[0] \}, \\ & \{ f \rightarrow \bullet[\text{SuperMinus7}], g \rightarrow \bullet[\text{Times}], h \rightarrow \bullet[\text{SuperPlus}], const \rightarrow \bullet[0] \} \end{aligned}$$

newConcepts :=

$$\text{Map}[\text{UseScheme}[\text{schPredFunctSymbol}, \#] \&, \text{possibleSubsts}]$$

As a remark we must note that after executing these instructions the variable *newConcepts* contains a list which represents all the possible function symbols that can be generated and it has the following form in the THEOREMA system:

newConcepts

$$(pred-funct-symbol[{}^TM SuperMinus, +, Identity, 1] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ ((x + 1)^- = x) \})$$

$$(pred-funct-symbol[SuperMinus1, +, {}^TM SuperPlus, 1] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus1[x + 1] = x^+) \})$$

$$(pred-funct-symbol[SuperMinus2, *, Identity, 1] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus2[x * 1] = x) \})$$

$$(pred-funct-symbol[SuperMinus3, *, {}^TM SuperPlus, 1] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus3[x * 1] = x^+) \})$$

$$(pred-funct-symbol[SuperMinus4, +, Identity, 0] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus4[x + 0] = x) \})$$

$$(pred-funct-symbol[SuperMinus5, +, {}^TM SuperPlus, 0] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus5[x + 0] = x^+) \})$$

$$(pred-funct-symbol[SuperMinus6, *, Identity, 0] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus6[x * 0] = x) \})$$

$$(pred-funct-symbol[SuperMinus7, *, {}^TM SuperPlus, 0] \Leftrightarrow \forall_{is-natural[x]} \wedge \{ (SuperMinus7[x * 0] = x^+) \})$$

-we obtain the predecessor function symbol by extracting from the *newConcepts* list the symbol we are interested in:

$$DefPredecessor := newConcepts[1]$$

After inventing the predecessor function symbol, we can easily introduce the definition of this function symbol into the theory using the 'Definition' key word from the THEOREMA system:

$$\begin{aligned} & \text{Definition[\"predecessor.2.3\",} \\ & \quad \text{any[is-natural[x]],} \\ & \quad (x + 1)^- = x \end{aligned}$$

New Propositions Introduced in the Theory. No new propositions-equivalent definitions can be introduced in the theory because no new knowledge schemes for introducing equivalent propositions can be written concerning the *predecessor* function symbol. Furthermore, no new structural propositions are introduced in the theory, because the *predecessor* function symbol does not verify any existing algebraic structure.

Propositions resulting from the interaction between all the notions already introduced in the theory. The only type of propositions concerning the *predecessor* function symbol are the propositions resulting from the interaction between all the notions introduced in the language of the theory. The following propositions can be introduced:

- the *sort* proposition of the predecessor function symbol, as an interaction between the *is-natural* predicate symbol and the *predecessor* function symbol.
- the *decomposition* proposition of the predecessor function symbol, as an interaction between the *is-natural*, the *equal* predicate symbols, the *one (1)* constant, the *addition* and the *predecessor* function symbol.

We will exemplify the introduction in the theory of the *decomposition* proposition of the predecessor function symbol. We introduce the proposition directly in the theory:

Proposition["nat.2.3: decomposition of predecessor"],
 any[is-natural[x]],
 $x = x^- + 1]$

After introducing the decomposition proposition of the predecessor function symbol into the theory, we prove it using the *NatProver* prover from the THEOREMA system:

```
Prove[
Proposition["nat.2.3: decomposition of predecessor"],
using → {Axiom["nat:generation"],
  Definition["nat.1.2: simple recursion: 1IdentitySuperPlus"],
  Proposition["nat.1.2:new simple recursion:2IdentitySuperPlus"],
  Proposition["nat.2.1:simple recursion: 4Times"],
  Definition["nat.2.1: simple recursion: 3Times"],
  Definition["predecessor.2.3"], }
by → NatProver]
```

The proof is successful.

8.1.3 Subtraction Function Symbol

Current Language Used for Exploration The current language that we use for exploration has the following form:

$$L[3] \leftarrow L[2] \cup \{ \text{ } \} := \bullet \text{language}[\bullet \text{constants}[0, 1],$$

$$\bullet \text{functions}[\bullet [\textit{SuperPlus}], \bullet [\textit{Identity}],$$

$$\bullet [\textit{Plus}], \bullet [\textit{Times}], \bullet [\textit{^}], \bullet [\textit{-}]],$$

$$\bullet \text{predicates}[\bullet [\textit{is-natural}], \bullet [=]]]$$

New Definition Introduced in the Theory We introduce the subtraction function symbol in the theory, because we need the addition proposition of this function symbol for proving propositions that will be introduced later in the theory using the complete induction principle. For example the *quotient-remainder* proposition can be proved using the complete induction principle

only if the addition proposition of the subtraction function symbol is already introduced in the theory. For introducing the subtraction function symbol, we follow the exploration steps:

-we write the knowledge scheme used for introducing new function symbols:

schSubtractFunctSymbol :=

$$\bullet [\underset{f,g,h}{\forall} (\text{subtract-funct-symbol}[f, g, h] \Leftrightarrow \underset{is-natural[x,y]}{\forall} \wedge \left\{ \begin{array}{l} (f[x, 0] = h[x]) \\ (f[g[x], y^+] = f[x, y]) \end{array} \right\})]$$

-we generate all the possible combinations that can be constructed using the *schSubtractFunctSymbol* knowledge scheme and all the notions already introduced in the theory:

possibleSubsts :=

$$\{ \{ f \rightarrow \bullet[\text{Subtract}], g \rightarrow \bullet[\text{SuperPlus}], h \rightarrow \bullet[\text{Identity}] \}, \\ \{ f \rightarrow \bullet[\text{Subtract1}], g \rightarrow \bullet[\text{SuperPlus}], h \rightarrow \bullet[\text{SuperPlus}] \}, \\ \{ f \rightarrow \bullet[\text{Subtract2}], g \rightarrow \bullet[\text{Identity}], h \rightarrow \bullet[\text{Identity}] \}, \\ \{ f \rightarrow \bullet[\text{Subtract3}], g \rightarrow \bullet[\text{Identity}], h \rightarrow \bullet[\text{SuperPlus}] \} \}$$

newConcepts :=

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

As a remark we must note that after executing these instructions the variable *newConcepts* contains a list which represents all the possible function symbols that can be generated and it has the following form in the THEOREMA system:

newConcepts

$$(\text{subtract-funct-symbol}[-, {}^{TM}\text{SuperPlus}, \text{Identity}] \Leftrightarrow \underset{is-natural[x,y]}{\forall} \wedge \left\{ \begin{array}{l} (x - 0 = x) \\ (x^+ - y^+ = x - y) \end{array} \right\})$$

$$(\text{subtract-funct-symbol}[\text{Subtract1}, {}^{TM}\text{SuperPlus}, {}^{TM}\text{SuperPlus}] \Leftrightarrow \underset{is-natural[x,y]}{\forall} \wedge \left\{ \begin{array}{l} (\text{Subtract1}[x, 0] = x^+) \\ (\text{Subtract1}[x^+, y^+] = \text{Subtract1}[x, y]) \end{array} \right\})$$

$$(\text{subtract-funct-symbol}[\text{Subtract2}, \text{Identity}, \text{Identity}] \Leftrightarrow \underset{is-natural[x,y]}{\forall} \wedge \left\{ \begin{array}{l} (\text{Subtract2}[x, 0] = x) \\ (\text{Subtract2}[x, y^+] = \text{Subtract2}[x, y]) \end{array} \right\})$$

$$(\text{subtract-funct-symbol}[\text{Subtract3}, \text{Identity}, {}^{TM}\text{SuperPlus}] \Leftrightarrow \underset{is-natural[x,y]}{\forall} \wedge \left\{ \begin{array}{l} (\text{Subtract3}[x, 0] = x^+) \\ (\text{Subtract3}[x, y^+] = \text{Subtract3}[x, y]) \end{array} \right\})$$

-we obtain the subtraction function symbol, by extracting from the *newConcepts* list the symbol we are interested in:

DefSubtract := *newConcepts*[1]

After inventing the subtraction function symbol, we can easily introduce the definition of this function symbol into the theory using the 'Definition' key word from the THEOREMA system:

Definition["subtraction.2.4",
 $any[is-natural[x], is-natural[y]],$
 $x - 0 = x$
 $x^+ - y^+ = x - y]$

New Propositions Introduced in the Theory No new propositions-equivalent definitions can be introduced in the theory because no knowledge schemes for inventing equivalent-propositions can be written concerning the *subtraction* function symbol. Furthermore, no new structural propositions are introduced in the theory, because the *subtraction* function symbol does not verify any existing algebraic structure.

Propositions resulting from the interaction between all the notions already introduced in the theory. The only type of propositions concerning the *subtraction* function symbol are the propositions resulting from the interaction between all the notions introduced in the language of the theory.

The following propositions can be introduced in the theory as different interactions between all the notions already introduced in the theory:

- the *right one* proposition of the subtraction function symbol, that results as an interaction between the *SuperMinus*, the *Minus* function symbols and the *is-natural* and *equal* predicate symbols.
- the *addition* proposition of the subtraction function symbol, that results as an interaction between the *SuperMinus*, the *Plus* function symbols, the *is-natural* and the *equal* predicate symbols and the *one (1)* constant symbol.

We will exemplify the introduction into the theory of the *addition* proposition of the subtraction function symbol.

We introduce the proposition directly into the theory using the 'Definition' key word from the THEOREMA system:

Proposition["nat.2.4: addition of subtraction"],
 $any[is-natural[x], is-natural[y]],$
 $(x+y) - y = x]$

After introducing the addition proposition of the subtraction function symbol into the theory, we prove it using the *NatProver* prover from the THEOREMA system:

```

Prove[
Proposition["nat.2.4: addition of subtraction"],
using → {Axiom["nat:generation"],
  Definition["nat.1.2: simple recursion: 1IdentitySuperPlus"],
  Proposition["nat.1.2:new simple recursion:2IdentitySuperPlus"],
  Proposition["nat.2.1:simple recursion: 4Times"],
  Definition["nat.2.1: simple recursion: 3Times"],
  Definition["predecessor.2.3"],
  Definition["subtraction.2.4"]}
by → NatProver]

```

The proof is succesful.

9 Exploration Round 5: New Function Symbols (quotient, remainder)

9.1 Current Language Used for Exploration

The current language used for exploration contains the language from the previous step of the exploration and the new introduced *lessThan* and *strictLessThan* relation symbols:

$$L[4] \leftarrow L[3] \cup \{\leq, <\} := \bullet \text{language}[\bullet \text{constants}[0, 1],$$

$$\bullet \text{functions}[\bullet \text{SuperPlus}, \bullet \text{Identity},$$

$$\bullet \text{Plus}, \bullet \text{Times}, \bullet \text{Power},$$

$$\bullet \text{SuperMinus}, \bullet \text{Minus}],$$

$$\bullet \text{predicates}[\bullet \text{is-natural}, \bullet \text{=}, \bullet \text{[}\leq\text{]}, \bullet \text{[}\<\text{]}]]$$

9.2 New Definitions Introduced in the Theory

Using new knowledge schemes we should be able to invent new function symbols. In order to introduce new function symbols in the theory, we follow the same steps previously described:

- we write the definition knowledge schemes used for inventing new function symbols. This schemes are recursive knowledge schemes having the recursive step incremented by y:

$$schRemFunctSym := \bullet [\forall_{f,g,h} (rem-funct-symbol[f, g, h] \Leftrightarrow$$

$$\forall_{\substack{is-nat[x,y] \\ y>0}} (f[x, y] = \begin{cases} g[x] & \Leftarrow x < y \\ h[f[x - y, y]] & \Leftarrow \text{otherwise} \end{cases})].$$

$$schQuotFunctSym := \bullet [\forall_{f,const,h} (quot-funct-symbol[f, const, h] \Leftrightarrow$$

$$\forall_{\substack{is-nat[x,y] \\ y>0}} (f[x, y] = \begin{cases} const & \Leftarrow x < y \\ h[f[x - y, y]] & \Leftarrow \text{otherwise} \end{cases})].$$

- we generate all the possible combinations that can be constructed using the new *schRemFunctSym* and *schQuotFunctSym* knowledge schemes and the list of all the possible combinations of the symbols from the current language of the theory:

$$\begin{aligned}
\text{possibleSubsts} := & \\
& \{ \{ f \rightarrow \bullet[\text{quot}], \text{const} \rightarrow \bullet[0], h \rightarrow \bullet[\text{SuperPlus}] \}, \\
& \{ f \rightarrow \bullet[\text{lessquot1}], \text{const} \rightarrow \bullet[0], h \rightarrow \bullet[\text{Identity}] \}, \\
& \{ f \rightarrow \bullet[\text{lessquot2}], \text{const} \rightarrow \bullet[1], h \rightarrow \bullet[\text{SuperPlus}] \}, \\
& \{ f \rightarrow \bullet[\text{lessquot3}], \text{const} \rightarrow \bullet[1], h \rightarrow \bullet[\text{Identity}] \}
\end{aligned}$$

$$\text{newConcepts} := \text{Map}[\text{UseScheme}[\text{schQuotFunctSymbol}, \#]\&, \text{possibleSubsts}]$$

$$\begin{aligned}
\text{newpossibleSubsts} := & \\
& \{ \{ f \rightarrow \bullet[\text{rem}], g \rightarrow \bullet[\text{Identity}], h \rightarrow \bullet[\text{Identity}] \}, \\
& \{ f \rightarrow \bullet[\text{rem1}], g \rightarrow \bullet[\text{Identity}], h \rightarrow \bullet[\text{SuperPlus}] \}, \\
& \{ f \rightarrow \bullet[\text{rem2}], g \rightarrow \bullet[\text{Identity}], h \rightarrow \bullet[\text{Identity}] \}, \\
& \{ f \rightarrow \bullet[\text{rem3}], g \rightarrow \bullet[\text{Identity}], h \rightarrow \bullet[\text{SuperPlus}] \}
\end{aligned}$$

$$\text{newConcepts} := \text{Map}[\text{UseScheme}[\text{schRemFunctSymbol}, \#]\&, \text{newpossibleSubsts}]$$

-we obtain two lists of new function symbols (the *newConcepts* and the *othernewConcepts* lists), that have the following form:

newConcepts

$$\begin{aligned}
& (\text{quot-funct-symbol}[\text{quot}, 0, \text{SuperPlus}] \Leftrightarrow \\
& \quad \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} \bigwedge \left\{ \begin{array}{l} (0 \Leftarrow x < y) \\ (\text{quot}[x - y, y]^+ \Leftarrow \text{otherwise}) \end{array} \right.)
\end{aligned}$$

$$\begin{aligned}
& (\text{quot-funct-symbol}[\text{lessquot1}, 0, \text{Identity}] \Leftrightarrow \\
& \quad \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} \bigwedge \left\{ \begin{array}{l} (0 \Leftarrow x < y) \\ (\text{lessquot1}[x - y, y] \Leftarrow \text{otherwise}) \end{array} \right.)
\end{aligned}$$

$$\begin{aligned}
& (\text{quot-funct-symbol}[\text{lessquot2}, 1, \text{SuperPlus}] \Leftrightarrow \\
& \quad \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} \bigwedge \left\{ \begin{array}{l} (1 \Leftarrow x < y) \\ (\text{lessquot2}[x - y, y]^+ \Leftarrow \text{otherwise}) \end{array} \right.)
\end{aligned}$$

$$\begin{aligned}
& (\text{quot-funct-symbol}[\text{lessquot3}, 1, \text{Identity}] \Leftrightarrow \\
& \quad \forall_{\substack{\text{is-natural}[x] \\ \text{is-natural}[y]}} \bigwedge \left\{ \begin{array}{l} (1 \Leftarrow x < y) \\ (\text{lessquot3}[x - y, y] \Leftarrow \text{otherwise}) \end{array} \right.)
\end{aligned}$$

otherNewConcepts

$$(rem-funct-symbol[rem, Identity, Identity] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (x \Leftarrow x < y) \\ (rem[x - y, y] \Leftarrow otherwise) \end{array} \right.)$$

$$(rem-funct-symbol[rem1, Identity, SuperPlus] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (x \Leftarrow x < y) \\ (rem1[x - y, y]^+ \Leftarrow otherwise) \end{array} \right.)$$

$$(rem-funct-symbol[rem2, SuperPlus, Identity] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (x^+ \Leftarrow x < y) \\ (rem2[x - y, y] \Leftarrow otherwise) \end{array} \right.)$$

$$(rem-funct-symbol[rem3, SuperPlus, SuperPlus] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \wedge \left\{ \begin{array}{l} (x^+ \Leftarrow x < y) \\ (rem3[x - y, y]^+ \Leftarrow otherwise) \end{array} \right.)$$

-we obtain from these two lists of new function symbols the traditional *quotient* and *remainder* function symbols:

$$DefQuotient := newConcepts[1]$$

$$DefRemainder := otherNewConcepts[1]$$

-we introduce the new definitions in the theory using the 'Definition' key word from the THEOREMA system:

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

$$\text{any}[is-nat[x], is-nat[y]], \\ rem[x, y] = \left\{ \begin{array}{l} x \quad \Leftarrow x < y \\ rem[x - y, y] \quad \Leftarrow otherwise \end{array} \right\}.$$

Definition["nat 5.2: simple recursion: quotient",

$$\text{any}[is-nat[x], is-nat[y]], \\ quot[x, y] = \left\{ \begin{array}{l} 0 \quad \Leftarrow x < y \\ quot[x - y, y] + 1 \quad \Leftarrow otherwise \end{array} \right\}.$$

9.3 New Propositions Introduced in the Theory

We introduce new propositions into the theory concerning the new quotient and remainder function symbols.

9.3.1 Propositions-Equivalent Definitions

We cannot write any new knowledge schemes that can introduce equivalent definitions for the new invented function symbols. So, no new propositions-equivalent definitions concerning the *quotient* or the *remainder* function symbols can be introduced in the theory being developed.

9.3.2 Structural Propositions

We do not have any algebraic structures that can be verified by the quotient or the remainder function symbols. So, we cannot introduce any new structural propositions concerning the quotient or the remainder function symbols in the theory being developed.

9.3.3 Propositions resulting from the interaction between all the notions already introduced in the theory

There is only one proposition resulting from the interaction between the *quotient* function symbol and all the other notions from the language of the theory, which is the *sort quotient* proposition. We introduce directly the proposition into the theory, as an interaction between the *is-natural* predicate symbol and the *quotient* function symbol:

$$\begin{aligned} &\text{Proposition[\"nat 5.1 : sort quotient\"],} \\ &\quad \text{any[is-natural[x], is-natural[y]],} \\ &\quad \text{is-natural[quot[x, y]]} \end{aligned}$$

After introducing the sort quotient proposition into the theory, we prove it using the *NatProver* prover from the THEOREMA system which implements the stepwise induction principle, the rules of natural deduction and some extra helpful propositions that have to be introduced in the theory in order to obtain a successful proof:

```
Prove[
Proposition[\"nat.5.1: sort quotient\"],
using → {Proposition[\"nat.1.2:sort\"], Proposition[\"nat.2.1:sort\"],
  Axiom[\"generation\"], Definition[\"nat.5.1:simple recursion: quotient\"],
  Proposition[\"helpfull quotient\"], Proposition[\"second helpfull quotient\"],
  Proposition[\"third helpfull quotient\"], Proposition[\"forth helpfull quotient\"],
  Proposition[\"fifth helpfull quotient\"]},
by → NatProver]
```

The proof is successful.

Remark. The extra propositions that are used in the previous proof are extracted from the unsuccessful proofs of the *sort quotient* proposition that are initially generated. Each time the proof does not succeed because no knowledge about an intermediary goal from the proof was available, the intermediary goal is added to the knowledge base as an extra proposition that has to be proved and then used for proving the original proposition. The extra propositions used for proving the sort quotient proposition have the following form:

$$\begin{aligned} &\text{Proposition[\"helpfull quotient\"],} \\ &\quad \text{quot[0,0]=1} \\ &\text{Proposition[\"second helpfull quotient\"],} \\ &\quad \text{is-natural[1]} \\ &\text{Proposition[\"third helpfull quotient\"],} \\ &\quad \text{any[is-natural[y]],} \\ &\quad \text{is-natural[quot[0, y^+]]} \end{aligned}$$

Proposition["forth helpfull quotient"],
 $any[is-natural[x]],$
 $is-natural[quot[x^+, 0]]$

Proposition["fifth helpfull quotient"],
 $any[is-natural[x], is-natural[y]],$
 $is-natural[quot[x^+, y^+]]$

All these extra propositions are new and they did not exist in the theory being developed. Therefore, they have to be first proven before being used for other proofs. The attempts made in this direction for proving the extra propositions were unsuccessful. New attempts will be made in order to accomplish this goal.

The other propositions used for proving the *sort quotient* proposition are propositions already introduced in the theory being developed and they have the following form:

Proposition["nat.1.2: sort"],
 $any[is-natural[x], is-natural[y]],$
 $is-natural[x + y]$

Proposition["nat.2.1: sort"],
 $any[is-natural[x], is-natural[y]],$
 $is-natural[x * y]$

The *generation* axiom and the *quotient* definition are also notions already introduced in the theory and they have the following form:

Axiom["nat: generation"],
 $any[is-natural[x]],$
 $is-natural[0]$
 $is-natural[x^+]$

The definition for the *quotient* function symbol is the definition already invented and introduced in the theory in the previous subsection called *New Definitions Introduced In The Theory*.

We also have a similar property for the *remainder* function symbol called *sort remainder*, which can be proved using the same considerations presented as for the *sort quotient* proposition. Another property which is the result of the interaction between the *is-natural*, $<$ relation symbols and the *addition*, *multiplication*, *quotient* and the *remainder* function symbol is the *quotient-remainder* proposition. We introduce directly the proposition into the theory:

Proposition["nat.5.3: quotient-remainder"],
 $any[is-natural[x], is-natural[y]],$
 $x=y*quot[x,y]+rem[x,y]$
 $rem[x,y]<y$

Trying to prove this proposition using the provers that implement the step-wise induction principle or the prover that implements the complete induction principle does not lead us to a successful proof. All the extra propositions that are also added to the knowledge base does not lead us to a successful proof, neither. Further attention must be paid when trying to proof this proposition using the prover that implements the complete induction principle because the proof should be successful using such a mechanism.

10 Exploration Round 6: New Relation Symbol (divides)

10.1 Current Language Used for Exploration

The current language contains the language from the previous step of the exploration and the new introduced function symbols: the *quotient* and the *remainder* function symbols:

$$L[5] \leftarrow L[4] \cup \{quot, rem\} := \bullet language[\bullet constants[0, 1], \\ \bullet functions[\bullet [Super Plus], \bullet [Identity], \\ \bullet [Plus], \bullet [Times], \bullet [Power] \\ \bullet [Super Minus], \bullet [Minus], \\ \bullet [quot], \bullet [rem]], \\ \bullet predicates[\bullet [is-natural], \bullet [=], \bullet [\leq], \bullet [<]]]$$

10.2 New Definition Introduced in the Theory

Using new knowledge schemes we should be able to invent new relation symbols. In order to introduce new relation symbols in the theory, we follow the steps previously described:

- we write the knowledge scheme used for inventing new symbols:

$$schDividesRelation := \bullet [\forall_{r,q,s,const} (is-divides-relation-symbol[r, q, s, const] \Leftrightarrow \\ \forall_{is-nat[x,y]} (r[x, y] \Leftrightarrow \left\{ \begin{array}{ll} const & \Leftarrow y = 0 \\ q[x, y] & \Leftarrow x > y \\ s[r[x, y - x]] & \Leftarrow otherwise \end{array} \right.))].$$

-we generate the list of all the possible combinations that can be constructed using the *schDividesRelation* knowledge scheme and the notions already introduced in the theory:

$$possibleSubsts := \\ \{ \{r \rightarrow \bullet [lessdivides1], p \rightarrow \bullet [True], q \rightarrow \bullet [True], const \rightarrow \bullet [True]\}, \\ \{r \rightarrow \bullet [lessdivides2], p \rightarrow \bullet [True], q \rightarrow \bullet [True], const \rightarrow \bullet [False]\}, \\ \{r \rightarrow \bullet [lessdivides3], p \rightarrow \bullet [True], q \rightarrow \bullet [False], const \rightarrow \bullet [True]\}, \\ \{r \rightarrow \bullet [lessdivides4], p \rightarrow \bullet [True], q \rightarrow \bullet [False], const \rightarrow \bullet [False]\}, \\ \{r \rightarrow \bullet [()], p \rightarrow \bullet [False], q \rightarrow \bullet [True], const \rightarrow \bullet [True]\}, \\ \{r \rightarrow \bullet [lessdivides5], p \rightarrow \bullet [False], q \rightarrow \bullet [True], const \rightarrow \bullet [False]\}, \\ \{r \rightarrow \bullet [lessdivides6], p \rightarrow \bullet [False], q \rightarrow \bullet [False], const \rightarrow \bullet [True]\}, \\ \{r \rightarrow \bullet [lessdivides7], p \rightarrow \bullet [False], q \rightarrow \bullet [False], const \rightarrow \bullet [False]\} \}$$

$$newConcepts := Map[UseScheme[schDividesRelation, \#]\&, possibleSubsts]$$

As a remark the list *newConcepts* of all the possible new relation symbols has the following form:

newConcepts

$$(is-divides-relation-symbol[lessdivides1, True, True, True] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (True \Leftarrow y = 0) \\ (True[x, y] \Leftarrow x > y) \\ (True[lessdivides1[x, y - x]] \Leftarrow otherwise) \end{array} \right.)$$

$$(is-divides-relation-symbol[lessdivides2, True, True, False] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (False \Leftarrow y = 0) \\ (True[x, y] \Leftarrow x > y) \\ (True[lessdivides2[x, y - x]] \Leftarrow otherwise) \end{array} \right.)$$

$$(is-divides-relation-symbol[lessdivides3, True, False, True] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (True \Leftarrow y = 0) \\ (True[x, y] \Leftarrow x > y) \\ (False[lessdivides3[x, y - x]] \Leftarrow otherwise) \end{array} \right.)$$

$$(is-divides-relation-symbol[lessdivides4, True, False, False] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (False \Leftarrow y = 0) \\ (True[x, y] \Leftarrow x > y) \\ (False[lessdivides4[x, y - x]] \Leftarrow otherwise) \end{array} \right.)$$

$$(is-divides-relation-symbol[, False, True, True] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (True \Leftarrow y = 0) \\ (False[x, y] \Leftarrow x > y) \\ (True[x|y - x] \Leftarrow otherwise) \end{array} \right.)$$

$$(is-divides-relation-symbol[lessdivides5, False, True, False] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (False \Leftarrow y = 0) \\ (False[x, y] \Leftarrow x > y) \\ (True[lessdivides5[x, y - x]]) \end{array} \right.)$$

$$(is-divides-relation-symbol[lessdivides6, False, False, True] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (True \Leftarrow y = 0) \\ (False[x, y] \Leftarrow x > y) \\ (False[lessdivides6[x, y - x]] \Leftarrow otherwise) \end{array} \right.)$$

$$(is-divides-relation-symbol[lessdivides7, False, False, False] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \left\{ \begin{array}{l} (False \Leftarrow y = 0) \\ (False[x, y] \Leftarrow x > y) \\ (False[lessdivides7[x, y - x]] \Leftarrow otherwise) \end{array} \right.)$$

From this list we extract the symbol we need for inventing the divides relation symbol:

$$Def\ Divides := newConcepts[5]$$

At this point we can introduce the definition for the divides relation symbol, using the 'Definition' key word from the THEOREMA system:

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

$$x|y = \left[\begin{array}{l} \text{False} \quad \Leftarrow y = 0 \\ \text{True} \quad \Leftarrow x > y \\ x|(y-x) \quad \Leftarrow \textit{otherwise} \end{array} \right].$$

10.3 New Propositions Introduced in the Theory

10.3.1 Propositions-Equivalent Definitions

No new propositions-equivalent definitions concerning the divides relation symbol can be introduced in the theory because we cannot write any new definition knowledge schemes.

10.3.2 Structural Propositions

The divides relation symbol verifies the partial ordering structure, which in fact means that the following properties are verified by the divides relation symbol: the reflexivity, the transitivity and the antisymmetry properties. We will exemplify the invention and the introduction into the theory of the reflexivity proposition of the divides relation symbol.

The *schPartialOrder* algebraic knowledge scheme used for inventing the structural properties of the divides relation symbol has the following form:

$$\begin{aligned} \text{schPartialOrder} := & \bullet [\forall_{p,r} (\textit{is-partial-ordering}[p, r] \Leftrightarrow \\ & \forall_{p[x,y,z]} \bigwedge \left\{ \begin{array}{l} (\textit{is-preorder}[p,r]) \\ ((r[x,y]) \wedge (r[y,x])) \Rightarrow (x = y) \end{array} \right\} \end{aligned}$$

The knowledge scheme for the preorder algebraic structure, which is used for defining the partial ordering algebraic structure, has the following form:

$$\begin{aligned} \text{schPreorder} := & \bullet [\forall_{p,r} (\textit{is-preorder}[p, r] \Leftrightarrow \\ & \forall_{p[x,y,z]} \bigwedge \left\{ \begin{array}{l} (r[x,x]) \\ ((r[x,y]) \wedge (r[y,z])) \Rightarrow (r[x,z]) \end{array} \right\} \end{aligned}$$

Using these algebraic knowledge schemes we can introduce the reflexivity of the divides relation symbol proposition into the theory:

$$\begin{aligned} \text{Proposition}["\textit{nat.6.1:reflexivity}()",] \\ \text{any}[\textit{is-natural}[x], \\ x|x] \end{aligned}$$

After we introduce the proposition into the theory we try to prove it using one of the implemented provers from the THEOREMA system:

```

Prove[
Proposition["nat.6.1: reflexivity()"],
using → {Definition["nat.6.1:simple recursion:new divides"]},
by → NatProver]

```

Unfortunately, the proof does not succeed. Further attention must be paid to this unsuccessful proof as to all the other proofs that fail when trying to prove them.

10.3.3 Propositions resulting from the interaction between all the notions already introduced in the theory

The following propositions can be introduced in the theory as different interactions between all the notions that exist in the language of the theory:

- the *right zero* and *left zero* properties of the divides relation symbol that are introduced as interactions between the 0 constant, the *is-natural* predicate symbol and the *divides* relation symbol.
- the *addition* property of the divides relation symbol that is introduced as an interaction between the *is-natural* predicate symbol, the *addition* ($+$) function symbol and the *divides* relation symbol.
- the *multiplication* property of the divides relation symbol that is introduced as an interaction between the *is-natural* predicate symbol, the *multiplication* ($*$) function symbol and the *divides* relation symbol.
- the *remainder* property of the divides relation symbol that is introduced as an interaction between the *is-natural* predicate symbol, the *rem* function symbol and the *divides* relation symbol.

We will exemplify the invention of the *remainder* proposition of the divides relation symbol. We introduce the proposition directly into the theory:

Proposition["nat.6.1: remainder()"],
 any[is-natural[x], is-natural[y]],
 $x|y \Leftrightarrow \text{rem}[y,x]=0$

We prove the proposition using the *NatProver* prover, the definition of the divides relation symbol and the previous invented properties concerning the divides relation symbol: the reflexivity, the transitivity, the antisymmetry, the right zero, the left zero, the addition and the multiplication properties of the divides relation symbol:

```
Prove[
  Proposition["nat.6.1: remainder()"],
  using → {Definition["nat.6.1:simple recursion:new divides"],
    Proposition["nat.6.1:reflexivity()"],
    Proposition["nat.6.1:transitivity()"],
    Proposition["nat.6.1:antisymmetry()"],
    Proposition["nat.6.1:rightzero()"],
    Proposition["nat.6.1:left zero()"],
    Proposition["nat.6.1:addition()"],
    Proposition["nat.6.1:multiplication()"]},
  by → NatProver]
```

The proof is successful.

11 Exploration Round 7: New Function Symbol (greatest common divisor)

11.1 Current Language used for exploration

The current language used for exploration contains the initial language from the previous step of the exploration and the new introduced *divides* (\mid) relation symbol:

$$\begin{aligned}
 L[6] \leftarrow L[5] \cup \{ \mid \} := \\
 & \bullet \text{language} [\bullet \text{constants} [0, 1], \\
 & \quad \bullet \text{functions} [\bullet \text{SuperPlus}, \bullet \text{Identity}, \\
 & \quad \quad \bullet \text{Plus}, \bullet \text{Times}, \bullet \text{quot}, \bullet \text{rem}], \\
 & \quad \bullet \text{predicates} [\bullet \text{is-natural}, \bullet \text{=}, \bullet \text{≤}, \bullet \text{<}, \bullet \text{[]}]]
 \end{aligned}$$

11.2 New Definition Introduced in the Theory

For introducing a new definition in the theory, we proceed in the same way as in the previous exploration rounds, when we have introduced other new mathematical notions. We follow the following exploration steps:

-we begin with writing the definition knowledge scheme, which is used for inventing new function symbols:

$$\begin{aligned}
 \text{schGreatFunctSymbol} := & \bullet [\forall_{f,g,h} (\text{great-funct-symbol}[f, g, h] \Leftrightarrow \\
 & \quad \forall_{\text{is-natural}[x,y]} \wedge \left\{ \begin{array}{l} (f[x, 0] = g[x]) \\ (f[x, y] = f[y, h[x, y]]) \end{array} \right\})]
 \end{aligned}$$

-we generate the list of all the possible combinations that can be constructed using the *schGreatFunctSymbol* knowledge scheme and the notions already introduced in the theory:

$$\begin{aligned}
 \text{possibleSubsts} := \\
 & \{ \{ f \rightarrow \bullet \text{gcd}, g \rightarrow \bullet \text{Identity}, h \rightarrow \bullet \text{rem} \}, \\
 & \{ f \rightarrow \bullet \text{gcd1}, g \rightarrow \bullet \text{Identity}, h \rightarrow \bullet \text{quot} \}, \\
 & \{ f \rightarrow \bullet \text{gcd2}, g \rightarrow \bullet \text{Identity}, h \rightarrow \bullet \text{+} \}, \\
 & \{ f \rightarrow \bullet \text{gcd3}, g \rightarrow \bullet \text{Identity}, h \rightarrow \bullet \text{*} \}, \\
 & \{ f \rightarrow \bullet \text{gcd4}, g \rightarrow \bullet \text{SuperPlus}, h \rightarrow \bullet \text{rem} \}, \\
 & \{ f \rightarrow \bullet \text{gcd5}, g \rightarrow \bullet \text{SuperPlus}, h \rightarrow \bullet \text{quot} \}, \\
 & \{ f \rightarrow \bullet \text{gcd6}, g \rightarrow \bullet \text{SuperPlus}, h \rightarrow \bullet \text{+} \}, \\
 & \{ f \rightarrow \bullet \text{gcd7}, g \rightarrow \bullet \text{SuperPlus}, h \rightarrow \bullet \text{*} \}
 \end{aligned}$$

$$\text{newConcepts} := \text{Map}[\text{UseScheme}[\text{schGreatFunctSymbol}, \#] \& \text{possibleSubsts}]$$

As a remark the list *newConcepts* of all the possible new function symbols that can be invented, has the following form:

newConcepts

$$(great-funct-symbol[gcd, Identity, rem] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd[x, 0] = x \\ gcd[x, y] = gcd[y, rem[x, y]] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd1, Identity, quot] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd1[x, 0] = x \\ gcd1[x, y] = gcd1[y, quot[x, y]] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd2, Identity, +] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd2[x, 0] = x \\ gcd2[x, y] = gcd2[y, x + y] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd3, Identity, *] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd3[x, 0] = x \\ gcd3[x, y] = gcd[x * y] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd4, SuperPlus, rem] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd4[x, 0] = x^+ \\ gcd4[x, y] = gcd4[y, rem[x, y]] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd5, SuperPlus, quot] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd5[x, 0] = x^+ \\ gcd5[x, y] = gcd5[y, quot[x, y]] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd6, SuperPlus, +] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd6[x, 0] = x^+ \\ gcd6[x, y] = gcd[y, x + y] \end{array} \right. \Bigg)$$

$$(great-funct-symbol[gcd7, SuperPlus, *] \Leftrightarrow \forall_{\substack{is-natural[x] \\ is-natural[y]}} \bigwedge \left\{ \begin{array}{l} gcd7[x, 0] = x^+ \\ gcd7[x, y] = gcd7[y, x * y] \end{array} \right. \Bigg)$$

From this list we extract the symbol we need for inventing the greatest common divisor function symbol:

DefGreatestCommonDivisor := newConcepts[1]

At this point we can introduce the definition for the greatest common divisor function symbol, using the *Definition* key word from the THEOREMA system:

Definition["nat 7.1: simple recursion: greatest common divisor ",
 $\text{any}[is\text{-natural}[x], is\text{-natural}[y]],$
 $\text{gcd}[x,0]=0$
 $\text{gcd}[x,y]=\text{gcd}[y,\text{rem}[x,y]]]$

11.3 New Propositions Introduced in the Theory

As in the previous exploration rounds we can introduce new propositions into the theory concerning the greatest common divisor function symbol introduced in the theory.

11.3.1 Propositions-Equivalent Definitions

No new propositions-equivalent definitions concerning the greatest common divisor function symbol can be introduced because we cannot write any new definition knowledge schemes.

11.3.2 Structural Propositions

We do not have any algebraic structures that can be verified by the greatest common divisor function symbol. So, we cannot introduce any new structural propositions concerning the greatest common divisor function symbol in the theory being developed.

11.3.3 Propositions resulting from the interaction between all the notions existing in the theory

The following propositions can be introduced in the theory as different interactions between all the notions that exist in the language of the theory:

- the *common divisor* and the *greatest common divisor* properties of the greatest common divisor function symbol that are introduced as interactions between the *is-natural* predicate symbol, the *divides* relation symbol and the *greatest common divisor* function symbol.

We will exemplify the invention of the *common divisor* proposition of the greatest common divisor function symbol. We introduce the proposition directly into the theory:

Proposition["nat.7.1: common divisor"],
 $\text{any}[is\text{-natural}[x], is\text{-natural}[y]],$
 $\text{gcd}[x,y] \mid x \wedge \text{gcd}[x,y] \mid y]$

We prove the proposition using the *NatProver* prover, the definition of the divides relation symbol and the previous invented properties concerning the divides relation symbol: the transitivity, the antisymmetry, the right zero, the left zero, the addition, the multiplication and the remainder properties of the divides relation symbol:

```

Prove[
  Proposition["nat.7.1: common divisor"],
  using → {Definition["nat.6.1:simple recursion:new divides"],
    Proposition["nat.6.1:transitivity()"], Proposition["nat.6.1:antisymmetry()"],
    Proposition["nat.6.1:rightzero()"], Proposition["nat.6.1:left zero()"],
    Proposition["nat.6.1:addition()"], Proposition["nat.6.1:multiplication()"],
    Proposition["nat.6.1: remainder()"]},
  by → NatProver]

```

The proof is successful.

12 Conclusions

In this technical report, we thoroughly describe how to introduce mathematical notions in a theory, and in this case, particularly in the natural numbers theory.

We have introduced in the theory that we develop important notions such as: the addition, the multiplication, the exponentiation, the predecessor, the subtraction, the quotient, the remainder, the greatest common divisor function symbols, the less than, the strict less than and the divides relation symbols. All these notions are explored using the model proposed by Bruno Buchberger. Each exploration round consists of several examples that reduces to: the invention of new notions using knowledge schemes from the library of schemes, the invention of new propositions-equivalent definitions using recursive knowledge schemes, the invention of structural propositions using algebraic knowledge schemes, the invention of propositions resulting from the interaction between all the notions already introduced in the theory.

Still, further exploration will be made concerning the: prime number theory, the prime decomposition theorem, etc.

We propose to prove the complete induction principle. The complete induction principle is important for our defined theory, because any sentence we can prove by complete induction principle we can also prove by stepwise induction principle, but the stepwise induction principle proof may be more complex. Therefore after proving the complete induction principle, we can use it to prove all the propositions from the theory: both the new propositions introduced in further exploration and the old propositions already introduced in the theory, but not successfully proved. We have started the proof of the complete induction principle as we have already seen in one of the previous chapter from this paper, but improvement must still be made in order to have a complete proof of this new inference mechanism.

We have also developed new induction provers for natural numbers: the *NatProver*, the *NatProverPC* and the *SimplerNatProverPC*. These induction provers use as inference rule the stepwise induction principle, the natural deduction rules for induction principle and some simplifying (rewriting) rules for mathematical formulae. We have also developed an induction prover that uses the complete induction principle as an inference rule: the *NewNatProver*, but improvements can still be made.

When the exploration is done, the introduced natural numbers theory will be important when studying other important case studies while exploring mathematical theories like: the theory of tuples, the theories related to Groebner

bases.

References

- [Buchberger B.,2004] B. Buchberger. Algorithm-Supported Mathematical Theory Exploration: A Personal View and Strategy. In: J. Campbell (ed.), *Proceedings of AISC 2004 (7th Conference on Artificial Intelligence and Symbolic Computation, Sep. 22-24, 2004, Research Institute for Symbolic Computation, Hagenberg, Austria)*, Lecture Notes in Artificial Intelligence, Springer.
- [Zohar Manna, Richard Waldinger, 1985] Zohar Manna, Richard Waldinger, *The Logical Basis for Computer Programming, Volume I, Deductive Reasoning*, Addison-Wesley Publishing Co, 1985.
- [Shoenfield, 1967] J.R. Schoenfield, *Mathematical Logic*, Addison-Wesley Publishing Co, SUA, 1967