Efficiently Simulating Higher-Order Arithmetic as a First-Order Theory Modulo

Guillaume Burel

Université Henri Poincaré & LORIA
Team Protheo – Bâtiment B
Campus Scientifique BP 239
54506 Vandœuvre-lès-Nancy Cedex, France

Abstract

In 1973, Parikh proved a speed-up theorem conjectured by Gödel 37 years before: there exist arithmetical formulæ that are provable in first-order arithmetic, but whose shorter proof in second-order arithmetic is arbitrarily smaller than any proof in first order. This can be generalized for all orders. On the other hand, resolution for higher-order logic can be simulated step by step in a first-order narrowing and resolution method based on deduction modulo, whose paradigm is to clearly identify deduction and computation to make proofs clearer and shorter. The question therefore arises whether deduction modulo permits to get rid of the speed-up in arithmetic.

In the continuation of a work of Dowek and Werner, we show here how to express higher-order arithmetic as a rewrite system modulo which deduction takes place. Nevertheless, we are taking care of the length of proofs: we prove that proofs in the usual presentation of higher-order arithmetic can be linearly translated into proofs in natural deduction modulo this system. In particular, higher order itself is translated using a subset of this rewrite system which is very simple (it is finite, terminating in linear time, confluent, left-linear), and which permits to transform the axiom schemata of higher-order arithmetic into a finite number of first-order axioms. We then use this to prove that the speed-up in arithmetic can be decomposed into two: a first speed-up between \( i \)-th order arithmetic and a finite first-order axiomatization of \( i + 1 \)-th order arithmetic, and a second one between this axiomatization and the usual formulation of \( i + 1 \)-th order arithmetic. The linear simulation of the latter using deduction modulo shows that building proofs modulo permits to avoid this second speed-up. These results allows us to prove that the speed-up conjectured by Gödel can be expressed as simple computation, therefore justifying the use of deduction modulo as an efficient first-order setting simulating higher order.

Key words: Proof theory, rewriting, higher-order logic, arithmetic, proof-length speed-ups

1. Introduction

Even if two logical systems are shown to be expressively equivalent, i.e. they can prove exactly the same formulæ, they can lead to very different proofs, in particular in terms of length. For instance, it is shown that
Frege systems have an exponential speed-up over resolution for propositional logic (Buss, 1987). However in mechanized theorem proving, the length of proofs has an importance: First, computers have limited capacities, and this can lead to a difference between the practical expressiveness of theoretically equivalent systems. Even if computing power is always increasing, so that one is no longer afraid to use SAT-solvers within verification tools (mainly because worst cases do not often occur in practice), it is not conceivable to build an automated theorem prover that produces proofs of non-elementary length. Second, the length of a proof is one (among others) criterion for defining the quality of a proof. Indeed, a smaller proof is often more readable and, in the case for instance of software certification and proof engineering, more communicable and in many cases also more maintainable. This notion of “good proofs” is translated by Dershowitz and Kirchner (2006); Bonacina and Dershowitz (2007) into a proof ordering, which of course may correspond to the comparison of proof lengths.

Obtaining a speed-up can also have a theoretical interest, because, as remarked by Parikh in the introductory paragraph of Gödel (1986), “the celebrated P-\text{NP}? question can itself be thought of as a speed-up question.” (See Cook and Reckhow, 1979) All this explains the research for new formalisms whose deductive systems provide smaller proofs, such as for instance the calculus of structures of Brünnler (2003) w.r.t. the sequent calculus of Gentzen (1934) (see Guglielmi, 2004).

In this paper, the length of a proof corresponds to its number of steps (sometimes called lines), whatever the actual size of the formulæ appearing in them is. Considering the minimal length of proofs, the definition of a speed-up is the following: given some function \( h \) over natural numbers, a system has a speed-up for \( h \) over another one, if there exists an infinite set of formulæ provable in both of them, such that, if the length of the proofs in the first system is \( l \) and the length in the second system is \( k \), then \( k > h(l) \).

In 1936, Gödel conjectured that there exists such a speed-up for all recursive functions between \( i \)-th-order and \( i + 1 \)-th-order arithmetic, no matter the formal system actually used. In other words, he stated that for all recursive functions \( h \), it is possible to find an infinite set of formulæ such that, for each of them, denoted by \( P \), if \( k \) is the minimal number of steps in the proofs of \( P \) in the \( i \)-th-order arithmetic (\( k \) is assumed to exist, so that \( P \) is provable in it), and \( l \) is the minimal number of steps in the proofs of \( P \) in the \( i + 1 \)-th-order arithmetic, then \( k > h(l) \).

This result was proved for first-order arithmetic by Parikh (1973), who actually proved a stronger theorem: this proof-length speed-up exists in fact also for non-recursive functions. This was generalized to all orders by Krajčíček (1989), and was proved for the true language of arithmetic by Buss (1994) (the former results used an axiomatization of arithmetic using ternary predicates to represent addition and multiplication). The theorem proved by Buss is stated as follow:

**Theorem 1 (Buss (1994, Theorem 3))** Let \( i \geq 0 \). Then there is an infinite family \( \mathcal{F} \) of \( \prod^0_1 \)-formulae such that

1. for all \( P \in \mathcal{F} \), \( Z_i \vdash P \)
2. there is a fixed \( k \in \mathbb{N} \) such that for all \( P \in \mathcal{F} \), \( Z_{i+1}^1 \frac{1\text{ steps } k}{k} P \)
3. there is no fixed \( k \in \mathbb{N} \) such that for all \( P \in \mathcal{F} \), \( Z_i^1 \frac{1\text{ steps } k}{k} P \).

\( Z_i \) corresponds to the \( i + 1 \)-th-order arithmetic (so \( Z_0 \) is in fact first-order arithmetic), and \( Z_i^1 \frac{1\text{ steps } k}{k} P \) means that \( P \) can be proved in at most \( k \) steps within a schematic system — i.e. a Hilbert-type (or Frege) system with a finite number of axiom schemata and inference rules — for \( i + 1 \)-th-order arithmetic. (In fact, Buss proved this theorem also for weakly schematic systems, i.e. schematic systems in which every tautology can be used as an axiom, as well as generalizations of axioms, but we will not use this fact here.)

Because this theorem is concerned in arithmetic, an intuitive notion of computation take place in the proofs. Indeed, as remarked by Poincaré, establishing that \( 2 + 2 = 4 \) using the definition of the addition is just a verification, and not a demonstration, so that in a proof occur in fact not only pure deduction but also computation. Therefore, the question arises whether this speed-up comes from the deductive or the computational part of the proofs, or both of them. Of course, the difference between computation and deduction cannot be clearly determined. Because of the Curry-Howard correspondence, the whole content of the proofs could be considered as computation. (Concerning proofs as programs and arithmetic, see Schwichtenberg (2007).) Here, this difference must be thought of as the distinction between what is straightforward (at least decidable), and what must be reasoned out.
Deduction modulo (Dowek, Hardin, and Kirchner, 2003) is a presentation of a given logic—and the formalisms associated with it—identifying what corresponds to computation. The computational part of a proof is put in a congruence between formulæ modulo which the application of the deduction rules takes place. This leads for instance to the sequent calculus modulo and to the natural deduction modulo. The congruence is better represented as a set of rewrite rules that can rewrite terms but also atomic propositions: indeed, one wants for instance to consider the definition of the addition or multiplication using rewrite rules over terms as part of the computation, but also the following rewrite rule:

\[ x \times y = 0 \rightarrow x = 0 \lor y = 0 \]

which rewrites an atomic proposition to a formula, so that the following simple natural-deduction-modulo proof of \( t \times t = 0 \) can be deduced from a proof \( \pi \) of \( t = 0 \):

\[
\frac{\pi}{\forall i \quad t = 0 \quad t \times t = 0 \quad \rightarrow \ t = 0 \lor t = 0}
\]

Deduction modulo is logically equivalent to the considered logic (Dowek et al., 2003, Proposition 1.8), but proofs are often considered as simpler, because the computation is hidden, letting the deduction clearly appear. Proofs are also claimed to be shorter for the same reason. Nevertheless, this fact was never quantified. This paper answers this issue. Of course, if there are no restriction on the rewrite rules that are used (for instance if it is allowed to use a rewrite system semi-deciding the validity of formulæ), it is not surprising that the length of the proofs can be unboundedly reduced. Notwithstanding, the rewriting system that we will consider in this paper for encoding higher order is very simple: is finite, terminating in quadratic time (in linear time in practical cases), confluent (i.e. deterministic) and linear (variables in the left-hand side only appear once).

Besides, it is possible, in deduction modulo, to build proofs of Higher-Order Logic using a first-order system (Dowek, Hardin, and Kirchner, 2001). Using this, a step of higher-order resolution is completely simulated by a step of ENAR, the resolution and narrowing method based on deduction modulo. It looks like this is also the case for the associated sequent calculi, however this was not clearly stated. Therefore, it seems reasonable to think that deduction modulo is able to give the same proof-length speed-ups as the ones occurring between \( i + 1^{st} \)- and \( i^{th} \)-order arithmetic. This paper therefore investigates how to relate proof-length speed-ups in arithmetic with the computational content of the proofs.

To prove that the origin of the speed-up theorem of Buss can be expressed as simple computation, we proceed in two steps: First, we show how to encode higher order using a rewrite system and a finite set of axioms, by generalizing the work of Kirchner (2006). This rewrite system, despite its simplicity, permits to obtain the proof-length speed-up. Second, extending the work of Dowek and Werner (2005), we will express higher-order arithmetic completely as a rewrite system modulo whom the inference rules of natural deduction will take place. Of course, this formulation permits the same speed-up. However, as Theorem 1 is proved only for schematic systems, we will begin by showing that proof lengths in schematic systems and natural deduction do not differ too much.

In the next section, we will recall the definition of a schematic system, and we will present such a system for \( i^{th} \)-order arithmetic. The section 3 will define formally what deduction modulo, and in particular natural deduction modulo consists in. In Section 4 we will give bounded translations between the schematic system for \( i^{th} \)-order arithmetic and natural deduction. Then, the main section 5 will present how to efficiently encode higher order, and higher-order arithmetic. Finally, in Section 6 we will apply these results to determine the origin of the speed-ups in arithmetic. We will conclude about the interest of working within a first-order system modulo to simulate higher order.
2. A schematic system for \( i \)-th-order arithmetic

2.1. Schematic systems

We recall here, using Buss’ terminology (1994), what a schematic system consists in. It is essentially an Hilbert-type (or Frege) proof system, i.e. valid formulæ are derived from a finite number of axiom schemata using a finite number of inference rules. Theorem 1 is true on condition that proofs are performed using a schematic system.

First, we recall how to build many-sorted first-order formulæ (see Gallier, 1986, Chapter 10), mainly to introduce the notations we will use. A (first-order) many-sorted signature consists of a set of function symbols and a set of predicates, all of them with their arity (and co-arity for function symbols). We denote by \( T(\Sigma, V) \) the set of terms built from a signature \( \Sigma \) and a set of variables \( V \). An atomic proposition is given by a predicate symbol \( A \) of arity \([i_1, \ldots, i_n]\) and by \( n \) terms \( t_1, \ldots, t_n \in T(\Sigma, V) \) with matching sorts. It is denoted \( A(t_1, \ldots, t_n) \). Formulæ can be built using the following grammar:

\[
P = \bot \mid A \mid P \land P \mid P \lor P \mid P \Rightarrow P \mid \forall x. P \mid \exists x. P
\]

where \( A \) ranges over atomic propositions and \( x \) over variables. \( P \Leftrightarrow Q \) will be used as a syntactic sugar for \((P \Rightarrow Q) \land (Q \Rightarrow P)\), as well as \( \neg P \) for \( P \Rightarrow \bot \). Positions in a term or a formula, free variables and substitutions are defined as usual (see Baader and Nipkow, 1998). The replacement of a variable \( x \) by a term \( t \) in a formula \( P \) is denoted by \( \{t/x\}P \), the subterm or subformula of \( t \) at the position \( p \) by \( t|_p \), and its replacement in \( t \) by a term or formula \( s \) by \( t[s]_p \).

Then, given a many-sorted signature of first-order logic, we can consider infinite sets of metavariables \( \alpha^i \) for each sort \( i \) (which will be substituted by variables), of term variables \( \tau^i \) for each sort \( i \) (which will be substituted by terms) and proposition variables \( A(x_1, \ldots, x_n) \) for each arity \([i_1, \ldots, i_n]\) (which will be substituted by formulæ).

Metaterms are built like terms, except that they can contain metavariables and term variables. Metaformula are built like formulæ, except that they can contain proposition variables (which play the same role as predicates) and metaterms.

A schematic system is a finite set of inference rules, where an inference rule is a triple of a finite set of metaformulæ (the premises), a metaformula (the conclusion), and a set of side conditions of the forms \( \alpha^j \) is not free in \( \Phi \) or \( s \) is freely substitutable for \( \alpha^j \) in \( \Phi \) where \( \Phi \) is a metaformula and \( s \) a metaterm of sort \( j \). It is denoted by

\[
\frac{\Phi_1 \ldots \Phi_n}{\Psi} (R)
\]

An inference with an empty set of premises will be called an axiom schema. An axiom schema without metaformula is an axiom.

2.2. \( i \)-th-order arithmetic

\( i \)-th-order arithmetic \((Z_{i-1})\) is a many-sorted theory with the sorts \( 0, \ldots, i - 1 \) and the signature

\[
\begin{align*}
0 : & 0 & + : [0; 0] & \rightarrow 0 & = : [0; 0] \\
s : & [0] & \rightarrow 0 & \times : [0; 0] & \rightarrow 0 & \in^j : [j; j + 1]
\end{align*}
\]

The schematic system we use here consists of the following inference rules:

14 \( + \) 2 \( \times \) \( i \) \textbf{axiom schemata of classical logic}. We take the one used by Gentzen (1934, Chapter 5) to prove the equivalence of his formalisms with an Hilbert-type proof system:

\footnote{\( \dashv \) is used for definitions.}
\[ A \Rightarrow A \] (1)
\[ A \Rightarrow B \Rightarrow A \] (2)
\[ (A \Rightarrow A \Rightarrow B) \Rightarrow A \Rightarrow B \] (3)
\[ (A \Rightarrow B \Rightarrow C) \Rightarrow B \Rightarrow A \Rightarrow C \] (4)
\[ (A \Rightarrow B) \Rightarrow (B \Rightarrow C) \Rightarrow A \Rightarrow C \] (5)
\[ (A \land B) \Rightarrow A \] (6)
\[ (A \land B) \Rightarrow B \] (7)
\[ (A \Rightarrow B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \land A) \Rightarrow B \Rightarrow C \] (8)
\[ A \Rightarrow (A \lor B) \] (9)
\[ B \Rightarrow (A \lor B) \] (10)
\[ (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow (A \lor B) \Rightarrow C \] (11)
\[ (A \Rightarrow B) \Rightarrow (A \Rightarrow B \Rightarrow \bot) \Rightarrow A \Rightarrow \bot \] (12)
\[ (A \Rightarrow B) \Rightarrow (A \Rightarrow B) \Rightarrow \bot \Rightarrow A \Rightarrow \bot \] (13)
\[ \forall \alpha. (A(\alpha)) \Rightarrow A(\tau) \] (14)
\[ (\tau \text{ is freely substitutable for } \alpha \text{ in } A(\alpha)) \]
\[ A(\tau) \Rightarrow \exists \alpha. A(\alpha) \] (15)
\[ (\tau \text{ is freely substitutable for } \alpha \text{ in } A(\alpha)) \]
\[ A \lor (A \Rightarrow \bot) \] (16)

1 + 2 × i inference rules of classical logic. Again, we take the one used by Gentzen (1934):

\[ \frac{A}{A \Rightarrow B} \] (17)

\[ \frac{A \Rightarrow B(\beta)}{A \Rightarrow \forall \alpha. B(\alpha)} \] (18)

\[ \frac{B(\beta)}{(\exists \alpha. B(\alpha)) \Rightarrow A} \] (19)

7 identity axiom schemata. They define the particular relation =:

\[ \forall \alpha . \alpha = \alpha \] (20)
\[ \forall \alpha . \alpha = \beta \Rightarrow s(\alpha) = s(\beta) \] (21)
\[ \forall \alpha . \alpha = \beta \Rightarrow a + \gamma = \beta + \gamma \] (22)
\[ \forall \alpha . \alpha = \beta \Rightarrow \gamma a = \gamma \beta \] (23)
\[ \forall \alpha . \alpha = \beta \Rightarrow \alpha \times \gamma = \beta \times \gamma \] (24)
\[ \forall \alpha . \alpha = \beta \Rightarrow \gamma \alpha = \gamma \beta \] (25)
\[ \forall \alpha . \alpha = \beta \Rightarrow A(\alpha) \Rightarrow A(\beta) \] (26)

7 Robinson’s axioms. They are the axioms defining the function symbols of arithmetic (Mostowski, Robinson, and Tarski, 1953):
3.1. Rewriting formulæ

Deduction modulo

In this schematic system, i.e. Nipkow (1998), and extend them to proposition rewriting (Dowek et al., 2003). We use standard definitions, as given by Baader and Werner (2003). In deduction modulo, formulæ are considered modulo some congruence defined by some algebraic equations. A term rewrite system is the pair of a term rewrite rule and some position. A formula can be rewritten to a term by a term rewrite rule. This relation is extended by congruence to all formulæ.

A proposition rewrite rule is the pair of an atomic proposition A and a formula P, such that all free variables of P appear in A. It is denoted A → P. A proposition rewrite system is a set of proposition rewrite rules. A formula Q can be rewritten to a formula R by a proposition rewrite rule A → P if there exists some substitution σ and some position p in Q such that σA = Q[p] and R = Q[σP[p]]. Semantically, this proposition rewrite relation must be seen as a logical equivalence between formulæ.

A rewrite system is the union of a term rewrite system and a proposition rewrite system. The fact that P can be rewritten to Q either by a term or by a proposition rewrite rule of a rewrite system R will be denoted by A → P. The transitive (resp. reflexive transitive) closure of this relation will be denoted by A → R (resp. A → R).

3.2. Natural deduction modulo

Using some equivalence defined by a term and proposition rewrite system R, we can define natural deduction modulo as do Dowek and Werner (2003). Its inference rules are represented in Figure 1. They are the same as the one introduced by Gentzen (1934), except that we work modulo the rewrite relation. Leaves of a proof that are not introduced by some inference rules (contrary to A in ⇒i for instance) are the assumptions of the proof. Note that if we do not work modulo, ⇒-e is exactly the same as (17).
\begin{center}
\begin{tabular}{c}
\begin{tabular}{ccc}
<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Rightarrow\cdot \frac{[A]}{B} )</td>
<td>if ( C \vdash^R A \Rightarrow B )</td>
</tr>
<tr>
<td>( &amp;\cdot \frac{A}{B} )</td>
<td>if ( C \vdash^R A \land B )</td>
</tr>
<tr>
<td>( \lor\cdot \frac{A}{B} )</td>
<td>if ( C \vdash^R A \lor B ) or ( C \vdash^R B \lor A )</td>
</tr>
</tbody>
</table>
| \( \exists\cdot \frac{\{y/x\}A}{B} \) | if \( C \vdash^R \forall x. A \) and 
\( y \) is not free in \( A \) and not in the assumptions of the proof above |
| \( \exists\cdot \frac{B}{A} \) | if \( C \vdash^R \exists x. C \) and 
\( B \vdash^R \{t/x\}C \) |
| \( \Rightarrow\cdot \frac{A \lor \{y/x\}A}{B} \) | if \( C \vdash^R A \lor \{y/x\}A \) and 
\( y \) is not free in \( A \) and not in the assumption of the proof above except \( \{y/x\}A \) |
| \( \perp\cdot \frac{A \lor \perp}{B} \) | if \( C \vdash^R \perp \) |
\end{tabular}

\end{tabular}
\end{center}

Fig. 1. Inference Rules of Natural Deduction Modulo.

The length of a proof is the number of inferences used in it. We will denote by \( T_N^R P \) the fact that there exists a proof of \( P \) of length at most \( k \) using a finite subset of \( T \) (\( T \) can be infinite) as assumptions. In the case where \( R = \emptyset \), we are back to pure natural deduction, and we will use \( T_N P \). Abusing notations, we will write \( Z_i N_k R P \) to say that there is a proof of \( P \) of length at most \( k \) using as assumptions a finite subset of instances of the axiom schemata (20) to (35).

Following Definition 1.4 of Dowek et al. (2003), a theory \( T \) is said compatible with a rewrite system \( R \) if:
- \( P \vdash^R Q \) implies \( T \vdash_P P \Leftrightarrow Q \);
- for every formula \( P \in T \), we have \( P \vdash_P P \).

For instance, \( B \Rightarrow A \) is compatible with \( A \rightarrow A \lor B \): it possible to prove \( A \Leftrightarrow A \lor B \) assuming \( B \Rightarrow A \) with the proof:

\begin{align*}
\Rightarrow\cdot & \frac{A}{B} \quad \text{(i)} \\
\&\cdot & \frac{A \lor B}{A} \Rightarrow \text{(ii)} \\
\lor\cdot & \frac{A \Rightarrow A \lor B}{A \lor B} \Rightarrow \text{(iii)} \\
A \Leftrightarrow A \lor B
\end{align*}

(other cases of equivalent formulae can be derived from it), and reciprocally, \( B \Rightarrow A \) has the following proof modulo \( A \rightarrow A \lor B \):

\begin{align*}
\lor\cdot & \frac{B}{A} \Rightarrow \text{(i)} \\
\Rightarrow\cdot & \frac{A \lor B}{A} \Rightarrow \text{(ii)} \\
A \Leftrightarrow A \lor B
\end{align*}

Given a rewrite system, a compatible theory always exists, and one can show that proving modulo a rewrite system is the same as proving without modulo but using a compatible theory as assumptions (Dowek et al., 2003, Proposition 1.8).

4. Translations between schematic systems and natural deduction

Buss' theorem is true in schematic systems, but deduction modulo is defined for natural deduction. It is important to get bounded translations between these formalisms to show that the speed-ups we will be considering are not artifacts of the deductive system.

7
4.1. From $Z_i\hat{=}^S$ to $Z_i\hat{=}^N$

We want to translate a proof in the schematic system of $Z_i$ into a proof in pure natural deduction using as assumptions instances of the axiom schemata (20) to (35).

For the axiom schemata and inference rules of classical logic, we use the same translation as Gentzen, for instance the axiom schema (4) is translated into the natural deduction proof

\[
\begin{align*}
\Rightarrow &-e \quad B \quad (ii) \\
\Rightarrow &-e \quad A \quad (iii) \\
\Rightarrow &-i \quad A \Rightarrow C \quad (iii) \\
\Rightarrow &-i \quad B \Rightarrow A \Rightarrow C \quad (i) \\
\Rightarrow &-i \quad (A \Rightarrow B \Rightarrow C) \Rightarrow B \Rightarrow A \Rightarrow C \\
\end{align*}
\]

and the inference rule (19) into

\[
\begin{align*}
\exists &-e \quad \exists \alpha^j, B(\alpha^j) \quad (i) \\
\Rightarrow &-e \quad B(\beta^j) \quad (ii) \\
\Rightarrow &-i \quad A \quad (i) \\
\Rightarrow &-i \quad \exists \alpha^j, B(\alpha^j) \Rightarrow A \\
\end{align*}
\]

(note that the side condition ensure that it is possible to consider that what will be substituted for $\beta$ is free in $A$ and the assumptions of the proof above $B(\beta^j) \Rightarrow A$).

All these inference rules have a translation whose length does not depend on the formulæ finally substituted in the proof.

In a schematic system proof, there is also a finite number of instances of the axioms schemata for identity, Robinson’s axioms and induction and comprehension schemata. We keep these instances as assumptions in natural deduction, so that we obtain a proof in natural deduction using as assumptions a finite subset of instances of the axiom schemata (20) to (35), and whose length is linear compared to the schematic system proof:

**Proposition 2** It is possible to translate a proof of length $n$ in the schematic system for $Z_i$ into a proof of length $O(n)$ in (pure) natural deduction using assumptions in $Z_i$.

\[
Z_i \hat{=}^S P \leadsto Z_i \hat{=}^N O(n) P
\]

4.2. From $Z_i\hat{=}^N$ to $Z_i\hat{=}^S$

In this section, we consider a proof of $P$ in natural deduction, using as assumptions finite instances of (20) to (35) in the language of $Z_i$. We translate it into a proof in the schematic system for $Z_i$.

This is essentially a generalization of the translation from the $\lambda$-calculus to combinatory logic (see Curry, Feys, and Craig, 1958). We define mutually recursively two functions by induction on the inference rules: $T$ transforms a proof of $P$ in natural deduction using assumptions $\Gamma$ into a proof of $P$ in the schematic system (1) to (19) plus $\Gamma$. $T_A$ transform a proof of $P$ in natural deduction using assumptions $\Gamma, A$ into a proof of $A \Rightarrow P$ in the schematic system (1) to (19) plus $\Gamma$. The translation is:

\[
\begin{align*}
T \left( \Rightarrow -i \begin{bmatrix} A & \pi \{ B \} \\ A \Rightarrow B \end{bmatrix} \right) &= T \left( \begin{bmatrix} A & \pi \{ B \} \\ \end{bmatrix} \right) \\
T \left( \Rightarrow -e \begin{bmatrix} A & \pi \{ B \} \\ A \Rightarrow B \end{bmatrix} \right) &= T \left( \begin{bmatrix} A & \pi \{ B \} \\ B \end{bmatrix} \right) \\
T \left( \land -i \begin{bmatrix} A & \pi \{ B \} \\ A \land B \end{bmatrix} \right) &= T \left( \begin{bmatrix} A & \pi \{ B \} \\ A \land B \end{bmatrix} \right)
\end{align*}
\]
\[ T \left( \land_e \frac{\pi}{A \land B} \right) = \frac{T(\pi)}{A \land B} \quad \text{(17)} \]

and similarly with (7) for the other side.

\[ T \left( \lor_i \frac{\pi}{A \lor B} \right) = \frac{T(\pi)}{A \lor B} \quad \text{(17)} \]

and similarly with (10) for the other side.

\[ T \left( \lor_e \frac{\pi_1}{\lor x, A} \frac{\pi_2}{y/x} \frac{[y/x]A}{A} \right) = \frac{T(\pi_1)}{\exists x, A} \quad \text{(17)} \]

Note that the side conditions are satisfied.

\[ T \left( \land_e \frac{\pi}{A \lor (A \Rightarrow \bot)} \right) = \frac{T(\pi)}{A \lor (A \Rightarrow \bot)} \quad \text{(16)} \]

\[ T(\pi) \]

\[ T(A) = \frac{A}{A} \]

\[ T_A \left( \Rightarrow_i \frac{\pi}{B \Rightarrow C} \right) = \frac{T_B(\pi)}{B \Rightarrow C} \]

\[ T_A \left( \Rightarrow_e \frac{\pi_1}{B} \frac{\pi_2}{[B]} \right) = \frac{T_A(\pi_1)}{B \Rightarrow C} \quad \text{(17)} \]

\[ T_A(\pi) \]

\[ T_A \left( \land_e \frac{\pi_1}{B \land C} \frac{\pi_2}{[B \land C]} \right) = \frac{T_A(\pi_2)}{B \land C} \quad \text{(17)} \]

\[ T_A(\pi) \]

\[ T_A \left( \land_e \frac{\pi_1}{B \land C} \right) = \frac{T_A(\pi_2)}{B \land C} \quad \text{(17)} \]

\[ T_A(\pi) \]

and similarly with (7) for the other side.
\[ T_A \left( \pi_1 \frac { [A] } { B \lor C } \right) \quad \vdash \quad (17) \quad B \Rightarrow (B \lor C) \quad (9) \quad T_A (\pi) \quad \frac { A \Rightarrow B } { (B \Rightarrow (B \lor C)) \Rightarrow A \Rightarrow (B \lor C) } \quad \vdash \quad (17) \quad A \Rightarrow (B \lor C) \]

and similarly with (10) for the other side.

\[ T_A \left( \pi_1 \frac { [A] } { B \lor C } \pi_2 \frac { [A,B] } { D } \pi_3 \frac { [A,C] } { D } \right) \quad \vdash \quad (T_A (\pi)) \quad \frac { A \Rightarrow D } { (B \Rightarrow (B \lor C)) \Rightarrow A \Rightarrow (B \lor C) } \quad \vdash \quad (17) \quad A \Rightarrow (B \lor C) \]

Note that the side conditions are satisfied.

\[ T_A \left( \pi_1 \frac { [A] } { \forall x. B } \frac { (t/x)B } { \forall x. B } \right) \quad \vdash \quad (T_A (\pi)) \quad \frac { A \Rightarrow \forall x. B } { (\forall x. B) \Rightarrow (t/x)B } \quad (17) \quad \frac { (\forall x. B) \Rightarrow (t/x)B } { A \Rightarrow (t/x)B } \quad \vdash \quad (T_A (\pi)) \quad \frac { A \Rightarrow (t/x)B } { \forall x. B } \quad (17) \quad \frac { (t/x)B \Rightarrow \exists x. B } { A \Rightarrow \exists x. B } \quad \vdash \quad (T_A (\pi)) \quad \frac { A \Rightarrow \exists x. B } { A \Rightarrow C } \quad (19) \quad \frac { \exists x. B \Rightarrow A \Rightarrow C } { A \Rightarrow A \Rightarrow C } \quad \vdash \quad (17) \quad A \Rightarrow C \]

Note that the side conditions are satisfied.

\[ T_A \left( \pi_1 \frac { [A] } { \exists e. \bot B } \right) \quad \vdash \quad (T_A (\pi)) \quad \frac { A \Rightarrow \bot } { (A \Rightarrow \bot) \Rightarrow A \Rightarrow B } \quad (13) \quad \frac { A \Rightarrow B } { B \Rightarrow A } \quad (2) \quad (\text{if the assumption } A \text{ is not actually used in } \pi.)
\]

The definition of \( T_A \) for \( \Rightarrow \cdot i \) is not looping, because they are no longer \( \Rightarrow \cdot i \) in \( T_B (\pi) \). Nevertheless, this case impose use to define what \( T_A \) means for a proof using the inference rules (18) and (19). (The translation of (17) is already defined because (17) is equal to \( \Rightarrow \cdot e. \))

\[ T_A \left( \pi_1 \frac { [A] } { B \Rightarrow C(\tau) } \right) \quad \vdash \quad (17) \quad \frac { A \Rightarrow B \Rightarrow C(\tau) } { (A \Rightarrow B \Rightarrow C(\tau)) \Rightarrow (A \land B) \Rightarrow C(\tau) } \quad \vdash \quad (17) \quad \frac { (A \land B) \Rightarrow C(\tau) } { (A \land B) \Rightarrow \forall a. C(\alpha) } \quad \vdash \quad (17) \quad \frac { (A \land B) \Rightarrow \forall a. C(\alpha) } { A \Rightarrow B \Rightarrow \forall a. C(\alpha) } \]

10
where \( \varpi_1 \) is any proof of \((A \Rightarrow B \Rightarrow C) \Rightarrow (A \wedge B) \Rightarrow C\), and \( \varpi_2 \) of \((A \wedge B) \Rightarrow C \Rightarrow A \Rightarrow B \Rightarrow C\), using the axiom schemata (1) to (8) and the inference rule (17). (Indeed, they are valid formulae of the intuitionistic propositional logic.)

\[
T_A \left( \pi \left( \frac{[A]}{B(\tau) \Rightarrow C} \right) \right) \equiv (17) \quad \frac{T_A(\pi)}{A \Rightarrow B(\tau) \Rightarrow C} \quad \frac{(A \Rightarrow B(\tau) \Rightarrow C) \Rightarrow B(\tau) \Rightarrow A \Rightarrow C}{(A \Rightarrow B(\tau) \Rightarrow C) \Rightarrow B(\tau) \Rightarrow A \Rightarrow C} \quad \cdots \quad (4)
\]

It can be verified that this definition transforms a proof of size \( n \) into a proof of size \( O(3^n) \). Due to Cook and Reckhow (1979, Corollary 3.4), we could have found, at least for the propositional part, a polynomial translation. Nevertheless all we need in this paper is the fact that the increase of the proof length in the translation is bounded.

**Proposition 3** It is possible to translate a proof of length \( n \) in the (pure) natural deduction using assumptions in \( Z_i \) into a proof of length \( O(3^n) \) in the schematic system for \( Z_i \).

\[
Z_i \stackrel{\text{w} \cdot \tau}{\rightarrow} P \sim Z_i \frac{\delta}{O(3^n)} \frac{\text{w} \cdot \tau}{\rightarrow} P
\]

5. Higher-order arithmetic as a first-order theory modulo

5.1. Encoding higher order using classes

This time, we translate a proof in the schematic system for \( Z_i \) into a proof in natural deduction modulo using as assumption a finite theory, with the axioms in (20) to (33), so without the axiom schemata (26), (34) and (35) which are replaced by three new axioms. The point is that using modulo it is possible to work using a finite theory of arithmetic.

To translate these schemata, we will use the work of Kirchner (2006) which permits to express first-order theories in a finite number of axioms. The idea is to transform some metaformula \( A(t_1, \ldots, t_n) \) used in an axiom schema into a formula of the form \( \langle t_1, \ldots, t_n \rangle \in \gamma \) where \( \gamma \) will be some term representing what formula will be actually substituted for \( A \).

Following F. Kirchner’s method, we add new sorts \( \ell \) for lists and \( c \) for classes, as well as new function symbols and predicate

\[
1^j : j \quad \text{nil} : \ell \quad \cup : [c; c] \rightarrow c \quad 0 : c
\]

\[
S^j : [j] \rightarrow j \quad :: : [j; \ell] \rightarrow \ell \quad \cap : [c; c] \rightarrow c \quad \mathcal{P}^j : [c] \rightarrow c
\]

\[
\cdot[^j] : [j; \ell] \rightarrow j \quad \subseteq : [j; j + 1] \rightarrow c \quad \Rightarrow : [c; c] \rightarrow c \quad \mathcal{C}^j : [c] \rightarrow c \quad \in : [f; e]
\]

\((\alpha_1, \ldots, \alpha_n)\) will be syntactic sugar for \( \alpha_1 :: \beta_1 \cdots :: \alpha_n :: \text{nil} \) for the appropriate \( j_m \). We change the axiom schemata (26), (34) and (35) into the following axioms:

\[
\forall \gamma^c. \forall \alpha^0. \beta^0. \alpha^0 = \beta^0 \Rightarrow \langle \alpha^0 \rangle \in \gamma^c \Rightarrow \langle \beta^0 \rangle \in \gamma^c
\]

\[
(36)
\]

\[
\forall \gamma^c. \langle 0 \rangle \in \gamma^c \Rightarrow \langle 0 \rangle \in \gamma^c \Rightarrow \forall \beta^0. \langle \beta^0 \rangle \in \gamma^c \Rightarrow \langle \beta^0 \rangle \in \gamma^c \Rightarrow \forall \alpha^0. \langle \alpha^0 \rangle \in \gamma^c
\]

\[
(37)
\]

For all \( 0 \leq j < i \),

\[
\forall \gamma^c. \exists \alpha^{i+1}. \forall \beta^j. \beta^j \in \langle \beta^j \rangle \in \gamma^c \Rightarrow \langle \beta^j \rangle \in \gamma^c
\]

\[
(38)
\]

The rewrite system \( WS_i \) is then the following:
\[ \forall \alpha, \forall \beta, \alpha = \beta \Rightarrow (\alpha^0) \in \gamma^c \Rightarrow (\beta^0) \in \gamma^c \quad (36) \]

\[ \forall \alpha^0, \beta^0, \alpha = \beta \Rightarrow A(\alpha^0) \Rightarrow A(\beta^0) \]

\[ \forall \alpha^0. (0) \in \gamma^c \Rightarrow (\forall \beta^0. \beta^m \in \gamma^c \Rightarrow s(\beta^0)) \in \gamma^c \Rightarrow \forall \alpha^0. (0) \in \gamma^c \quad (37) \]

\[ A(0) \Rightarrow (\forall \beta^0. A(\beta^0) \Rightarrow A(s(\beta^0))) \Rightarrow \forall \alpha^0. (0) \in \gamma^c \quad (38) \]

Fig. 2. Translations of the axiom schemata (26), (34) and (35).

\[
\begin{align*}
l[\text{nil}] & \rightarrow t \\
V[t :: i] & \rightarrow t \\
S^j(n)[t :: i] & \rightarrow n[i]^j \\
s(n)[l]^0 & \rightarrow s(n[l]^0) \\
(t_1 + t_2)[l]^0 & \rightarrow t_1[l]^0 + t_2[l]^0 \\
(t_1 \times t_2)[l]^0 & \rightarrow t_1[l]^0 \times t_2[l]^0 \\
l \in \epsilon(t_1, t_2) & \rightarrow t_1[l]^0 \epsilon t_2[l]^0 + 1 \\
l \in \epsilon(t_1, t_2) & \rightarrow t_1[l]^0 \epsilon t_2[l]^0 + 1
\end{align*}
\]

This rewrite system has the following properties:

- It is finite (for a given \(j\)).
- It is terminating: let \(\mu(t)\) be the sum of the size of the left argument of all \(\cdot[\cdot]^j\) in \(t\) plus the sum of the size of the second argument of all \(\cdot\) if \(|t|\) denotes the size of \(t\),

\[
\mu(t) = \sum_{t_p=s[\cdot]} |s| + \sum_{t_p= \cdot s} |s| .
\]

Clearly, for all \(t\), \(\mu(t) \leq |t|^2\). Furthermore, if there is no nested \(\cdot[\cdot]^j\) nor \(\epsilon\), then \(\mu(t) \leq |t|^2\). Each step of rewriting strictly diminishes \(\mu\); if \(p \rightarrow q\) then \(\mu(p) > \mu(q)\). So \(R_i\) terminates in quadratic time, and all formulae of the form \((t_1, \ldots, t_n) \in t\) where \(t\) contains no \(\cdot[\cdot]^j\) can be rewritten in linear time.

- It is confluent: the only critical pairs, of the form:

\[
f(t_1, \ldots, t_n) \quad \mathcal{R}_i \quad f(t_1, \ldots, t_n)[n\text{nil}] \quad \mathcal{R}_i \quad f(t_1[n\text{nil}], \ldots, t_n[n\text{nil}]),
\]

are easily joinable.

- It is left-linear, i.e. variables appears only once on the left-hand side of each rule.

Proposition 2 of Kirchner (2006) says that it is possible, for any formula \(P\) of the language of \(i\)th-order arithmetic, to prove

\[
\exists E. \forall x_1, \ldots, x_n. (x_1, \ldots, x_n) \in E \Leftrightarrow P .
\]

Moreover, the proof of this proposition show us how to construct the witness \(E\). We will denote it by \(E_P^{x_1, \ldots, x_n} \).

Remark that no \(\cdot[\cdot]^j\) appears in it. Then, one can prove that \(\{t_1, \ldots, t_n\} \in E_P^{x_1, \ldots, x_n} \Leftrightarrow \{t_1/x_1, \ldots, t_n/x_n\} \).

For instance, consider the formula \(P \equiv x = 0 \lor \exists y. x \leq y\). Then \(E_P^x\) equals \(\{1, S(0)\} \cup \mathcal{P}^1(\mathbb{Z}^+(1, 1))\)

This proof \(E_P^x\) can be rewritten to \(t = 0 \lor \exists x. t \leq x\).

Consequently, the axiom schemata (26), (34) and (35) are replaced by the proofs in Figure 2. In these translations, we need to instantiate \(\gamma\) with some \(E_P^x\). It is well-known that the instantiations are the most problematic rules in deductive systems, at least for automated provers (e.g. they are what leads to nondeterminism and/or nontermination of tableau methods for first-order logic), because the instantiated term must be somehow guessed. Nevertheless, the instantiation here is entirely and automatically determined by the formula used in the schema, so that no harm is done.

Let \(FZ_i\) be the theory \((20)–(23), (27)–(33), (36)–(38)\), consisting only of a finite number of axioms.

Using this, a proof \(\pi\) of \(P\) in the schematic system for \(Z_i\), can be translated into a proof of \(P\) in natural deduction modulo \(WS_i\) with using assumptions in \(FZ_i\) whose length is linear compared to the length of \(\pi\).
Proposition 4 It is possible to translate a proof of length \( n \) in the schematic system for \( Z_i \) into a proof of length \( O(n) \) in the natural deduction modulo \( WS_i \) using assumptions in \( FZ_i \).

\[
Z_i \vdash F Z_i \ 
\sim 
F Z_i \vdash_{\text{FZ}_i} \ W S_i \ 
\]

This result can also be stated entirely in natural deduction.

Theorem 5 For all \( i \geq 0 \), there exists a (finite) rewrite system \( R_i \) such that for all formulæ \( P \), if \( Z_i \vdash P \) then \( FZ_i \vdash_{\text{FZ}_i} \ W S_i \vdash P \).

PROOF. We replace the instance of the axiom schemata (26), (34) and (35) by proofs using the axioms (36), (37) and (38) as indicated in Figure 2. \( \square \)

5.2. Higher-order arithmetic as purely computational theory

We define here higher-order arithmetic entirely as a rewrite system modulo whom inference will be applied. This is in line with the work of Dowek and Werner (2005) who express first-order arithmetic as a theory modulo. The idea is to combine this with the rewrite system of the previous section, to get a characterization of higher-order arithmetic. Notwithstanding, we will look carefully at the length of proofs in the translations.

Dowek and Werner (2005) use the following method to introduce the induction schema for first-order arithmetic: they add a new predicate \( N \) of arity \([0]\) which essentially says that an element is a natural number, and thus can be used in the induction schema. \( N(n) \) can therefore be rewritten to \( \forall p. \ 0 \in p \Rightarrow (\forall y. \ N(y) \Rightarrow y \in p \Rightarrow s(y) \in p) \Rightarrow n \in p \). Then, function symbols \( f^{y_1,\ldots,y_n}_P \) for each formula \( P \) of first-order arithmetic with free variables \( x, y_1, \ldots, y_n \) are added, as well as rewrite rules \( x \in f^{y_1,\ldots,y_n}_P (y_1, \ldots, y_n) \rightarrow P \).

To prove a formula using induction, we need to know that the variables used in the proof are natural numbers, hence the need for a translation \( \cdot \mid \cdot \) of the formulæ: \( \forall x. \ P \) is translated into \( \forall x. \ N(x) \Rightarrow [P] \), \( \exists x. \ P \) into \( \exists x. \ N(x) \land [P] \), the translation of the other logical connectors being just the connection of the translations of the subformulæ. Using this, it is proved (Dowek and Werner, 2005, Proposition 13) that we obtain a conservative extension of first-order arithmetic.

Nevertheless, the length of the proofs in the given translation is not conserved. Indeed, to translate a proof whose last step is

\[
\forall \cdot \ 
\varepsilon \pi \ 
\{t/x\} P
\]

we have to transform it into a proof

\[
\forall \cdot \ 
\varepsilon \pi \ 
\{t/x\} P
\]

\[
N(t) \ 
\Rightarrow \epsilon \ 
N(t) \Rightarrow \{t/x\} P
\]

The problem is that the length of the proof \( \pi \) depends on the size of \( t \).

Here we use a different approach: the induction axiom will be directly translated into the following rewrite rule

\[
 x \in p \rightarrow x \in p \vee (0 \in p \land \forall y. \ y \in p \Rightarrow s(y) \in p)
\]

This rule permits to express the induction schema as the intuitionistically equivalent schema \( \forall x. \ P(x) \leftrightarrow P(x) \lor (P(0) \land \forall y. \ P(y) \Rightarrow P(s(y))) \), combined with the rule for \( \pi \) of order \( 1 \). Note that doing so, we lose the confluence of our system.

Kirchner (2006) already applied his method to first-order arithmetic, to get a finite rewrite system, contrarily to Dowek and Werner (2005). The preceding rule that we used for the induction can be applied only on first-order terms, and therefore becomes \( x :{^0}\ \text{nil} \in p 
\Rightarrow \ 
(0 :{^0}\ \text{nil} \in p \land \forall y. \ y :{^0}\ \text{nil} \in p \Rightarrow s(y) :{^0}\ \text{nil} \in p) \).

Here we extend this method to all orders, as done in the previous section. We also need to express the comprehension schemata as a rewrite rule. This can be done by skolemization: we introduce new function
symbols \( \text{comp}^i : \{c\} \rightarrow j + 1 \) and the rewrite rule \( x \in^i \text{comp}^i(y) \rightarrow x ::^i \text{nil} \in y \). If we do not use (16) as axiom, we therefore obtain a formulation of higher-order Heyting arithmetic through the following rewrite system \( \mathcal{HA}_{\mathcal{A}^\text{mod}} \):

Arithmetic rules:

\[
\begin{align*}
\text{pred}(0) & \rightarrow 0 \\
\text{pred}(s(x)) & \rightarrow x \\
0 + y & \rightarrow y \\
s(x) + y & \rightarrow s(x + y)
\end{align*}
\]

Null(0) \rightarrow \top

\( s(x) + y \rightarrow s(x + y) \)

Null(s(x)) \rightarrow \bot

Axiom schemata:

\[
\begin{align*}
x = y & \rightarrow \forall z. (x \in z) \Rightarrow (y \in z) & x \in^i \text{comp}^i(y) & \rightarrow x ::^i \text{nil} \in y \\
 x ::^0 \text{nil} \in p & \rightarrow (x \in p) \vee ((0) \in p \land \forall y. (y \in p) \Rightarrow (s(y)) \in p) & \text{null} \in s \rightarrow \text{null} \in s
\end{align*}
\]

Substitutions and classes: we use \( \mathcal{WS}_i \), but as the signature is bigger, we also need the following rules:

\[
\begin{align*}
\text{pred}(\ell)[\ell]^0 & \rightarrow \text{pred}(\ell)[\ell]^0 & \ell \in \text{Null}(t) & \rightarrow \text{Null}(t)[\ell]^0
\end{align*}
\]

With this rewrite system, we can linearly simulate higher-order arithmetic in deduction modulo:

**Theorem 6** For all \( i \) there exists a finite rewrite system \( \mathcal{HA}_{\mathcal{A}^\text{mod}} \) such that for all formula \( P \) in the language of \( Z_i \), if \( Z_i \frac{N}{\text{k steps}} P \) then \( \frac{N}{\text{k steps}} \mathcal{HA}_{\mathcal{A}^\text{mod}} P \).

**PROOF.** It is sufficient to prove that all instances of the axiom schemata of \( Z_i \) can be proved in a bounded number of steps.

(20) can be proved by

\[
\begin{align*}
\Rightarrow i & \quad \langle \alpha^0 \rangle \in p^c (i) \\
\forall i & \quad \langle \alpha^0 \rangle \in p^c \Rightarrow \langle \alpha^0 \rangle \in p^c \\
\forall i & \quad \forall \alpha^0. \alpha^0 = \alpha^0 \\
\forall i & \quad \forall \alpha^0. \forall \alpha^0. \alpha^0 = \alpha^0 \\
\Rightarrow e & \quad \langle \alpha^0 \times \gamma^0 \rangle \in p^c \Rightarrow \langle \alpha^0 \times \gamma^0 \rangle \in p^c (i) \\
\forall e & \quad \langle \alpha^0 \rangle \in E_x \Rightarrow \langle \beta^0 \rangle \in E_x
\end{align*}
\]

(21) to (28) are proved using \( x = y \rightarrow \forall z. (x \in z) \Rightarrow (y \in z) \) in at most 8 steps. We give here the proof for (24), with \( E_x \updownarrow \vdash (S(\alpha^0) \times S(\gamma^0), 1 \times S(\gamma^0)) \)

\[
\begin{align*}
\Rightarrow i & \quad \langle \alpha^0 \times \gamma^0 \rangle \in p^e (i) \\
\forall i & \quad \langle \alpha^0 \times \gamma^0 \rangle \in p^e \Rightarrow \langle \alpha^0 \times \gamma^0 \rangle \in p^e \\
\Rightarrow e & \quad \langle \alpha^0 \times \gamma^0 \rangle = \alpha^0 \times \gamma^0 \\
\forall e & \quad \langle \alpha^0 \rangle \in E_x \Rightarrow \langle \beta^0 \rangle \in E_x
\end{align*}
\]

Let \( E \updownarrow \vdash (\hat{=} (1, S(0)) \supset \emptyset) \supset P(\hat{=} (S(1), s(1))) \). (29) is proved by

\[
\begin{align*}
\Rightarrow i & \quad \langle s(y) \rangle \in p^c (i) \\
\forall i & \quad \langle s(y) \rangle \in p^c \Rightarrow \langle s(y) \rangle \in p^c (i) \\
\Rightarrow e & \quad \langle s(y) \rangle = s(y) \\
\forall e & \quad \langle s(y) \rangle \in E \Rightarrow s(y) \in E
\end{align*}
\]

\[
\begin{align*}
\Rightarrow i & \quad \langle 0 \rangle \in p^c (ii) \\
\forall i & \quad \langle 0 \rangle \in p^c \Rightarrow \langle 0 \rangle \in p^c (ii) \\
\Rightarrow e & \quad 0 = 0 \Rightarrow \bot (i) \\
\Rightarrow i & \quad \langle s(y) \rangle = s(y) \Rightarrow \exists \beta^0. \alpha^0 = s(\beta^0) (i) \\
\forall i & \quad y \in E \Rightarrow s(y) \in E
\end{align*}
\]

\[
\begin{align*}
\Rightarrow i & \quad \langle 0 \rangle \in E \land \forall y. (y) \in E \Rightarrow (s(y)) \in E \\
\forall i & \quad \alpha^0 \in E \\
\forall i & \quad \forall \alpha^0. (\neg \alpha^0 = 0) \Rightarrow \exists \beta^0. \alpha^0 = s(\beta^0)
\end{align*}
\]
(30) to (33) are easy to prove using the arithmetical rules and the rule for =.

(34) has the following proof:

\[
\begin{align*}
\forall i & \forall \beta^0. P(\beta^0) \Rightarrow P(s(\beta^0)) \\
\forall i. (0) \in E_P^x & \land \forall \beta^0. (\beta^0) \in E_P^x \Rightarrow (s(\beta^0)) \in E_P^x \\
\forall i. (\alpha^0) \in E_P^x & \Rightarrow \forall \alpha^0. P(\alpha^0) \\
\Rightarrow i & P(0) \Rightarrow (\forall \beta^0. P(\beta^0) \Rightarrow P(s(\beta^0))) \Rightarrow \forall \alpha^0. P(\alpha^0) \\
\end{align*}
\]

(35) has the following proof:

\[
\begin{align*}
\Rightarrow i & \beta^j \in E_A^x (i) \\
\land i & \beta^j \in \text{comp}^1(E_A^x) \Rightarrow \beta^j \in E_A^x (i) \\
\Rightarrow i & \beta^j \in \text{comp}^1(E_A^x) \\
\forall i & \beta^j \in \text{comp}^1(E_A^x) \Leftrightarrow \beta^j \in E_A^x (i) \\
\exists i & \text{comp}^1(E_A^x) \Leftrightarrow A(\beta^j) \\
\exists i & 0 \alpha^{j+1}, \forall \beta^j, \beta^j \in \text{comp}^1(E_A^x) \Leftrightarrow A(\beta^j)
\end{align*}
\]

What we obtain is a conservative extension:

**Theorem 7** For all formula \( P \) in the language of \( Z_i \), if \( \mathcal{N} \models _{\mathcal{HHA}_i^\text{mod} } P \) then \( Z_i \models P \).

**PROOF.** For first-order arithmetic, the only difference between \( \mathcal{H} \) of Dowek and Werner (2005) and our system is the induction schema. If we translate \( N(n) \) into \( \forall p. 0 \in p \Rightarrow (\forall y. y \in p \Rightarrow s(y) \in p) \Rightarrow n \in p \), because we use an intuitionistically equivalent formulation of the induction schema, we can prove that we have a conservative extension of \( \mathcal{H} \) (of its variant in fact, see Dowek and Werner, 2005, Remark 2).

Then we apply the method of Kirchner (2006), which gives a conservative extension. Finally we skolemize the axioms corresponding to the comprehension schemata, and therefore we obtain a conservative extension (see Hermant, 2006). \( \Box \)

**Note 1** With the rule that we use for arithmetic, we cannot extend the proof of the normalization through reducibility candidates as done by Dowek and Werner (2005), or through super consistency by Dowek (2006). This remains currently an open question whether our system normalizes or not.

6. Applications to proof-length speed-ups

Because of Theorem 5 and Theorem 6, there is obviously no proof-length speed-up between \( Z_i \), \( FZ_i \) modulo \( WS_i \), and \( \emptyset \) modulo \( \mathcal{H} \). Furthermore, there exists a speed-up between all these and \( Z_{i-1} \), which can be decomposed as follow.

6.1. Speed-up over compatible theories

In this section, we prove that there exists a speed-up between \( FZ_i \) modulo \( WS_i \) and \( FZ_i \) plus any finite theory compatible with \( WS_i \). But first, we prove that there is no need for a complicated rewrite system to get such a speed-up.

**Proposition 8** Consider the rewrite system \( R \) consisting only of the rule \( s(x) + y \rightarrow x + s(y) \), there is an infinite family \( F \) such that such that for all finite theories \( T \) compatible with \( R \),

(i) for all \( P \in F \), \( T \overset{\text{1 step}}{\models}_R P \)

(ii) there is a fixed \( k \in \mathbb{N} \) such that for all \( P \in F \), \( \overset{\text{k steps}}{\models}_R P \)

(iii) there is no fixed \( k \in \mathbb{N} \) such that for all \( P \in F \), \( T \overset{\text{k steps}}{\models}_R P \)
PROOF. Consider the family of formulae $P(n + n) \Rightarrow P(n + n)$ for some unary predicate $P$, where $n$ denotes the usual representation of the natural number $n$ using 0 and $s$. Then it is quite clear that $\frac{n}{\mathcal{R}} P(n + n) \Rightarrow P(n + n)$. Let $\mathcal{T}$ be a finite theory compatible with $\mathcal{R}$. By definition $\mathcal{T} \models \frac{n}{\mathcal{R}} P(n + n) \Rightarrow P(n + n)$, but it is impossible to find a proof that takes less than $O(n)$ steps. (In the theory we may have some formulæ such as $s^m(x) + y = x + s^n(y)$ but they will only divides the minimal number of steps by $m$, and we can only have a finite number of such formulæ. The theorem is of course wrong if infinite theories are allowed, because one could add $\mathcal{F}$ to some theory compatible with $\mathcal{R}$ to get proofs with a bounded number of steps.) □

This is therefore no surprise that, if we consider $FZ_i$ plus a finite theory compatible with $\mathcal{WS}_i$, we get a speed-up with $Z_i$ (or with $FZ_i$ modulo $\mathcal{WS}_i$). That shows the interest of using deduction modulo. (The same kind of proof permits also to show the existence of a speed-up between $\mathcal{HHA}^{\mathit{mod}}_i$ and $FZ_i$ plus a finite theory compatible with $\mathcal{WS}_i$.)

**Proposition 9** For all $i$, there is an infinite family $\mathcal{F}$ such that such that for all finite theories $\mathcal{T}_i$ compatible with $\mathcal{WS}_i$,

(i) for all $P \in \mathcal{F}$, $FZ_i, \mathcal{T}_i \models P$

(ii) there is a fixed $k \in \mathbb{N}$ such that for all $P \in \mathcal{F}$, $FZ_i, \mathcal{T}_i \models {}^k P$

(iii) there is no fixed $k \in \mathbb{N}$ such that for all $P \in \mathcal{F}$, $FZ_i, \mathcal{T}_i \models {}^k P$

**PROOF.** Consider the set of formulæ corresponding to all instantiations of the comprehension schema for $j = i - 1$. Obviously, $Z_{i-1}$ is not enough to prove all of them, so that (38) has to be used in the proofs in $FZ_i, \mathcal{T}_i$. Nevertheless, the term of sort $c$ instantiated in it cannot have a bounded size. Then, the decomposition of this term using the finite theory $\mathcal{T}_i$ compatible with $\mathcal{WS}_i$ cannot be done in a bounded number of steps. In $FZ_i$ modulo $\mathcal{WS}_i$, this formulæ can be proved in one step as done in Fig. 2. □

6.2. Speed-up in arithmetic modulo

It is also possible to get a speed-up between $FZ_i$ plus any theory compatible with $\mathcal{WS}_i$ and $Z_{i-1}$.

**Proposition 10** For all $i > 0$, there is an infinite family $\mathcal{F}$ such that for all theory $\mathcal{T}_i$ compatible with $\mathcal{WS}_i$,

(i) for all $P \in \mathcal{F}$, $Z_{i-1} \models P$

(ii) there is a fixed $k \in \mathbb{N}$ such that for all $P \in \mathcal{F}$, $FZ_i, \mathcal{T}_i \models {}^k P$

(iii) there is no fixed $k \in \mathbb{N}$ such that for all $P \in \mathcal{F}$, $Z_{i-1} \models {}^k P$

**PROOF.** If we look at Buss’ proof of Theorem 1, the infinite family of formulæ he use are of the form $P(n)$ where $\forall n. P(n)$ can be proved in $Z_i$ whereas in $Z_{i-1}$, $P(n)$ can be proved, but not with less than $n$ steps. So to get a speed-up it is sufficient to prove that $\forall n. P(n)$ can be proved in $FZ_i$ plus $\mathcal{T}_i$, which is the case because of Theorem 5. We also need Proposition 3 to show that if the length of the proofs in $Z_{i-1}$ was bounded, it would be the same in $Z_{i-1}$, hence a contradiction with Theorem 1. □

We can summarize with the following schema:
7. Conclusion and perspectives

We have shown how to encode higher-order arithmetic through the rewrite system $\mathcal{HHA}^{\text{mod}}$ without inflating the length of proofs. In particular, using a generalization of the work of Kirchner (2006), we have translated higher-order axiom schemata into many-sorted first-order axioms modulo $\mathcal{WS}$. Using this, we have proved that the speed-up in arithmetic already occurs using a first-order finite axiomatization of $\mathbb{Z}_{i+1}$ (Proposition 10), but that it is better to express this theory as a rewrite system modulo which rules will be applied (Proposition 9). This shows the power of separating computation and deduction. This kind of speed-ups must not be considered as cheating, by hiding part of the proofs in the congruence. This must be thought of as a way to separate what is deduced and what is computed. To find a proof, both parts need to be built. To check the proof however, only the deductive part is necessary, because the rest can be effectively computed during the verification (hence the need to have a decidable congruence, even better if it is determined by simple deterministic algorithm). This can be applied to automated and interactive theorem proving, as well as in representation of proofs in natural language (where all computational details are often implicitly left the reader).

The fact that the difference between $i^{\text{th}}$-order and $i-1^{\text{th}}$-order arithmetic can be expressed as computation is not surprising, because, if one looks carefully, the proof of Theorem 1 given by Buss (1994) deeply relies on the fact that it is possible to define some truth predicate for the formulæ of the preceding order. Therefore, in a sense, it is possible, in $i+1^{\text{st}}$-order arithmetic, to compute the validity of a formula in $i^{\text{th}}$-order arithmetic.

It should be looked if this truth predicate could be expressed as a rewrite system, therefore relating more strongly Buss’ proof and the speed-up obtained using deduction modulo.

These results are encouraging indicators that it is as good to work directly in higher-order logics, as is done in the current interactive theorem provers, such as Coq (2006) or Isabelle/HOL (Nipkow, Paulson, and Wenzel, 2002), or using a first-order implementation of these logics, as could be done in a proof assistant based on deduction modulo (or on its sequel named superdeduction developed by Brauner, Houtmann, and Kirchner, 2007). This paper gives clues to answer positively this question. It remains to be proved that such results extends to pure higher-order logic. A direction to do so would be to look directly at the difference of the lengths of proofs in the expression of HOL in the sequent calculus modulo (Dowek et al., 2001), or of every PTS in $\lambda \Pi$ modulo (Cousineau and Dowek, 2007).

Acknowledgments. The author wishes to thank G. Dowek, T. Hardin and C. Kirchner for many discussions and comments about this paper.

References


