Log. Univers.

https://doi.org/10.1007/s11787-021-00278-1

Logica Universalis

k-Provability in PA

Paulo Guilherme Santos and Reinhard Kahle

Abstract. We study the decidability of k-provability in PA—the relation 'being provable in PA with at most k steps'—and the decidability of the proof-skeleton problem—the problem of deciding if a given formula has a proof that has a given skeleton (the list of axioms and rules that were used). The decidability of k-provability for the usual Hilbert-style formalisation of PA is still an open problem, but it is known that the proof-skeleton problem is undecidable for that theory. Using new methods, we present a characterisation of some numbers k for which k-provability is decidable, and we present a characterisation of some proof-skeletons for which one can decide whether a formula has a proof whose skeleton is the considered one. These characterisations are natural and parameterised by unification algorithms.

Mathematics Subject Classification. Primary 03B10; Secondary 03B25.

Keywords. k-provability, Peano arithmetic, Proof-skeleton problem, Decidability.

1. Introduction

k-provability is the notion of provability '$\vdash_{k\text{steps}}$', i.e. the notion of *being provable, in a certain theory, with at most k steps.* This notion has been studied for different theories and with different purposes. In [6,11], and [8] the decidability of this relation was studied for several formalisations of Peano arithmetic (PA). Kreisel's conjecture—an open problem in k-provability [4]—was studied in [3,7–9,11,12], and [1]. We recommend [10] for a detailed account of this and other notions of provability.

In [2], it was proved that k-provability is undecidable for the sequent calculus of arithmetic with an infinite number of relation-symbols. Furthermore,

This work was funded by the following FCT-projects: Centro de Matemática e Aplicações (UIDB/00297/2020), and Bolsa de Doutoramento (SFRH/BD/143756/2019). The research was also supported by the Udo Keller Foundation. This work was awarded the Prèmio de Lõgica Amìlcar Sernadas 2020, the Portuguese logical prize of UNILOG.

Published online: 09 June 2021

in [6], this relation was proved to be decidable for several formulations of PA where the universal instantiation schema is replaced by other schemata. The usual universal instantiation schema is:

Uni. Inst $(\forall x.\varphi) \to \varphi^x_t$, where t is substitutable for x in φ.

It is an open problem whether k-provability for PA with the usual instantiation schema is decidable [6]. From [6,8], and [10, p. 103] we know that the proof-skeleton problem is undecidable for PA with the usual instantiation schema; by *proof-skeleton problem* we mean the problem of deciding if a given formula has a proof whose skeleton (the list of axioms and rules that were used) is the considered one.

In this paper, we will address the proof-skeleton problem and k-provability; we will:

1. Characterise some proof-skeletons for which it is decidable whether a given formula has a proof with the considered skeleton;
2. Characterise some values of k for which it is decidable whether a formula can be proven in k steps.

These characterisations are natural—in the sense that they emerge from simple generalisation of concepts—and parameterised by unification algorithms (for a type of systems that we are going to develop). Our approach is valid for several theories that extend PA. We will consider theories of arithmetic formulated in Hilbert-style systems having the following logical axioms (see [5, p. 112] for further details):

L1) $(\varphi \to (\psi \to \mu)) \to ((\varphi \to \psi) \to (\varphi \to \mu))$;
L2) $\varphi \to (\psi \to \varphi)$;
L3) $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$;
L4) $(\forall x.\varphi) \to \varphi^x_t$, where t is substitutable for x in φ;
L5) $\forall x.(\varphi \to \psi) \to (\forall x.\varphi \to \forall x.\psi)$;
L6) $\varphi \to \forall x.\varphi$, where x does not occur free in φ;
L7) $\forall x.x = x$;
L8) $\forall x.\forall y.\forall z.(x = y \wedge y = z \to x = z)$;
L9) $\forall x.\forall y.x = y \to y = x$;
L10) $\forall x_0.\forall x_1.\forall x_2.\forall x_3.(x_0 = x_1 \wedge x_2 = x_3 \to x_0 + x_2 = x_1 + x_3)$;
L11) $\forall x.\forall y.(x = y \to S(x) = S(y))$.

We do not allow the occurrence of any other predicates besides '=' (for instance, we assume that one is not given a predicate '$<$' for the usual relation $<$ in $\mathbb{N}$). Furthermore, we consider the following two rules:

$$\frac{\varphi\varphi \to \psi}{\psi}\text{MP} \qquad\qquad \frac{\varphi}{\forall x.\varphi}\text{Gen}$$

It is important to observe that these axioms are *schemata* in the sense that they can be substituted by any formula and any variable which satisfy certain conditions. The non-logical axioms of Robinson arithmetic (Q) are:

Q1) $\forall x.\forall y.(S(x) = S(y) \to x = y)$;
Q2) $\forall x.\neg 0 = S(x)$;
Q3) $\forall x.x + 0 = x$;

Q4) $\forall x.\forall y.x + S(y) = S(x + y)$;
Q5) $\forall x.x \times 0 = 0$;
Q6) $\forall x.\forall y.x \times S(y) = (x \times y) + x$;
Q7) $\forall x.(\neg x = 0 \to \exists y.x = S(y))$.

PA is obtained from Q by adding the induction schema:

PA1) $\varphi_0^y \wedge \forall x.(\varphi_x^y \to \varphi_{S(x)}^y) \to \forall x.\varphi_x^y$, where y is free in φ and x is substitutable for y in φ.

Observe that we are considering the signature of the logic as only having the universal quantifier, implication sign, and negation sign: whenever another connective appears, it should be written using only implication and negation signs; for instance $\varphi \wedge \psi := \neg(\varphi \to \neg\psi)$. Other options could have been made here.

2. The Theory PA′

In this section we develop a version of PA, namely PA′. For that, we will present some useful results.

Theorem 2.1. *The schema*

Inst. 1 $(\forall x.\varphi) \to \varphi_t^x$, *where t is substitutable for x in φ*

has the same instances as the two following schemata considered together:

Inst. 2 $(\forall x.\varphi) \to \varphi_t^x$, *where t is substitutable for x in φ and x does not occur in t;*

Inst. 3 $(\forall x.\varphi_x^y) \to \varphi_t^y$, *where t is substitutable for y in φ, x does not occur free in φ, the variable y does not occur free under the scope of a $\forall x$ quantifier in φ, the variable y is not the variable x, and y does not occur in t.*

Proof. Let us analyse the following cases:

Inst. 2 $\Longrightarrow$ **Inst. 1** This is immediate, since all instances of **Inst. 2** are directly instances of **Inst. 1**.

Inst. 3 $\Longrightarrow$ **Inst. 1** Suppose that one is given $\mu := (\forall x.\varphi_x^y) \to \varphi_t^y$, where t is substitutable for y in φ, x does not occur free in φ, the variable y does not occur free under the scope of a $\forall x$ quantifier in φ, the variable y is not the variable x, and y does not occur in t. Take $\xi := \varphi_x^y$. Consider the two following situations:

y **does not occur free in** φ For this case, $\xi = \varphi_x^y = \varphi = \varphi_t^y$. As x does not occur free in φ, we conclude that $\xi_t^x = \xi$ and t is substitutable for x in φ, thus t is substitutable for x in ξ. Consequently,

$$\mu = ((\forall x.\varphi_x^y) \to \varphi_t^y) = ((\forall x.\xi) \to \xi) = ((\forall x.\xi) \to \xi_t^x),$$

so μ is an instance of **Inst. 1**.

y **occurs free in** φ Suppose, aiming a contradiction, that t is *not* substitutable for x in $\xi = \varphi_x^y$. Then, x occurs free in ξ and there is a variable z in t which is captured by a quantifier $\forall z$ in ξ_t^x. As x do not occur free in φ by hypothesis, this means that there is a variable z in t which is captured by a quantifier $\forall z$ in φ_t^y; which contradicts the fact that t is substitutable for y in φ. So, t is substitutable for x in ξ. As the variable y does not occur free under the scope of a $\forall x$ quantifier and x does not occur free in φ, we conclude that $\xi_t^x = (\varphi_x^y)_t^x = \varphi_t^y$. Hence,

$$\mu = ((\forall x.\varphi_x^y) \to \varphi_t^y) = ((\forall x.\xi) \to \xi_t^x),$$

and so μ is an instance of **Inst. 1**.

Inst. 1 $\implies$ **Inst. 2,Inst. 3** Consider $\mu' := (\forall x.\varphi) \to \varphi_t^x$, where t is substitutable for x in φ. Consider the following cases:

x **does not occur in** t In this case, μ' is an immediate instance of **Inst. 2**.

x **occurs in** t Take $\chi := \varphi_y^x$, where y is a fresh variable not occurring in φ (not even in the quantifiers of φ) and in t. Clearly, the variable y does not occur free under the scope of a $\forall x$ quantifier in χ. As t is substitutable for x in φ and y does not occur in φ, it follows that t is substitutable for y in χ. Furthermore, x does not occur free in χ. It is clear that $\chi_x^y = \varphi$. As y does not appear in φ (not even in quantifiers of φ) and all free occurrences of x in φ are being replaced by y in χ, we have that $\chi_t^y = \left(\varphi_y^x\right)_t^y = \varphi_t^x$. Thus,

$$((\forall x.\chi_x^y) \to \chi_t^y) = ((\forall x.\varphi) \to \varphi_t^x) = \mu',$$

and so μ' is an instance of **Inst. 3**.

The result follows by the previous case analysis. □

Theorem 2.2. *The following schemata have the same instances:*

Ind. 1 $\varphi_0^y \wedge \forall x.(\varphi_x^y \to \varphi_{S(x)}^y) \to \forall x.\varphi_x^y$, *where* y *is free in* φ *and* x *is substitutable for* y *in* φ*;*

Ind. 2 $\varphi_0^y \wedge \forall x.(\varphi_x^y \to \varphi_{S(x)}^x) \to \forall x.\varphi_x^y$, *where* y *is free in* φ*, the variable* x *is not the variable* y*, and* x *is substitutable for* y *in* φ*.*

Proof. We have the following cases to study:

Ind. 2 $\implies$ **Ind. 1** It is immediate, since the instances of **Ind. 2** are directly instances of **Ind. 1**.

Ind. 1 $\implies$ **Ind. 2** Suppose that $\mu := (\varphi_0^y \wedge \forall x.(\varphi_x^y \to \varphi_{S(x)}^y) \to \forall x.\varphi_x^y)$ is an instance of **Ind. 1**. If x is not y, then it is immediately an instance of **Ind. 2**. So, suppose that x is y in μ. Thus, $\mu = (\varphi_0^x \wedge \forall x.(\varphi \to \varphi_{S(x)}^x) \to \forall x.\varphi)$.

Take y a fresh variable not occurring in φ (not even in quantifiers) and $\psi := \varphi^x_y$. As x is free in φ and y does not appear at all in φ, it follows that y is free in ψ. As x is free in φ, we conclude that x is substitutable for y in ψ. Furthermore, as y does not occur at all in φ, $\psi^y_x = \left(\varphi^x_y\right)^y_x = \varphi$, $\psi^y_0 = \left(\varphi^x_y\right)^y_0 = \varphi^x_0$, and $\psi^y_{S(x)} = \left(\varphi^x_y\right)^y_{S(x)} = \varphi^x_{S(x)}$. Therefore, we have

$$\mu = (\varphi^x_0 \wedge \forall x.(\varphi \rightarrow \varphi^x_{S(x)}) \rightarrow \forall x.\varphi) = (\psi^y_0 \wedge \forall x.(\psi^y_x \rightarrow \psi^y_{S(x)}) \rightarrow \forall x.\psi^y_x)$$

All the cases were considered. □

Now we define PA′.

Definition 2.3. Let PA′ be PA from before where the universal instantiation axiom is replaced by the schemata **Inst. 2** and **Inst. 3**, and where the induction axiom is replaced by **Ind. 2**.

Theorem 2.4. *The two following statements are equivalent:*

1. PA $\vdash_{ksteps} \varphi$;
2. PA′ $\vdash_{ksteps} \varphi$.

Proof. From theorems 2.1 and 2.2, we know that the axioms in PA′ that are a replacement of the axioms of PA have exactly the same instances. So, in a proof of PA an occurrence of **Inst. 1** can be replaced by an occurrence of **Inst. 2** or **Inst. 3** to obtain a proof in PA′ exactly with the same formulas, in particular with the same length. The same idea applies to substitutions of **Ind. 1** in proofs of PA by **Ind. 2** to obtain proofs in PA′. Furthermore, **Inst. 2** and **Inst. 3** can be replaced by **Inst. 1** in the same fashion, and **Ind. 2** by **Ind. 1**. □

By the previous result, we know that the decidability of PA $\vdash_{k\text{steps}}$ reduces to the decidability of PA′ $\vdash_{k\text{steps}}$. We will consider the axioms **Inst. 2** and **Inst. 3**, and **Ind. 2**—they have the nice syntactical feature that in the replacements one cannot have a variable being substituted by a term where that very variable occurs. We are considering the number of steps as being the number of rules that are being applied—here one could also consider the number of proof lines, the results that we are going to present can be adapted for that situation.

3. Main Results

We were inspired by Parikh systems (see, for instance, [6]) for the systems that we have developed, but we use very similar terminology to the one used in [6] with very different meanings (the reader should always have this in mind). The biggest difference between our approach and the approach followed in [6] is that we have developed a general way to obtain the provable formulas via schemata and in the latter the authors' focus in schemata occurs mainly in the axioms (they do not extend that notion to the provable formulas as

we do). In that paper, it was developed a technique to study the decidability of k-provability for some theories using unification. We will develop a new technique that depends on a different way to unify—we create a technique to unify some of the schemata that generate the provable formulas.

3.1. Provable Schemata

The general idea of our approach is to attach a meaning to the combinatorial nature of general proof structures, namely to the different ways to combine, in a given number of steps, the axioms of the considered theory. Let us see, as an example, the general structure that corresponds to the following arrangement of the axioms: $\mathrm{MP}([L]), \mathrm{MP}([L]), [L]))$. This means that one firstly applies MP to an L1) implication using an L2) axiom, and to the result of that, which must be an implication, one applies an axiom of the form L2). Starting from the first application, to apply to the left side of $(\varphi \to (\psi \to \mu)) \to ((\varphi \to \psi) \to (\varphi \to \mu))$ something of the form of $\varphi \to (\psi \to \varphi)$, one must have $\mu = \varphi$. Hence, the application of the first MP yields something of the form $(\varphi \to \psi) \to (\varphi \to \varphi)$. Now, to apply L2) to $(\varphi \to \psi) \to (\varphi \to \varphi)$, one needs $\psi = \xi \to \varphi$. In this conditions, the general shape/structure of $\mathrm{MP}([L]), \mathrm{MP}([L]), [|rmL)]))$ is $\varphi \to \varphi$.

Clearly, any other way to arrange the axioms in the considered shape is a particular case of $\varphi \to \varphi$. Moreover, $\varphi \to \varphi$ codifies, in a unique schema, all the ways to combine the axioms in $\mathrm{MP}([L]), \mathrm{MP}([L]), [L]))$. A similar analysis could be carried out for (some of) the other combinations of axioms, resulting in a finite list of schemata that generate, via substitutions, all instances of the other schemata that are obtainable in a given number of steps, k (this only works for some values of k due to undecidability issues). Hence, for some values of k, there are finitely many provable schemata that give rise to the formulas that are provable in k steps. It is important to observe that the previous idea does not work for all proof-skeletons (see theorem 5.1 from [8], and theorem 14.1 from [10, p.103]).

Now we move to formalise the previous ideas. Having in mind what was previously observed, the general shape of a schema is nothing but $F[\varphi_0, \ldots, \varphi_{n_0}, t_0, \ldots, t_{n_1}, v_0, \ldots, v_{n_2}]\&C$, where $\varphi_0, \ldots, \varphi_{n_0}$ stand for formula-variables, where $t_0, \ldots, t_{n_1}$ stand for term-variables, and where $v_0, \ldots, v_{n_2}$ stand for variable-variables; F stands for the arrangement of the logical symbols; and C stands for a condition on the variables, on the formulas, and on the terms. All the variables in the previous schema are exactly that, variables, they do not stand for actual entities; for instance, the formula-variables do not stand for actual formulas. Let us see two examples:

- '$\varphi \to (\psi \to \varphi)$' is a schema, where $F[\varphi_0, \varphi_1] := \varphi_0 \to (\varphi_1 \to \varphi_0)$ only has formula-variables, and where there are no conditions;
- '$\varphi_0^x \wedge \forall y.(\varphi_y^x \to \varphi_{S(y)}^x) \to \forall y.\varphi_y^x$, where y is free in φ, the variable x is not the variable y, and x is substitutable for y in φ' is a schema where $F[\varphi, x] := \varphi_0^x \wedge \forall y.(\varphi_y^x \to \varphi_{S(y)}^x) \to \forall y.\varphi_y^x$, and $C :=$'y is free in φ, the variable x is not the variable y, and x is substitutable for y in φ'.

Let us fix throughout the rest of the paper: $C_0(\varphi, x)$:='x is free in φ', $C_1(\varphi, x)$:='x is not free in φ', $C_2(\varphi, t, x)$:='t is substitutable for x in φ', $C_3(t, x)$:='x does not occur in t', and $C_4(x, y)$:='the variable x is different from the variable y', $C_5(\varphi, x, y)$:= 'the variable y does not occur free under the scope of a $\forall x$ quantifier in φ', $C_6(\varphi, x)$:= 'x occurs in φ', $C_7(\varphi, x)$:='x does not occur free in φ', and $C_8(\varphi, x, y)$:='there is a a free occurrence of the variable y in φ that does not occur under the scope of a $\forall x$ quantifier'. Observe that all the previous conditions are decidable. We will assume that T is a (fixed) theory of arithmetic that extends the presented version of PA$'$ by adding schemata that depend on formulas, terms, and variables, without having any conditions on the schemata. We now move to define formally what a proof-skeleton is.

Definition 3.1. We define inductively the notion of *proof–skeleton*:

Basis case A_i is a proof–skeleton if A_i is the number of an axiom of T;

Induction step If $\mathcal{S}_0$ and $\mathcal{S}_1$ are proof–skeleton, then $\mathrm{MP}(\mathcal{S}_0, \mathcal{S}_1)$ and $\mathrm{Gen}(\mathcal{S}_0)$ are proof–skeleton.

We say that a proof–skeleton $\mathcal{S}$ has k steps if it has k applications of rules (k might be 0).

Now we define the notion of a schema.

Definition 3.2. We define inductively the notion of *term-structure*:

Basis case Every variable-variable, every term–variable, and 0 are a term–structure.

Induction step If r and s are term-structures, then $S(r)$, $r + s$, $r \times s$, and $s^{\vec{x}}_{\vec{t}}$, where $\vec{x}$ are variable-variables and $\vec{t}$ are term–structures, are also term-structures.

We define inductively the notion of *formula-structure*:

Basis case Every formula-variable is a formula-structure. Furthermore, $r = s$ is a formula-structure, with r and s term-structures.

Induction step If F and G are formula-structure, then the following are formula-structures:

- $F \to G$,
- $\neg F$,
- $\forall v.F$, where v is a variable–variable,
- $(F)^{\vec{v}}_{\vec{t}}$, where $\vec{v}$ are variable-variables, and $\vec{t}$ are a term-structure.

We say that F is a *sub-formula-structure* of G if F is a formula-structure that occurs in G. We say that A is an *atom* if A is a formula-variable, or if $A = \varphi^{x_0 \ldots x_\ell}_{t_0 \ldots t_\ell}$, where φ is either a formula-variable or a formula-structure of the form $r = s$, with r and s term-structures. We say that an expression of the form

$$F[\varphi_0, \ldots, \varphi_{n_0}, t_0, \ldots, t_{n_1}, v_0, \ldots, v_{n_2}] \& \bigvee_{i \in I} \&_{j \in J_i} \sim^{k^0_j} C_{k^1_j}(A_i, t_{k^2_j}, v_{k^3_j})$$

or of the form

$$F[\varphi_0, \ldots, \varphi_{n_0}, t_0, \ldots, t_{n_1}, v_0, \ldots, v_{n_2}] \& \bot$$

is a *schema*, where I and J_i are sets of indices (possibly empty, in which case we omit the conditions), $F[\varphi_0, \ldots, \varphi_{n_0}, t_0, \ldots, t_{n_1}, v_0, \ldots, v_{n_2}]$ is a formula-structure, and A_i are atoms. Here $C_{k_j^1}$ stand for a (syntactical) representation of the condition in the theory T previously mentioned (we also allow formula-structures to occur inside the conditions, but this will be avoided using several conventions). We allow term-structures to occur inside the conditions.

Every axiom of PA$'$ is a schema. Furthermore, every axiom of T is a schema.

Convention 3.3. In every occurrence of $\varphi_{\vec{s}}^{\vec{x}}$ or $t_{\vec{s}}^{\vec{x}}$ in schemata, we do not allow the variables that are being changed to occur in the replacing term. Furthermore, we do not allow a variable to occur in a replacement being mapped to different terms and we do not allow repeated occurrences of the same change in the replacement (for instance $t_{s\ s}^{x\ x}$).

It is important to stress that schemata are syntactical objects, even the conditions in them are syntactical (that have a semantical interpretation). The symbols $\sim$, $\bigvee$, and $\&$ are syntactical representations of the connectives in the meta-language (negation, disjunction, and conjunction, respectively).

Convention 3.4. As $(\varphi \to \psi)_t^x = \varphi_t^x \to \psi_t^x$, $(\neg\varphi)_t^x = \neg\varphi_t^x$, and

$$(\forall y.\varphi)_t^x = \begin{cases} \forall y.\varphi, & x = y \\ \forall y.\varphi_t^x, & x \neq y, \end{cases}$$

hold for all formulas, we will assume these identities for schemata. This means that in a schema one can move all occurrences of $(\cdot)_t^x$ inside the formula–structure.

With the conventions that we are going to present, we will extend the syntactical equality (and we will continue to denote it simply by '='). Sometimes to emphasise that $\mathcal{E}_0 = \mathcal{E}_1$ syntactically we will say that '$\mathcal{E}_0$ is $\mathcal{E}_1$' (we will also use it to express that $\mathcal{E}_0$ and $\mathcal{E}_1$ are syntactically the same after some suitable substitution).

Convention 3.5. We assume the following identities:

- $\varphi_{t_0 \cdots t_n}^{x_0 \cdots x_n} = \varphi_{t_{f(0)} \cdots t_{f(n)}}^{x_{f(0)} \cdots x_{f(n)}}$, where $f : \{0, \ldots, n\} \to \{0, \ldots, n\}$ is a bijection (this feature is not troublesome because we are interpreting the replacements as simultaneous replacements that satisfy convention 3.3).
- The usual properties of replacements, for example:

$$(x + S(s))_t^x = (t + S(s_t^x)).$$

- The usual identities for propositional (meta-)logic, for example:
  - $\sim^{2n} C = C$, $\sim^{2n+1} C = \sim C$,
  - $C \bigvee C = C$,

  - $\sim (C \,\&\!\!\!\&\, C') = (\sim C) \bigvee (\sim C')$ (where C and C' are conditions), and so on.
- For C_0:
  - $C_0(\neg F, x) = C_0(F, x)$,
  - $C_0(F \to G, x) = C_0(F, x) \,\&\!\!\!\&\, C_0(G, x)$,
  - $C_0(\forall y.F, x) = C_0(F, x) \,\&\!\!\!\&\, C_4(x, y)$.
- $C_1(F, x) = \sim C_0(F, x)$.
- For C_2:
  - $C_2(\neg F, t, x) = C_2(F, t, x)$,
  - $C_2(F \to G, t, x) = C_2(F, t, x) \,\&\!\!\!\&\, C_2(G, t, x)$,
  - $C_2(\forall y.F, t, x) = C_7(\forall y.F, x) \bigvee (C_3(t, y) \,\&\!\!\!\&\, C_2(F, t, x))$.
- For C_5:
  - $C_5(\neg F, x, y) = C_5(F, x, y)$,
  - $C_5(F \to G, x) = C_5(F, x, y) \,\&\!\!\!\&\, C_5(G, x, y)$,
  - $C_5(\forall z.F, x, y) = (C_4(x, z) \,\&\!\!\!\&\, C_5(F, x, y)) \bigvee (\sim C_4(x, z) \,\&\!\!\!\&\, C_7(F, y))$.
- For C_8:
  - $C_8(\neg F, x, y) = C_8(F, x, y)$,
  - $C_8(F \to G, x) = C_8(F, x, y) \bigvee C_8(G, x, y)$,
  - $C_8(\forall z.F, x, y) = (C_4(x, z) \,\&\!\!\!\&\, C_8(F, x, y))$.
- Similarly for the other conditions.

It is important to observe that in all the conditions one can arrange the formula-structures in such a way that inside the conditions one has only atoms (this will follow from the conventions that we are going to make, when considered together).

Convention 3.6. Whenever we are considering, at the same time, different schemata, we will implicitly assume that they do not have common variables (this is just a useful technical feature that does not have any conceptual reason).

Definition 3.7. A *substitution* σ is a function that assigns: formula-variables to formula-structures, term-variables to term-structures, and variable-variables to variable-variables.

It is important to observe that if one applies a substitution σ to a schema one might be increasing the number of term-variables and variable-variables.

Definition 3.8. We define inductively the *provable schemata* by:

Basis case Every schema that is an axiom is a provable schema;
Induction step If

- $F[\varphi^0_0, \ldots, \varphi^0_{n_0}, t^0_0, \ldots, t^0_{n_1}, v^0_0, \ldots, v^0_{n_2}] \& \bigvee_{i \in I^0} \,\&\!\!\!\&_{j \in J^0_i} \sim^{k^{0,0}_j} C_{k^{0,1}_j}(A^0_i, t_{k^{0,2}_j}, v_{k^{0,3}_j})$, and
- $G[\varphi^1_0, \ldots, \varphi^1_{n_3}, t^1_0, \ldots, t^1_{n_4}, v^1_0, \ldots, v^1_{n_5}] \;\to\; H[\varphi^2_0, \ldots, \varphi^2_{n_6}, t^2_0, \ldots, t^2_{n_7}, v^2_0, \ldots, v^2_{n_8}] \& \bigvee_{i \in I^1} \,\&\!\!\!\&_{j \in J^1_i} \sim^{k^{1,0}_j} C_{k^{1,1}_j}(A^1_i, t_{k^{1,2}_j}, v_{k^{1,3}_j})$

are provable schemata and there is σ such that

$$F[\sigma(\varphi_0^0),\ldots,\sigma(\varphi_{n_0}^0),\sigma(t_0^0),\ldots,\sigma(t_{n_1}^0),\sigma(v_0^0),\ldots,\sigma(v_{n_2}^0)] = G[\sigma(\varphi_0^1),\ldots,\sigma(\varphi_{n_3}^1),\sigma(t_0^1),\ldots,(t_{n_4}^1),\sigma(v_0^1),\ldots,\sigma(v_{n_5}^1)],$$

one says that

$$H[\sigma(\varphi_0^2),\ldots,\sigma(\varphi_{n_4}^2),\sigma(t_0^2),\ldots,\sigma(t_{n_5}^2),\sigma(v_0^2),\ldots,\sigma(v_{n_8}^2)]\& \bigvee_{i\in I^0} \&_{j\in J_i^0} \sim^{k_j^{0,0}} C_{k_j^{0,1}}(\sigma(A_i^0),\sigma(t_{k_j^{0,2}}),\sigma(v_{k_j^{0,3}}))\& \bigvee_{i\in I^1} \&_{j\in J_i^1} \sim^{k_j^{1,0}} C_{k_j^{1,1}}(\sigma(A_i^1),\sigma(t_{k_j^{1,2}}),\sigma(v_{k_j^{1,3}})) \ \left(\& C\right)$$

is a provable schema; furthermore,

$$\forall v.F[\varphi_0^0,\ldots,\varphi_{n_0}^0,t_0^0,\ldots,t_{n_1}^0,v_0^0,\ldots,v_{n_2}^0]\& \bigvee_{i\in I^0} \&_{j\in J_i^0} \sim^{k_j^{0,0}} C_{k_j^{0,1}}(A_i^0,t_{k_j^{0,2}},v_{k_j^{0,3}}) \left(\& C\right)$$

is also a provable schema. C is a possibly added condition that arises from conventions, for instance from convention 3.4 by adding C_4 conditions.

A provable schema S is *provable in k steps* if in the construction of S as a provable schema were used, at most, k steps (we do not count the application of conventions as steps nor the axiom case as a step). A provable schema that is an axiom has skeleton equal to the number of the axiom; if a provable schema S has skeleton $\mathcal{S}$, then the corresponding provable schema obtained using the universal rule, $\forall v.S$, has skeleton $\mathrm{Gen}(\mathcal{S})$; if S_0 and S_1 are provable schemata that have skeletons $\mathcal{S}_0$ and $\mathcal{S}_1$, respectively, and S is a schemata obtained using the MP construction from S_0 and S_1, then S has skeleton $\mathrm{MP}(\mathcal{S}_0,\mathcal{S}_1)$.

Convention 3.9. The equality in the previous definition should be read as follows: there is a substitution σ such that, after applying the conventions to both formula-structures considered in the definition, one gets syntactical equality. For each way of applying the conventions and for a fixed substitution one might get new provable schemata (for each way one gets a new provable schema). Thus, a schema is provable if there are a substitution and several applications of the conventions that make the conditions of the definition (in particular the equality) hold. In practice, convention 3.4 will be applied in the following way: in a schema, either one has the same variable occurring in a quantifier and in a replacement, and then one eliminates the replacement; or one proceeds as the convention suggests and one adds a condition C_4 to differentiate the variables. This is assumed, for instance, in the provable schemata by considering both situations after the application of σ (this information then goes to the condition C). We will make the same assumption for the other conventions that we are going to establish. Whenever we consider a schema in the previous conditions, we are, in fact, considering all the schemata that are obtained using the previous procedure. In practice, we also allow that in the previous definition

more conditions are added. Furthermore, we will use the notion of the previous definition where σ represents the application of several substitutions and conventions.

We now pause to give some examples. Clearly, $(\varphi \to (\psi \to \mu)) \to ((\varphi \to \psi) \to (\varphi \to \mu))$ is a provable schema, since it is an axiom. We also have that $(\varphi \to \psi) \to (\varphi \to \varphi)$ is a provable schema, since it can be obtained from the schemata $(\varphi \to (\psi \to \mu)) \to ((\varphi \to \psi) \to (\varphi \to \mu))$ and $\varphi \to (\psi \to \varphi)$ (by considering the substitution such that $\sigma(\varphi) := \varphi$, $\sigma(\psi) := \psi$, and $\sigma(\mu) := \varphi$). Moreover, for the same reason, $(\varphi \to \varphi) \to (\varphi \to \varphi)$ (by considering the substitution such that $\sigma(\varphi) := \varphi$, $\sigma(\psi) := \varphi$, and $\sigma(\mu) := \varphi$) is a provable schema. It is a good exercise to check that $\varphi \to \varphi$ is a provable schema.

Definition 3.10. We say that Σ is a *concrete-substitution* for the schema

$$F[\varphi_0, \ldots, \varphi_{n_0}, t_0, \ldots, t_{n_1}, v_0, \ldots, v_{n_2}] \& \bigvee_{i \in I} \&_{j \in J_i} \sim^{k_j^0} C_{k_j^1}(A_i, t_{k_j^2}, v_{k_j^3})$$

if Σ assigns formula-variables to actual formulas, term-variables to actual terms, and variable-variables to actual variables, in such a way that

$$\bigvee_{i \in I} \&_{j \in J_i} \sim^{k_j^0} C_{k_j^1}(\Sigma(A_i), \Sigma(t_{k_j^2}), \Sigma(v_{k_j^3}))$$

is a true condition and such that, in any occurrence of $G^{y_0 \cdots y_n}_{s_0 \cdots s_n}$ in F (with G a term-structure or a formula-structure), no $\Sigma(y_i)$ occurs in any $\Sigma(s_j)$, and there are no $\Sigma(y_i)$ with different attributions. It also needs to be the case that in every occurrence of $G^{y_0 \cdots y_n}_{s_0 \cdots s_n}$ in the schema, each $\Sigma(s_i)$ is substitutable for $\Sigma(y_i)$ in $\Sigma(G)$ and that no $\Sigma(y_i)$ is $\Sigma(y_j)$ with $i \neq j$ (there are no repetitions of the same change). For a schema S, we use the notation $\Sigma(S)$ to denote the result of performing the substitution Σ in the schema S whenever the substitution satisfies the definition.

To respect the previous definition, when one applies convention 3.4, one should add the condition $C_7(\forall y.\varphi, x) \bigvee C_3(t, y)$ in the context of provable schemata.

Convention 3.11. We assume that $\left(t^{\vec{x}}_{\vec{s}}\right)^y_r = t^{\vec{x}\ y}_{\vec{s}\ r}$, and the analogue identity for formula-structures, if there are no other occurrences of the basis of the replacement, t, in the schema that is being considered (here we are assuming that no $\vec{x}$ is a y, the suitable changes should be applied for the other case and the respective conditions should be added in the presence of provable schemata). This offers no problem with the concrete-substitution interpretation because if one has a concrete-substitution that satisfies the left-hand-side of the equation, one can construct a concrete-substitution that satisfies the right-hand-side and vice-versa. For example, if we had $(t^x_s)^y_r$ with $\Sigma(x) := x_0$, $\Sigma(s) := S(x_1)$, $\Sigma(y) := x_1$, $\Sigma(r) := x_2$, then by considering a concrete-substitution Σ' such that $\Sigma'(t) := \Sigma(t)$, $\Sigma'(x) := x_0$, $\Sigma'(y) := x_1$, $\Sigma'(s) := S(x_2)$, and $\Sigma'(r) := x_2$,

we would get

$$\Sigma\left((t_s^x)_r^y\right) = \left(\Sigma(t)_{\Sigma(s)}^{\Sigma(x)}\right)_{\Sigma(r)}^{\Sigma(y)} = \left(\Sigma(t)_{S(x_1)}^{x_0}\right)_{x_2}^{x_1} = \Sigma(t)_{S(x_2)\ x_2}^{x_0 x_1}$$
$$= \Sigma'(t)_{S(x_2)\ x_2}^{x_0 x_1} = \Sigma'(t)_{\Sigma'(s)\ \Sigma'(r)}^{\Sigma'(x)\ \Sigma'(y)} = \Sigma'(t_s^{x\ y}_{\ r}).$$

This means that if there is a concrete-substitution of one member of the equality, then there is a concrete-substitution for the other member. This identity will hold only if none of the $\vec{x}$ occurs in r (one should add the suitable condition to express this fact).

Observe that one could also have in a schema, besides the considered term-structure, the term-structures $(t_z^x)_s^z$ and t_r^y. In that type of situations, we take x' a totally fresh variable (i.e. not occurring at all) and the first term-structure is replaced by $t_{ss}^{x'\ z}$ and the second is replaced by $t_r^{y\ x'}$ (the same for formula-structures), assuming that z is not x (one should add add the suitable conditions for that).

More generally, $\left(t_{\vec{r}}^{\vec{x}}\right)_{\vec{s}}^{\vec{y}}$ is replaced by $(t')_{\vec{r}_{\vec{s}}^{\vec{y}}\ \vec{s}}^{\vec{x'}\ \vec{y}}$, with $\vec{x'}$ all fresh and t', and in the other occurrences of t where $\vec{x}$ are not being changed (one should do a case analysis for this using conditions C_4 and one should add the fact that, for the new t, x does not occur in t), one places $(t')_{\vec{x}}^{\vec{x'}}$ and one proceeds in a similar fashion for the other cases; in the previous situation we need to assume that none $\vec{x}$ occurs in $\vec{y}$, one should also consider the case where some of the $\vec{x}$ are $\vec{y}$ and add the suitable conditions in the presence of provable schemata, which simplifies the analysis.

We assume this convention also for formula-structures. More precisely, $\left(\varphi_{\vec{r}}^{\vec{x}}\right)_{\vec{s}}^{\vec{y}}$ is replaced by $(\varphi')_{\vec{r}_{\vec{s}}^{\vec{y}}\ \vec{s}}^{\vec{x'}\ \vec{y}}$, with $\vec{x'}$ all fresh and φ' fresh, and in the other occurrences of φ where $\vec{x}$ are not being changed, one places $(\varphi')_{\vec{x}}^{\vec{x'}}$; for the previous replacement to work we need, in the context of provable schemata, to add to the conditions: $\vec{x'}$ is free in φ, $\vec{x}$ is substitutable for $\vec{x'}$ in φ, $\vec{r}$ is substitutable for $\vec{x'}$ in φ, and $\vec{x}$ is not free in φ (the justification for all this procedure is that one needs these conditions for the previous reasoning for terms to be applied for formulas, namely to be able, in the presence of a concrete-substitutions, to go back from the image of the replaced formula-structure to the image of the initial one). This means that, without loss of generality, we will assume that in the schemata all these reductions were already applied—in practice, this means that several case analyses ought to be done. As our interpretation of the schemata is obtained via concrete-substitutions, we do not have a problem, since everything fits the definition.

Convention 3.12. In every occurrence of x_t^y in a provable schemata, one should consider the cases where x is the same as y, that entails $x_t^y = t$, and the case opposite case, that entails $x_t^y = x$. These situations will give rise to several provable schemata that are originated from a single syntactical expression. For each situation, we should add accordingly the conditions C_4 or $\sim C_4$ (c.f. convention 3.9).

Lemma 3.13. *Consider formulas φ and ψ, and schemata*

- $S_0 := F[\varphi^0_0,\dots,\varphi^0_{n_0},t^0_0,\dots,t^0_{n_1},v^0_0,\dots,v^0_{n_2}]\&\bigvee_{i\in I^0}\&_{j\in J^0_i}\sim^{k^{0,0}_j}C_{k^{0,1}_j}(A^0_i,t_{k^{0,2}_j},v_{k^{0,3}_j})$,
- $S_1 := G[\varphi^1_0,\dots,\varphi^1_{n_3},t^1_0,\dots,t^1_{n_4},v^1_0,\dots,v^1_{n_5}]\to H[\varphi^2_0,\dots,\varphi^2_{n_6},t^2_0,\dots,t^2_{n_7},v^2_0,\dots,v^2_{n_8}]\&\bigvee_{i\in I^1}\&_{j\in J^1_i}\sim^{k^{1,0}_j}C_{k^{1,1}_j}(A^1_i,t_{k^{1,2}_j},v_{k^{1,3}_j})$.

If there are concrete-substitutions Σ_0 and Σ_1 such that $\Sigma_0(S_0)=\varphi$ and $\Sigma_1(S_1)=\varphi\to\psi$, then there are a substitution σ such that

$$F[\sigma(\varphi^0_0),\dots,\sigma(\varphi^0_{n_0}),\sigma(t^0_0),\dots,\sigma(t^0_{n_1}),\sigma(v^0_0),\dots,\sigma(v^0_{n_2})]=$$
$$G[\sigma(\varphi^1_0),\dots,\sigma(\varphi^1_{n_3}),\sigma(t^1_0),\dots,\sigma(t^1_{n_4}),\sigma(v^1_0),\dots,\sigma(v^1_{n_5})]$$

and a concrete-substitution Σ such that $\Sigma(S_2)=\psi$, where

$$S_2 = H[\sigma(\varphi^2_0),\dots,\sigma(\varphi^2_{n_6}),\sigma(t^2_0),\dots,\sigma(t^2_{n_7}),\sigma(v^2_0),\dots,\sigma(v^2_{n_8})]\&$$
$$\bigvee_{i\in I^0}\&_{j\in J^0_i}\sim^{k^{0,0}_j}C_{k^{0,1}_j}(\sigma(A^0_i),\sigma(t_{k^{0,2}_j}),\sigma(v_{k^{0,3}_j}))\&$$
$$\bigvee_{i\in I^1}\&_{j\in J^1_i}\sim^{k^{1,0}_j}C_{k^{1,1}_j}(\sigma(A^1_i),\sigma(t_{k^{1,2}_j}),\sigma(v_{k^{1,3}_j}))\ \left(\&\,C\right).$$

Proof. By hypothesis, $\Sigma_0(S_0)=\varphi$ and $\Sigma_1(S_1)=\varphi\to\psi$. This means that there are concrete (and not variable) formulas $\varphi^0_0,\dots,\varphi^0_{n_0},\varphi^1_0,\dots,\varphi^1_{n_2}$, concrete terms $t^0_0,\dots,t^0_{n_1},t^1_0,\dots,t^1_{n_3}$, and concrete variables $v^0_0,\dots,v^0_{n_2},v^1_0,\dots,v^1_{n_5}$ obeying the semantical translation of the syntactical conditions such that

$$\begin{aligned}\varphi =& F[\varphi^0_0,\dots,\varphi^0_{n_0},t^0_0,\dots,t^0_{n_1},v^0_0,\dots,v^0_{n_2}]\\ =& G[\varphi^1_0,\dots,\varphi^1_{n_2},t^1_0,\dots,t^1_{n_3},v^1_0,\dots,v^1_{n_5}].\end{aligned}$$

Hence, the outer layout of implication signs, negation signs, universal quantifier signs, and parenthesis in S_0 and in S_1 can be made the same. For instance, the formula-structures $\varphi\to(\psi\to\varphi)$ and $\varphi\to((\varphi\to\varphi)\to\varphi)$ can have the same outer layout of signs, but $\varphi\to\varphi$ and $(\varphi\to\varphi)\to\varphi$ cannot—we use this expression to say that they have a common outer structure of parenthesis, implication signs, negation signs, and universal quantifier sings (we will mention it as simply the *layout*). A layout is nothing but a sequence of symbols: parenthesis, implication signs, negation sings, and universal quantifier signs. For example, $(\forall())\to(\neg()\to())$ is a layout. We will say that a layout L is a *sub-layout* of L' if L is a subsequence of L'. We call *entry* to the content of a layout inside implications such that they do not contain further implications.

It is not hard to see that there are substitution σ_0 and σ_1, and concrete-substitutions Σ'_0 and Σ'_1, such that the layout of $\sigma_0(F)$ and $\sigma_1(G)$ is the same as the layout of φ and $\Sigma'_0(\sigma_0(S_0))=\varphi$ and $\Sigma'_1(\sigma_0(S_1))=\varphi\to\psi$. It is enough for the procedure that we are going to describe that the layout is the same with possible exception of the entries in it because the entries are going to be accounted in the procedure *per se*. For example, consider the layouts $(\neg\forall(\neg()))\to(()\to(\neg()))$ and $()\to(()\to(\neg()))$; although they are not the same, for the purposes of the procedure that we are going to

develop the differences do not matter because they occur only at the level of the entries (this will be accounted for in the procedure); furthermore, when we are considering entries we always consider the biggest one, for instance, in the first layout, although (), $\neg()$, $\forall(\neg())$, and $\neg\forall(\neg())$ are all entries of the layout that correspond to the same position, we will consider the biggest one, i.e. $\neg\forall(\neg())$. In all, without loss of generality, we may assume that F and G have the same layout. We assume that the changes of convention 3.11 were already made in such a way that one does not have compositions of replacements (from the information of the concrete-substitutions one can find the correct way to apply the convention). Using the concrete-substitutions, one can find the correct way to apply the convention 3.4 and add that information to a condition C, composed of several C_4 conditions. After this, one can still find concrete-substitutions such that $\Sigma'_0(\sigma_0(S_0)) = \varphi$ and $\Sigma'_1(\sigma_0(S_1)) = \varphi \to \psi$.

We do the following to construct a substitution σ and a concrete-substitution Σ (we are implicitly considering convention 3.6 and all the other conventions):

1. If x is a variable-variable that is mapped to $\Sigma_i(x)$, then x is assigned to a new fresh-variable, $\sigma(x)$, and $\Sigma(\sigma(x)) := \Sigma_i(x)$. At this stage, one should also identify the variables that are mapped to the same concrete variables and distinguish the variables that are mapped to different variables using conditions C_4 and $\sim C_4$.
2. One proceeds in a similar way with the term-variables.
3. Starting from the first entry of the (common) layout, if the content of any of the entries is a formula–variable, then one assigns that formula-variable to the content of the other entry and moves to the next entry (in the end we might need to make some adjustments to this step).
4. Suppose now that the contents of both entries are not formula-variables.
   4.1. Suppose that the contents are $X^0_0 \cdots X^0_{\ell_0}(\varphi_i)^{v^0_0 \cdots v^0_n}_{t^0_0 \cdots t^0_n}$, in S_0, and $X^1_0 \cdots X^1_{\ell_1}(\varphi_k)^{v^1_0 \cdots v^1_m}_{t^1_0 \cdots t^1_m}$, in S_1, where φ_i and φ_j are formula-variables (here $X^i_0 \cdots X^i_{\ell_i}$ are arrays of quantifiers and negation symbols).
      4.1.1. It might be needed to make a substitution in order to $\ell_0 = \ell_1$ (for the case where one has all the layout equal with possible exception of the entries in it). If $\ell_0 < \ell_1$, this is achieved by $\sigma(\varphi_i) := X^1_{\ell_0+1} \cdots X^1_{\ell_1}\varphi'$, with φ' fresh (it is possible that it is necessary to make adjustments to the variables occurring in the quantifiers so that none of them is $v^0_0 \cdots v^0_n$ and to add the suitable conditions). Using σ, one unifies the variables occurring in the same place in the quantifiers (their value is already fixed by stage 1).
      4.1.2. One then assigns for each component in the entry the corresponding image through the concrete-substitution, with the difference that the occurrences of actual variables in the formulas are replaced by occurrences of variable-variables. One does the same for the variables and term-structures in the replacements.
      4.1.3. Define the concrete-substitution accordingly.

4.1.4. Move to the next entry.

4.2. Suppose that the contents are $X_0^0 \cdots X_{\ell_0}^0 (r_0 = s_0)_{t_0^0 \cdots t_n^0}^{v_0^0 \cdots v_n^0}$, in S_0, and $X_0^1 \cdots X_{\ell_1}^1 (r_1 = s_1)_{t_0^1 \cdots t_m^1}^{v_0^1 \cdots v_m^1}$, in S_1, where r_0, r_1, s_0, s_1 are term-structures.

4.2.1. One proceeds as in 4.1 for the quantifier variables.

4.2.2. We can perform substitutions to both entries in such a way that they become exactly equal to the content of the actual formula, but with the difference that actual variables in the formula are represented by variable-variables (the same idea that was applied before). For example, if the contents are $x + t = y$ and $t' = t''$, and they are both mapped using the concrete-substitutions to $x + (z + S(0)) = 0$, then one considers $\sigma(t) := z + S(0)$, with z a variable-variable, $\sigma(t') := x + (z + S(0))$, and $\sigma(t'') := 0$.

4.2.3. One does the suitable adaptions for the replacements. For instance, if the content of an entry is $t_s^x = S(x) + 0$ and the concrete-substitution is defined for the entry by $\Sigma_i(t) := \Sigma(x) \times (0 + x_0)$ and $\Sigma_i(s) := x_3 \times S(0)$; then one defines for this entry $\sigma(t) := \sigma(x) \times (0 + y)$, and $\sigma(s) := z \times S(0)$, entailing that

$$\begin{aligned}\sigma(t_s^x = S(x) + 0) &= \left(\sigma(t)_{\sigma(s)}^{\sigma(x)} = S(\sigma(x)) + 0\right) \\ &= ((z \times S(0)) \times (0 + y) = S(\sigma(x)) + 0)\,.\end{aligned}$$

Observe that one might need to add several C_4 conditions.

4.2.4. Define accordingly the concrete-substitution for this situation by assigning the variable-variables to the corresponding concrete variables.

4.2.5. One can perform substitutions in such a way that the term-structures that appear are equal to their image through the concrete substitution where the occurrence of variables in the actual formula is replaced by an occurrence of variable-variables in the term-structures, as described before (all this can be done because we have the guarantee of the existence of the concrete-substitutions).

4.2.6. Define the concrete-substitution for this case by assigning the variable-variables to the actual variables that they represent accordingly.

4.2.7. With the previous construction, in particular we have

$$\sigma((r_0 = s_0)_{t_0^0 \cdots t_n^0}^{v_0^0 \cdots v_n^0}) = \sigma((r_1 = s_1)_{t_0^1 \cdots t_m^1}^{v_0^1 \cdots v_m^1}).$$

4.2.8. If any variable-variable or term-variable is already assigned, make the suitable adaptations (this is always possible and reduces to a case analysis).

4.2.9. Move to the next entry.

4.3. If the content of one entry is $X_0^0 \cdots X_{\ell_0}^0 (r_0 = s_0)_{t_0^0 \cdots t_n^0}^{v_0^0 \cdots v_n^0}$ and the content of the other entry is $X_0^1 \cdots X_{\ell_1}^1 (\varphi_k)_{t_0^1 \cdots t_m^1}^{v_0^1 \cdots v_m^1}$, one proceeds in a similar way by attributing, via σ, the formula-variable φ_k in such a way that in the end of the substitution one is left with a version of the actual formula where variables are replaced by variable-variables.

4.4. Move to the next entry.

5. Make the suitable changes in all the variables in order to have

$$\Sigma(H[\sigma(\varphi_0^2), \ldots, \sigma(\varphi_{n_6}^2), \sigma(t_0^2), \ldots, \sigma(t_{n_7}^2), \sigma(v_0^2), \ldots, \sigma(v_{n_8}^2)]) = \psi.$$

It is not hard to see that the constructed concrete-substitution obeys the conditions of the considered schemata and the added ones (while applying the conventions in a suitable way). □

We could have presented a shorter proof of the previous results, but we decided to exhibit this one because it contains important ideas that are going to be used in several contexts. We can now prove that, using concrete-substitutions, k-provability of formulas is, in a sense, the same as k-provability of schemata.

Theorem 3.14. *The two following statements are equivalent:*

C1 $T \vdash_{k\text{steps}} \varphi$*;*

C2 There are S *a provable schema in* k *steps and a concrete-substitution* Σ *such that* $\Sigma(S) = \varphi$*.*

Proof. By definition of provable schema in k steps, we have that **C2** implies **C1** (this can be more formally proven by induction on k).

Let us prove that **C1** implies **C2** by induction on k, the number of steps. Clearly, if φ is an axiom, then there are an axiom schema S and a concrete-substitution Σ such that $\Sigma(S) = \varphi$. Suppose, by induction hypothesis, that the result holds for k. Furthermore, assume that $T \vdash_{k+1\text{steps}} \varphi$. In the last steps of a proof of φ we either apply the rule MP or the rule Gen:

MP In this case, there is a formula ψ such that $T \vdash_{k_0 \text{ steps}} \psi \to \varphi$ and $T \vdash_{k_1\text{steps}} \psi$, with $k = k_0 + k_1$. By induction hypothesis, there are schemata satisfying the conditions of lemma 3.13. By the lemma, it follows that there is a provable schema H and a concrete-substitution Σ such that $\Sigma(H) = \varphi$. As each of the schemata used in the lemma is, by induction hypothesis, provable in k_0 and k_1 steps (respectively), it follows that H is a schema provable in $k + 1$ steps.

Gen In this case, φ is of the form $\forall x.\psi$. By hypothesis, $T \vdash_{k\text{steps}} \psi$. So, by induction hypothesis, there are a provable schema in k steps, H, and a concrete-substitution Σ such that $\Sigma(H) = \psi$. Consider the provable schema in $k + 1$ steps obtained from H via the universal schema from definition 3.8, let us call it H'. Clearly, one can make the suitable changes in such a way that $\Sigma(H') = \forall x.\psi = \varphi$.

The result follows by induction. □

Lemma 3.15. *Consider a formula φ, and schemata S_0 and S_1. Suppose that there are a concrete-substitution Σ and a substitution σ (that might include the application of several substitutions and conventions) such that $\Sigma(S_0) = \varphi$ and $\sigma(S_1) = S_0$. Then, there is a concrete-substitution Σ' such that $\Sigma'(S_1) = \varphi$.*

Proof. It is not hard to see that this follows from the considered definitions and from the fact that all the conventions are compatible with the concrete-substitution interpretation. □

3.2. Decidability of Schemata

The next result has a similar content to proposition 2.2 from [6].

Lemma 3.16. *Given a condition of the form $\&_{i\in I} \sim^{k_i} C_{k_j^1}(A_i, t_i, v_i)$, one can computationally decide if there is a concrete substitution Σ such that*

$$\Sigma\left(\&_{i\in I} \sim^{k_i} C_{k_j^1}(A_i, t_i, v_i)\right) = \&_{i\in I} \sim^{k_i} C_{k_j^1}(\Sigma(A_i), \Sigma(t_i), \Sigma(v_i))$$

is true; furthermore, witnesses can be found in affirmative case. The result still holds if the image of certain meta-variables are a priori *fixed under a concrete-substitution.*

Proof. The following idea is a procedure for the case where all the atoms that are not fixed are formula-variables and where the term-variables that are not fixed occur without replacements:

1. Start by considering enough variables to satisfy the occurrences of the conditions C_4 and $\sim C_4$.
2. Then, consider every formula as being equal to $0 = 0$ and every term as being equal to 0.
3. After that, focus on C_0 conditions. For every occurrence of $C_0(\varphi, x)$, make the attribution $\varphi := (\varphi \wedge x = x)$ (if x is not yet free in φ). Proceed in a similar way to $\sim C_7$ (for this condition we just need one free occurrence). For the occurrences of $C_1(\varphi, x)$ make $\varphi := (\varphi \wedge \forall x.x = x)$, in particular if $\sim C_7(\varphi, x)$ and also $C_1(\varphi, x)$, then attribute instead $\varphi := (\varphi \wedge x = x) \wedge (\forall x.x = x)$.
4. Similarly for $\sim C_3$, for every occurrence of $\sim C_3(t, x)$, consider $t := (t{+}x)$.
5. Each time $\sim C_2(\varphi, t, x)$ occurs, (one should always compare what one is doing here with the C_5 and C_2 conditions for the same formula-variables and variable-variables) consider y a fresh variable, and attribute $t := (t + y)$ and $\varphi := (\varphi \wedge \forall y.x = 0)$.
6. For each occurrence of $\sim C_5(\varphi, x, y)$, take $\varphi := (\varphi \wedge \forall x.y = 0)$. For $C_8(\varphi, x, y)$ we might need to consider $\varphi := (\varphi \wedge y = 0)$, in particular if $\sim C_5(\varphi, x, y)$ and $C_8(\varphi, x, y)$ occur, then attribute $\varphi := ((\varphi \wedge \forall x.y = 0) \wedge y = 0)$. If $\sim C_8(\varphi, x, y)$ occurs, one should consider two cases:
   - 6.1. y does not occur free in φ, i.e. $C_7(\varphi, y)$;
   - 6.2. Or $\sim C_7(\varphi, y)$ and in all free occurrences of y in φ they occur under the scope of a $\forall x$ quantifier. For this situation one acts in a similar way to the one described for $\sim C_5(\varphi, x, y)$ in all free occurrences of y (one places $\forall x$ in all free occurrences of y in φ).

7. Now test, for the considered attributions, if the occurrences of C_0, C_1, C_2, C_3, C_5, $\sim C_6$, C_7, C_8, and $\sim C_8$ are satisfied. In negative case (one should consider all possible situations), reject.
8. If $C_6(\varphi, x)$ is in the expression, test if the condition is already satisfied for the considered attributions. If not, then take $\varphi := (\varphi \wedge \forall x.x = 0)$ and test the conditions again.

If any variable, term, or formula is already fixed, the previous analyses remain valid (some adaptations are needed, for instance in the beginning of the algorithm)—this simplifies the algorithm to a case analysis. Furthermore, it is not hard to adapt it for replacements and for more complex term-structures (see the identities below). Suppose now that we have atoms of the form $(r = s)^{\vec{x}}_{\vec{t}}$. Then, we do the following:

1. We start by listing all the possible ways to apply the $\vec{x}$ to $r = s$, i.e. all the ways to apply the replacements (including the way in which the variables $\vec{x}$ do not occur in $r = s$). Each possibility will give rise to a separate analysis.
2. We proceed as in step 1 until no further replacements are applicable to term-structures.
3. In the previous step, one is left with several occurrences of $t^{\vec{y}}_{\vec{t'}}$.
4. One proceeds in a way similar to the previous algorithm (if possible, i.e. if no contradiction was reached). Observe that such an analysis is simplified, since, for instance, being free in the considered formula reduces to occurring in the formula (because there are no quantifiers).

One can also adapt accordingly the idea of the initial procedure. One should also have in mind the following identities concerning t^x_s:

- For C_2, with z a totally fresh variable,

$$C_2(\varphi, t^y_s, x) = \Big(C_3(t,y) \,\&\, C_2(\varphi, t, x)\Big) \vee \Big(\sim C_3(t,y) \,\&\, C_2(\varphi, s, x) \,\&\, C_2(\varphi, t^y_z, x)\Big).$$

- For $\sim C_3$,

$$\sim C_3(t^y_s, x) = \Big(C_3(t,y) \,\&\, \sim C_3(t,x)\Big) \vee \Big(\sim C_3(t,y) \,\&\, C_4(x,y) \,\&\, \Big(\sim C_3(t,x) \vee \sim C_3(s,x)\Big)\Big).$$

The case where we have atoms of the form $\varphi^{\vec{x}}_{\vec{t}}$ is a particular case of the previous analysis when one has in mind the following identities (that can be extended for more complex replacements):

- For C_0,

$$C_0(\varphi^y_t, x) = \Big(C_7(\varphi, y) \,\&\, C_0(\varphi, x)\Big) \vee \Big(\sim C_7(\varphi, y) \,\&\, C_2(\varphi, t, y) \,\&\, \Big(\Big(C_4(x,y) \,\&\, C_0(\varphi, x) \,\&\, C_3(t,x)\Big) \vee \Big(C_4(x,y) \,\&\, C_0(\varphi, x) \,\&\, \sim C_3(t,x) \,\&\, C_5(\varphi, x, y)\Big)\Big)\Big).$$

- For C_2,

$$C_2(\varphi_s^y, t, x) = \Big(C_7(\varphi, y) \,\&\!\!\!\&\, C_2(\varphi, t, x)\Big) \bigvee \Big(\sim C_7(\varphi, y) \,\&\!\!\!\&\, C_2(\varphi, s, y) \,\&\!\!\!\&\, \Big(\sim C_4(x, y) \bigvee \Big(C_4(x, y) \,\&\!\!\!\&\, C_3(s, x) \,\&\!\!\!\&\, C_2(\varphi, t, x)\Big) \bigvee \Big(C_4(x, y) \,\&\!\!\!\&\, \sim C_3(s, x) \,\&\!\!\!\&\, C_2(\varphi, t, x) \,\&\!\!\!\&\, C_2(\varphi, t, y)\Big)\Big)\Big).$$

- For C_5,

$$C_5(\varphi_s^x, z, y) = \Big(C_7(\varphi, x) \,\&\!\!\!\&\, C_5(\varphi, z, y)\Big) \bigvee \Big(\sim C_7(\varphi, x) \,\&\!\!\!\&\, C_2(\varphi, s, x) \,\&\!\!\!\&\, \Big(\sim C_4(x, y) \bigvee \Big(C_4(x, y) \,\&\!\!\!\&\, C_3(s, y) \,\&\!\!\!\&\, C_5(\varphi, z, y)\Big) \bigvee \Big(C_4(x, y) \,\&\!\!\!\&\, \sim C_3(s, y) \,\&\!\!\!\&\, C_5(\varphi, z, y) \,\&\!\!\!\&\, C_5(\varphi, z, x)\Big)\Big)\Big).$$

- For $\sim C_7$,

$$\sim C_7(\varphi_t^y, x) = \Big(C_7(\varphi, y) \,\&\!\!\!\&\, \sim C_7(\varphi, x)\Big) \bigvee \Big(\sim C_7(\varphi, y) \,\&\!\!\!\&\, C_2(\varphi, t, y) \,\&\!\!\!\&\, \Big(C_4(x, y) \,\&\!\!\!\&\, \Big(\sim C_7(\varphi, x) \bigvee \Big(\sim C_3(t, x) \,\&\!\!\!\&\, C_8(\varphi, x, y)\Big)\Big)\Big)\Big).$$

- For C_8,

$$C_8(\varphi_s^x, z, y) = \Big(C_7(\varphi, x) \,\&\!\!\!\&\, C_8(\varphi, z, y)\Big) \bigvee \Big(\sim C_7(\varphi, x) \,\&\!\!\!\&\, C_2(\varphi, s, x) \,\&\!\!\!\&\, \Big(C_4(x, y) \,\&\!\!\!\&\, C_3(s, y) \,\&\!\!\!\&\, C_8(\varphi, z, y)\Big) \bigvee \Big(C_4(x, y) \,\&\!\!\!\&\, \sim C_3(s, y) \,\&\!\!\!\&\, \Big(C_8(\varphi, z, y) \bigvee C_8(\varphi, z, x)\Big)\Big)\Big).$$

The result follows by this observation and the previous analysis (and by a version of disjunctive normal form for meta-connectives). □

Theorem 3.17. *Given a schema S and a formula φ, it is decidable whether there is a concrete-substitution Σ such that $\Sigma(S) = \varphi$.*

Proof. Consider φ a formula and S a schema to which all the conventions were already applied. If the condition in S is *a priori* false, i.e. if using the rules of convention 3.5 one can obtain $\bot$, then there is no concrete-substitution. Suppose now that S is of the form

$$F[\varphi_0, \ldots, \varphi_{n_0}, t_0, \ldots, t_{n_1}, v_0, \ldots, v_{n_2}] \& \bigvee_{i \in I} \&\!\!\!\&_{j \in J_i} \sim^{k_j^0} C_{k_j^1}(A_i, t_{k_j^2}, v_{k_j^3}),$$

where the condition is not *a priori* false. It is not hard to see that one can computationally decide whether there is a substitution σ such that $\sigma(F)$ and φ have the same layout (the structure of parenthesis, implication signs, negation signs, and universal quantifier signs described before). The idea is the following:

1. If F and φ have different types, then reject (by different types we mean that, for instance, one is a negation and the other is an implication).
2. If they have the same type, then one goes to the layout to the left of both outermost implication signs.

3. If the layout is the same, then one goes to the right and does the same move throughout the process.
4. If one reaches an incompatibility—for example one implication sign versus one negation sign or universal quantification sign—one rejects.
5. One does the same for negations and universal quantifications.
6. Using attributions to the formula-variables, one locally matches the layout of φ.
7. One proceeds by going to the inside the respective layouts until one reaches either a rejection or an equal layout (this procedure must eventually stop because the layout of the formula, just like the layout of a formula-structure, is finite).

In this briefly described way, one is not creating unnecessary changes in F, it is minimal in that sense. If F cannot be changed to have the same layout as φ, then there is no concrete-substitution (all this is decidable). Suppose that there is such a substitution σ that makes the layout the same. Consider σ in the mentioned minimal conditions, $F' := \sigma(F)$, and $S' := \sigma(S)$. Assume that convention 3.11 was already applied to the schema, as well as convention 3.4 (this will yield a finite number of schemata to which one should do the analysis that follows). Having in mind that F' and φ have the same layout, it is decidable whether there is Σ such that $\Sigma(F') = \varphi$. The idea is the following:

1. Just like what was done in 4 of the proof of lemma 3.13, let us consider all the entries in the layout of φ (that is the same layout of F').
2. Now, consider the finite list of variables occurring in φ under the scope of universal quantifiers and the finite list of terms occurring in φ.
3. One checks if there is any incompatibility between the array of quantifiers and negation signs of each entry of F' and of φ.
4. If there is, one rejects.
5. Otherwise, one defines Σ for the variable-variables occurring in the quantifiers accordingly.
6. Starting from the first entry and vanishing all entries, one takes for each entry (where the corresponding entry in the formula-structure has a replacement) the respective formula where the occurrences of some terms are replaced by fresh variables—ones should analyse all possibilities. Then, one checks if it is possible to assign formulas, terms, and variables to the entries of F' in such a way that one obtains φ and they satisfy the (decidable) condition of the schema F'. One does this for all entries. For single occurrences of term-variables without replacements, one simply assigns the corresponding term that occurs in the actual term.
7. More precisely, one tests all possible substitutions by considering the formulas and the terms where the occurrences of some terms are replaced by fresh variables (one should vanish over all possibilities), one sees what the substituting term should look like, and one tests all the (finite number of) possibilities by making the fresh variables equal to some of the variables of the considered replacement. For each test one sees if the conditions of the schema are satisfied. For example, if one has the formula

$(x+0)+z=y$ in the actual formula and the corresponding formula-structure $(t_0)^{x_0}_{s_0}=x_1$ in the schema, then one should assign x_1 to the variable y and, as in t_0 we are considering one replacement, one should consider the following possibilities for t_0:

- t_0 as being x_0, with s_0 being $(x+0)+z$;
- t_0 as being $(x+0)+x_0$, where s_0 is z;
- t_0 as being x_0+z, where s_0 is $x+0$;
- t_0 as being $(x_0+0)+z$, with s_0 being x;
- t_0 as being $(x+x_0)+z$, with s_0 being 0;
- t_0 as being $(x+0)+z$ with x_0 not occurring in t_0, where x_0 and s_0 are "arbitrary" in what t_0 is concerned; in practise this means that either they are assigned in the next entries, or they are to be considered as not assigned, which means that one has just to further study them if they appear in the conditions—see the proof of lemma 3.16 for a more detailed account.

More generally, consider the case where a term-structure in the schema needs to be equal, under a concrete-substitution, to a certain term. Then, one needs to satisfy an equality similar to

$$\Sigma\big(((t_0)^{\vec{x_0}}_{\vec{s_0}}+S(t_1))\times t_2\big)\\ =((S(x+y)+S(S(0)))+S(S(0)+(x\times z)))\times((x\times y)+0),$$

i.e.

$$\Big(\Sigma(t_0)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})}+S(\Sigma(t_1))\Big)\times\Sigma(t_2)\\ =((S(x+y)+S(S(0)))+S(S(0)+(x\times z)))\times((x\times y)+0).$$

This entails that

$$\begin{cases}\Sigma(t_0)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})} &= (S(x+y)+S(S(0)))\\ S(\Sigma(t_1)) &= S(S(0)+(x\times z))\\ \Sigma(t_2) &= ((x\times y)+0)\end{cases}$$

Thus, the image of t_2 under the concrete-substitution that one wants to construct is fixed, as well as the image of t_1 (if they were already fixed one should test if a contradiction is obtained). This means that, for the desired equality, it only remains to be analysed the equality $\Sigma(t_0)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})}=(S(x+y)+S(S(0)))$. For this equality, one makes a (finite) case analysis as before by means of fresh variables (this will yield a similar analysis for the term-structures $\vec{s_0}$). If $\Sigma(t_0)$ is already assigned to, for instance, $\Sigma(t_0)=S(\Sigma(x_0)+y)+S(\Sigma(x_i))$, then one substitutes the already attributed t_0 in the desired equality, namely

$$(S(\Sigma(x_0)+y)+S(\Sigma(x_i)))^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})}=S(x+y)+S(S(0)),$$

which entails that

$$S\Big(\Sigma(x_0)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})}+y^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})}\Big)+S\Big(\Sigma(x_i)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})}\Big)=S(x+y)+S(S(0)),$$

and so

$$\begin{cases} \Sigma(x_0)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})} & = x \\ y^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})} & = y \\ \Sigma(x_i)^{\Sigma(\vec{x_0})}_{\Sigma(\vec{s_0})} & = S(0); \end{cases}$$

something that can, once again, be easily solved through a case analysis. Throughout the process one should add the suitable conditions on the variables. After that, one substitutes the new information about the variables and one sees if any contradiction is reached.

8. We do the previous procedure for each entry of the layout and also for atoms—in fact, for formula-variables with replacements the fresh variables analysis remains valid: if one has $\varphi^{\vec{x}}_{\vec{s}}$, then one considers fresh term-variables t_0 and t_1, one considers φ as being $(t_0 = t_1)$, and one proceeds the analysis as before. For each case in the analysis of a given entry, one should consider all the sub-cases in the other entries for the choices that were made—this gives rise to a tree of possible cases; moving from one entry to another, either a new case analysis is created, or one reaches a contradiction, which, by its turn, forces the considered case in the already established case analysis to change (in particular this yields that the algorithm as a whole halts). One can computationally check if there are any incompatibilities at any stage; if any incompatibility is detected, one should consider another case in the analysis, if one reaches an incompatibility with all cases, it means that there is no concrete-substitution in the desired conditions. Observe that, for each entry, there is a finite number of ways to do the considered procedure, which entails that in the whole schema there is also a finite number of ways to consider all the possible cases.
9. In the end of a case analysis, one should test to see whether the conditions of the schema are satisfied. This is achieved using lemma 3.16. One should also test if Σ can be made in such a way that satisfies the definition 3.10.
10. If the previous steps are not possible, one should reject.

From lemma 3.15, if there is a concrete-substitution Σ' such that $\Sigma'(S') = \varphi$, then there is a concrete substitution such that $\Sigma(S) = \varphi$. All the mentioned construction yields that one can decide whether there is a concrete-substitution Σ such that $\Sigma(S) = \varphi$. □

It is important to observe that in the former algorithm it is not fundamental that the convention 3.11 is applied: if one has, for instance, the term–structure $(t^x_r)^y_s$ and one wants it to be equal to a certain term, then one proceeds by forcing T^y_s to be equal to that term using the considered analysis, and then one imposes $\Sigma(t^x_r) = \Sigma(T)$ and makes a similar analysis for that fact; this means that if convention 3.11 was not applied, then one has to do several times the creation of the case analysis of stage 7 from the previous algorithm. We opted to firstly apply the convention because it simplifies the analysis and avoids having chained replacements.

3.3. Decidable of Some Proof-skeletons and k-Provability

As mentioned in the introduction, the decidability of k-provability for PA with the usual instantiation schema is an open problem and the proof-skeleton problem is in general undecidable for that version of PA. Nevertheless, we will characterise some values of k for which k-provability is decidable and some proof-skeletons for which the corresponding proof-skeleton problem is decidable.

Definition 3.18. We say that a proof-skeleton $\mathcal{S}$ is *stable (for T)* if there is a finite list of provable-schemata $\mathcal{L}_{\mathcal{S}}$ such that:

Stability A formula φ has a proof whose skeleton is $\mathcal{S}$ if, and only if, there are a schema S in $\mathcal{L}_{\mathcal{S}}$ and a concrete-substitution Σ that satisfy $\Sigma(S) = \varphi$.

We say that a number k is *stable (for T)* if all proof-skeletons with length at most k are stable.

For stable proof-skeleton, the corresponding proof-skeleton problem is decidable, as the next result confirms.

Theorem 3.19. *If a proof-skeleton $\mathcal{S}$ is stable, then, for any formula φ, it is decidable whether φ has a proof whose skeleton is $\mathcal{S}$.*

Proof. By definition, φ has a proof whose skeleton is $\mathcal{S}$ if, and only if, there are a schema S in $\mathcal{L}_{\mathcal{S}}$ and a concrete-substitution Σ that satisfy $\Sigma(S) = \varphi$. The decidability follows from the fact that $\mathcal{L}_{\mathcal{S}}$ is finite and from theorem 3.17 (one tests computationally for each element of the finite list $\mathcal{L}_{\mathcal{S}}$). □

If k is stable, then the respective k-provability is decidable, as the next result confirms.

Theorem 3.20. *If k is a stable number, then it is decidable whether $T \vdash_{ksteps} \varphi$ or not.*

Proof. The result follows from the fact that for each k there is a finite number of proof-skeletons with length k and from theorem 3.19. □

Theorem 3.21. *There is a maximum k stable number for PA$'$.*

Proof. Suppose that there is no maximum k stable number for PA$'$. As the fact that k is a stable number implies that, for all $s \leq k$, s is a stable number; it follows that all numbers are stable for PA$'$. Thus, all proof-skeletons are stable. In particular, the proof-skeletons from the proof of theorem 6.1 from [6]—which is similar to the proof of 5.1 from [8]—are stable; from the previous theorem, it follows that it is decidable whether a formula has a proof whose skeleton is the skeletons from the proof of theorem 6.1 from [6], which contradicts that very theorem (when one adapts the proof to the theory PA$'$). □

Although theorem 3.20 presents a characterisation of some proof-skeletons that have the respective proof-skeleton problem decidable, we have no information on how the lists were created; furthermore, so far we have no general way to generate (some of the) stable proof-skeletons. We now proceed to develop a general way to generate stable proof-skeletons.

From [8], we know that second-order unification is in general undecidable. In the proof presented in that paper, the main idea was to represent, inside the context of second-order unification, numerals, addition, and multiplication; something that is delivered by, c.f. [8], equations that include:

$$\begin{aligned} &s(\tau) = \tau(a/s(a)),\\ &\quad \tau(a/\sigma_1, b/s(b), c/a \circ (b \circ c)) = \sigma_2 \circ (\sigma_3 \circ \tau). \end{aligned}$$

In the previous equations, we followed the notation of the considered paper, where s denotes a unary function-symbol, $\circ$ denotes a binary one, and σ_2, σ_3, τ term-variables. It is important to observe that, in the previous equations, one has in the two members of the equality sign occurrences of the same term-variable and occurrences of variables being replaced by something where that very variables occur. While this last feature can be avoided, using convention 3.11, the first feature will be something that we will not allow in the algorithms that we are going to develop; for example, $S(t^{x_0}_{x_1}) = t^{x_0}_{S(x_1)}$ has an infinity of solutions, namely $t = S^n(x_0)$, that cannot be written in a closed form like $t = \mathcal{E}$, where in $\mathcal{E}$ we do not have occurrences of natural numbers variables like $S^n(x_0)$. We will create an algorithm that will be able to solve *some* systems of the form

$$\begin{cases} r_0 &= s_0\\ &\vdots\\ r_n &= s_n, \end{cases}$$

where $r_0, \ldots, r_n, s_0, \ldots, s_n$ are term-structures (we will assume that the equations in the left do not have common variables with the equations on the right): by a solution we mean a common substitution to the left and the right side such that one achieves the equality between them maybe after the application of several of the conventions (in fact, we might achieve the equality after the application of several substitutions and several conventions)—for each substitution there might be several ways to apply the conventions that yield several solutions. The main move that we are going to make is to avoid the mentioned occurrences of equal term-variables while the algorithm is running, this offers no problem since we will not develop a general algorithm and since the reasoning of [8] cannot be applied—we force the algorithm not to have the needed conditions for the proof, and thus avoid the undecidability; we do this with the cost that the algorithm will reject or not halt systems that indeed have a solution.

Algorithm 3.22. We will describe an algorithm that, for certain cases, sees whether there is a substitution σ such that

$$\begin{cases} \sigma(r_0) &= \sigma(s_0)\\ &\vdots\\ \sigma(r_n) &= \sigma(s_n), \end{cases}$$

where $r_0, \ldots, r_n, s_0, \ldots, s_n$ are term-structures. Furthermore, such a σ can be found (for some cases) without the introduction of unnecessary complexity. Following convention 3.6, we assume that the r_i's and s_j's do not have common meta-variables. Furthermore, we assume that convention 3.11 was already applied. The algorithm is as follows:

1. Starting from $i = 0$, do the following to construct a list of equations:
   1.1. If r_i and s_i are term-structures without term-variables, then see if it is possible to identify the variable-variables in such a way that $\sigma(r_i) = \sigma(s_i)$ and add this fact to the list; if it is not possible, then reject.
   1.2. If r_i or s_i is of the form $t^{\vec{x}}_{\vec{s}}$, then add to the list of equations $t^{\vec{x}}_{\vec{s}} = s_i$, when $r_i = t^{\vec{x}}_{\vec{s}}$, and $r_i = t^{\vec{x}}_{\vec{s}}$, for the other case.
   1.3. If r_i and s_i are both term-structures with an outermost occurrence of a function-symbol, then test whether the function-symbol is the same.
   1.4. If it is not, then reject.
   1.5. If $r_i = S(t_0)$ and $r_1 = S(t_1)$, with t_0 and t_1 term-structures, then apply the previous procedure to t_0 and t_1 and add $t_0 = t_1$ to the list. Do the same for $+$ and $\times$. This means that for each pair of terms r_i and s_i, one should see if $r_i = t_0 + t_1$ and $s_i = t_2 \times t_3$, or $r_i = t_2 \times t_3$ and $s_i = t_0 + t_1$. If it occurs, then reject; and for each pair of terms r_i and s_i with $r_i = t_0 \circ t_1$ and $s_i = t_2 \circ t_3$, do the previous procedure for t_0 with t_2, and t_1 with t_3, where $\circ$ is $+$ or $\times$, and add $t_0 = t_2$ and $t_1 = t_3$ to the list.
   1.6. Do the previous procedure until no further reductions are possible (this must stop after a finite number of steps). Thus, one should apply the procedure until one reaches either a rejection or has analysed all possible cases.
   1.7. Increment i until all the values for i were considered.
2. Let us now assume that a list was build without reaching a rejecting state.
3. As mentioned in 1.1, one has to make the suitable variable identification (for example using fresh variable-variables). For instance, if one has in the list an equation of the form

$$(\cdots + x) \times \cdots = (\cdots + y) \times \cdots ,$$

   where x and y occur in the same place of the layout of the function-symbols, then one has to assign x and y to a new fresh common variable-variable. With this we get, we get the true equality

$$(\cdots + \sigma(x)) \times \cdots = (\cdots + \sigma(y)) \times \cdots .$$

   If any of them was already assigned, assign all the variables previously assigned to this new common fresh variable.
4. Proceed in a similar way with the occurrence of term-variables that do not occur under the scope of a replacement. This means that if one has

an equation of the form

$$((t_0 \times t_1) + S(0)) \times \cdots = ((t_2 \times y) + t_3) \times \cdots ,$$

then $\sigma(t_0) = \sigma(t_2) = t$, with t fresh, $\sigma(t_1) = \sigma(y)$, and $\sigma(t_3) = S(0)$.

5. Let us now briefly describe the most complex case.
6. Suppose we have in the list equations of the following form (observe that the analysis of the list can be reduced to the analysis of the next system), where we are implicitly considering in the left-side the r-part, and in the right-side the s-part:

$$n \text{ equations} \begin{cases} (t_0)^{\vec{x}_0}_{\vec{s}_0} = (\cdots + (x + 0)) \times (\cdots (y + x) \times 0) \\ (t_0)^{\vec{x}_2}_{\vec{s}_2} = ((\cdots \times (x \times 0))) \times t_2 \\ (\cdots \times (x + S(0))) + 0 = (t_4)^{\vec{x}_5}_{\vec{s}_5} \\ \vdots \\ (t_5)^{\vec{x}_6}_{\vec{s}_6} = (\cdots + (y \times 0)) \times S(S(0)). \end{cases}$$

Do the following (the next procedure is a kind of co-recursion because one should apply the whole procedure to smaller parts of the very same procedure):

- 6.1. Firstly, substitute the already changed variable-variables and term-variables in the equations (one should do this step at every stage of the algorithm).
- 6.2. If at any stage one obtains an equality where in both members one has an occurrence of $t^{\vec{x}}_{\vec{s}}$, one should output **problem**.
- 6.3. If at any stage one obtains a non-trivial equality (i.e. not yet syntactically satisfied) where in both members one has an occurrence of the same term-variable one should output **problem**.
- 6.4. Starting from $i = 1$ (up to n), solve, if possible, the first equation in the following way:
  - 6.4.1 Consider the term-structure that corresponds to the right-side (respectively left-side) of the equation where some occurrences of term-structures in the considered equation were replaced by fresh variables (analyse all the finitely many possibilities for this).
  - 6.4.2 Analyse all the possibilities to identify the fresh variables with the variables-variables $\sigma(\vec{x}_0)$ and see what the term-structures $\sigma(\vec{s}_0)$ ought to be—keep in mind that all the conventions ought to be satisfied (this fact can be verified in a decidable way). This idea was also used at stage 7 in the second procedure of the proof of theorem 3.17. If one has one of the variables $\sigma(\vec{x}_0)$ occurring in the opposite side, one should reject, since that is an impossibility. In the context of (provable) schemata, one should add condition C_4 for the variables that are being considered as being different throughout the procedure (the analogous situation for the other conventions). One should always substitute the already found solutions in the new equations.

6.4.3 In $\sigma(\vec{s_0})$ there might occur complex term-structure that would require an analysis similar to the one that we are considering (for example, s_0 could be $t^x_{t_0+t_1}$) and would yield a new system to be solved (to create the system we replicate the previous steps for the creation of the list and the fresh variables analysis). Without loss of generality, we might proceed our analysis because the algorithm will account for all the cases due to its co-recursive nature. We allow the occurrence of equations of the form

$$(t_0)^{\vec{x_0}}_{\vec{s_0}} = ((x + y) \times 0) + (S(t_1) + S(0)).$$

6.4.4. For instance, in the case analysis of the previous stage, one might have $t^{x_0}_{r^{x_1}_s} = (x+y) \times z$. One should consider the cases as in 7 in the second procedure of the proof of lemma 3.17 and as mentioned before. One of the cases is $\sigma(t) = \sigma(x_0) \times \sigma(z)$ with $\sigma(r^{x_1}_s) = \sigma(x + y)$. Then, for the previous equality, one should also carry a similar analysis. For example, $\sigma(r) = \sigma(x_1)+\sigma(y)$, with $\sigma(s) = \sigma(x)$. In all, for the considered case, we get

$$\begin{aligned}\sigma(t^{x_0}_{r^{x_1}_s}) &= \sigma(t)^{\sigma(x_0)}_{\sigma(r)^{\sigma(x_1)}_{\sigma(s)}} = (\sigma(x_0)\times\sigma(z))^{\sigma(x_0)}_{\sigma(r)^{\sigma(x_1)}_{\sigma(s)}} = \sigma(r)^{\sigma(x_1)}_{\sigma(s)} \times \sigma(z)\\ &= (\sigma(x_1) + \sigma(y))^{\sigma(x_1)}_{\sigma(s)} \times \sigma(z) = (\sigma(x) + \sigma(y)) \times \sigma(z)\\ &= \sigma((x + y) \times z).\end{aligned}$$

Observe that, in the previous analysis, we considered the variables as being different when the replacements are applied because they are fresh, they do not occur at all. Furthermore, observe that, for the considered case, $\sigma(z)$ cannot be identified with $\sigma(x_0)$ just like $\sigma(x_1)$ cannot be identified with $\sigma(y)$ (otherwise we would get a contradiction, in particular if we apply concrete-substitutions)—one should add the suitable conditions to express this fact. This means that the replacements that we are considering here act only on the fresh variables, without the possibility of overlapping the left with the right side, as mentioned.

6.4.5. If a term-variable occurs in the other side of the equation, one should also consider, together with the fresh variables construction, the case where for that entry one places a fresh term-variable in the term-structure that one is creating. For example, if one has

$$(t_0)^x_s = (S(0) + y) + t_1,$$

one of the cases that one should analyse is the case where $\sigma(t_0) = (\sigma(x) + \sigma(y)) + t$, with t fresh, which entails that $\sigma(t_1) = t^{\sigma(x)}_{\sigma(s)}$. This serves to cover the possibility, under the interpretation of the equality via concrete-substitutions, that the

concrete term that results from t_1 was placed using the considered replacement to some other term in that very position in the function-symbols layout.

6.4.6. Take note of the possible ones—the admissible possibilities form a tree, in the sense that for one case one might have to analyse a great variety of sub-cases. Unlike the proof of theorem 3.17 where the second algorithm only asks for a successful path in the tree, in this algorithm we want to study all admissible possibilities, since we want to find all the solutions to the system.

6.4.7. If one reaches an impossibility in the case analysis one should reject that case and consider the other remaining cases.

6.5. If, at any stage, there are no solutions, one should reject.

6.6. Suppose that the system was solved, if possible, for $i < n$. Then, do the following:

6.6.1. Consider the $(i+1)$th equation, say $(t_5)^{\vec{x}_6}_{\vec{s}_6} = (\cdots + (y \times 0)) \times S(S(0))$.

6.6.2. Substitute the already found term-structures and, if necessary, create new systems for the equalities that emerge.

6.6.3. In particular, if t_5 was already found, then substitute it for the found solution, make a case analysis for the applications of the replacements, and solve the considered equation. If t_5 was not yet assigned, then solve the equation using the fresh variables analysis and with 6.2.5.

6.6.4. Substitute in the previously obtained solutions the new ones and if necessary solve the new equations that emerge.

7. Output all the found solutions.

We say that a system is *successful* if it halts and is not rejected as a whole (nevertheless we allow that in the case analysis some cases are rejected), and if in the case analysis no **problem** situation appeared. What we described is not a complete description of the algorithm, but it has the main ideas that are necessary to develop the much more complex and involved complete description of the algorithm. We considered particular cases in the algorithm to emphasise the main ideas. For instance, in the hard case that we presented—the one with a new system—, a more general approach is needed to write the complete algorithm.

Let us give an example of the some steps of the previous procedure. Suppose that one is given the system

$$\begin{cases} (t_0)^{x_0\ x_1}_{s_0\ s_1} &= (x \times y) + t_1 \\ (t_0)^{x_3}_{s_3} &= (x \times x) + S(0). \end{cases}$$

As the algorithm suggests, one should start by considering the first equation (the other cases should also be studied). One should consider all the sub-term-structures of $(x \times y) + t_1$ and replace some occurrences by fresh variables, and then unify them, using substitutions, with x_0 or x_1; furthermore, one should proceed with the term-variables as described in 6.4.5. For example, for

$(x \times y) + t_1$, one could consider, after the suitable unifications, t_0 as being $(x_0 \times x_1) + t$, t_1 as being $t_{s_0}^{x_0}{}_{s_1}^{x_1}$, where t is fresh, s_0 as being x, and s_1 as being y. This constitutes a solution to the first equation.

Following the algorithm, then one should substitute the obtained solution in the second equation, giving $((x_0 \times x_1) + t)_{s_3}^{x_3} = (x \times x) + S(0)$, i.e.

$$((x_0)_{s_3}^{x_3} \times (x_1)_{s_3}^{x_3}) + t_{s_3}^{x_3} = (x \times x) + S(0).$$

This entails that

$$\begin{cases} (x_0)_{s_3}^{x_3} \times (x_1)_{s_3}^{x_3} & = x \times x \\ t_{s_3}^{x_3} & = S(0). \end{cases}$$

So, after another case analysis, one of the cases is

$$\begin{cases} (x_0)_{s_3}^{x_3} & = (x_1)_{s_3}^{x_3} = x \\ t_{s_3}^{x_3} & = S(0). \end{cases}$$

Hence, for this case, $x_0 = x$ and x_3 is not x_3, or $x_0 = x_3$ and $s_3 = x$; similarly for x_1. Let us now focus in the second equation from the previous system. We proceed as before with a case analysis, one of the cases yields that t as being $S(0)$; after that, one substitutes t in t_0, yielding $(x_0 \times x_1) + S(0)$, and in t_1, yielding $S(0)$. Observe that in the considered cases no **problem** was identified (it is not hard to see that the previous system is successful).

In the end we get, if not all the cases are rejections and no **problem** was obtained, substitutions σ in the desired conditions. Furthermore, such substitutions are most general ones, in the sense that they do not introduce unnecessary complexity and unnecessary identifications (just like a most general unifier).

Convention 3.23. We assume that **A** is a generic algorithm that solves some unification (of term-structures) problems (for example by a case analysis) and without the introduction of unnecessary complexity—i.e. has as output a finite number of most general unifiers in the sense that we used previously—, with some situations where it might output **problem**. We assume that **A** works in a very similar way to algorithm 3.22 (for instance, it might use a case analysis, it might use substitution of found solutions, it might use the creation of the systems, by considering all possible situation for the system, for some of them might output **problem**, etc). We will say that **A** is *successful* for a given system if it halts and no **problem** situations were identified during the computations, just like what was considered for the previous algorithm. We assume that **A** gives informations about the conditions needed for each step in the context of provable schemata (just like algorithm 3.22) and that has the following feature:

Sub. property If **A** is successful for the system

$$\begin{cases} r_0 & = s_0 \\ & \vdots \\ r_n & = s_n, \end{cases}$$

and there are concrete-substitutions Σ_0 and Σ_1 such that

$$\begin{cases} \Sigma_0(r_0) & = \Sigma_1(s_0) \\ & \vdots \\ \Sigma_0(r_n) & = \Sigma_1(s_n), \end{cases}$$

then there are a solution σ of the system that is constructed by the algorithm **A** and a concrete-substitution Σ such that

$$\begin{cases} \Sigma_0(r_0) = \Sigma(\sigma(r_0)) & = \Sigma(\sigma(s_0)) = \Sigma_1(s_0) \\ & \vdots \\ \Sigma_0(r_n) = \Sigma(\sigma(r_n)) & = \Sigma(\sigma(s_n)) = \Sigma_1(s_n). \end{cases}$$

We also assume that the previous property is compatible with the conventions that the algorithm gives as additional information and that if t is a term-structure, then $\Sigma(\sigma(t)) = \Sigma_i(t)$, for $i = 0, 1$, depending on the concrete-substitution for which the value of t defined (this last condition is useful when one is dealing with provable schemata).

It is important to observe that algorithm 3.22 has the **Sub. property**: if the algorithm 3.22 is successful for a considered system and one is given concrete-substitutions as before, then, guided by the concrete-substitutions Σ_0 and Σ_1, one can use the algorithm 3.22 to obtain a desired solution σ—here the concrete-substitutions can be used to identify the choices that one has to make while running the algorithm; moreover, the algorithm will be successful (by hypothesis); the existence of a concrete-substitution Σ with the mentioned properties follows from the fact that all the choices that were made for the construction of σ were guided by the concrete-substitutions Σ_0 and Σ_1.

Observe that one can conceive several algorithm that satisfy the **Sub. Property**; for instance, one can extend algorithm 3.22 to account for other possibilities without outputting **problem** so often. For example, one could extend algorithm 3.22 by allowing situations where one has occurrences of replacements in the two sides of the equality, like

$$(t_0)_{s_0}^{x_0} = (S(S(0)) + x_1) \times (t_1)_{s_1}^{x_1},$$

under the proviso that no variable-variable in the replacement in the opposite side can be identified with a variable-variable that occurs in the considered replacement. Observe that x_1 is a variable-variable in the replacement that occurs in the opposite side of the considered replacement, but, under the concrete-substitution interpretation, it cannot be x_0, since it occurs again in the right side, namely in $(S(S(0)) + x_1)$. Then, one makes the usual fresh variable analysis, but, similarly to stage 6.4.5, one also accounts for the possibility of internal application of the replacement; in the considered example, this means that one of the cases to be considered is the one where t_0 is $(x_0 + x_1) \times t_{s_1}^{x_1}$, with t fresh, s_0 is $S(S(0))$, and t_1 is $t_{s_0}^{x_0}$. Let us briefly justify

what was described. Let us suppose now that t_0, t_1, s_0, and s_1 are concrete terms, and x_0, x_1 concrete variables. Assume that x_0 is not x_1 and that

$$(t_0)_{s_0}^{x_0} = (t_1)_{s_1}^{x_1},$$

where both variables are being replaced, i.e. x_0 occurs in t_0 and x_1 in t_1. It is not hard to conclude that x_0 cannot occur in s_1, just like x_1 cannot occur in s_0. Consider t as being the term obtained from $(t_0)_{s_0}^{x_0}$ (that is the same as $(t_1)_{s_1}^{x_1}$) by replacing the occurrences of s_0 that were placed using the replacement by x_0, and the same for s_1 and x_1. Then, t has x_0 and x_1 as variables. Furthermore, it follows that

$$\begin{cases} t_{s_0}^{x_0} = & t_1 \\ t_{s_1}^{x_1} = & t_0, \end{cases}$$

as desired. This justifies the described procedure that one can add to algorithm 3.22. Observe that in the justification it was used the fact that x_0 is not x_1, otherwise the construction of t could fail; for example, for $t_0 = S(S(x_0))$, $s_0 = x$, $t_1 = S(x_1)$, and $s_1 = S(x)$ one has that $(t_0)_{s_0}^{x_0} = (t_1)_{s_1}^{x_1}$, but one cannot construct t as before, since s_0 is being placed in the same place as s_1. It is worth mentioning that the described impossibility is in the heart of the undecidability of second-order unification: keep in mind that, for example, $S(t_{x_1}^{x_0}) = t_{S(x_1)}^{x_0}$ has an infinity of solutions, namely $t = S^n(x_0)$, that cannot be written in a closed form without using natural numbers, in fact a variation of this is used to represent the natural numbers inside the context of second-order unification in [8] (the idea of the proof of the undecidability is to represent natural numbers, addition, and multiplication inside second-order unification and apply Matijasevič's theorem).

We believe that algorithm 3.22 halts for every input, but we do not have a proof of that fact or a counter-example to it; the reason for this is that we believe that if a loop situation is reached using substitutions and replacements, then one must have an occurrence of the same term-variable in both sides of a given equation, something that is accounted for by the algorithm by just outputting **problem**. All this concern about the halting nature of algorithm 3.22 is not necessary for what we are going to develop because we want to account for other algorithms that might not halt on every input (that is why we assumed that the previous convention), thus the halting nature of algorithm 3.22 is a side discussion to our goal.

We move to create the desired lists that give a more concrete inside of theorem 3.19.

List 3.24. We now proceed to create lists, for each k and each each algorithm $\mathbf{A}$, of provable schemata, $\mathcal{L}_{\mathbf{A},k}$, and proof-skeletons, $\mathfrak{L}_{\mathbf{A},k}$.

Basis case The list $\mathcal{L}_{\mathbf{A},0}$, for the case $k = 0$, is simply the (finite) list of axioms. The list $\mathfrak{L}_{\mathbf{A},0}$ is the list of the numbers of the schemata that are axioms.

Inductive step Suppose that the lists $\mathcal{L}_{\mathbf{A},s}$ and $\mathfrak{L}_{\mathbf{A},s}$, with $s \leq k$, where already created. Add all elements of $\mathcal{L}_{\mathbf{A},k}$ to $\mathcal{L}_{\mathbf{A},k+1}$, and all elements of $\mathfrak{L}_{\mathbf{A},k}$ to $\mathfrak{L}_{\mathbf{A},k+1}$. Consider the following cases:

Gen If S is a schema in $\mathcal{L}_{\mathbf{A},k}$, then pick x a variable-variable not occurring in S and add $\forall x.S$ to $\mathcal{L}_{\mathbf{A},k+1}$—the respective schema obtained by placing a universal quantifier, just like in definition 3.8. If $\mathcal{S}$ is a proof-skeleton in $\mathfrak{L}_{\mathbf{A},k}$, then add $\mathrm{Gen}(\mathcal{S})$ to $\mathfrak{L}_{\mathbf{A},k+1}$.

MP Take $k = k_0 + k_1$. Do the following:

1. Consider S_0 a schema in $\mathcal{L}_{\mathbf{A},k_0}$ and S_1 a schema in $\mathcal{L}_{\mathbf{A},k_1}$.
2. If S_1 is a universal quantification or a negation, then reject and consider another pair.
3. Suppose that
   - $S_0 := F[\varphi_0^0, \ldots, \varphi_{n_0}^0, t_0^0, \ldots, t_{n_1}^0, v_0^0, \ldots, v_{n_2}^0] \& \bigvee_{i \in I^0} \&_{j \in J_i^0} \sim^{k_j^{0,0}} C_{k_j^{0,1}}(A_i^0, t_{k_j^{0,2}}, v_{k_j^{0,3}})$, and
   - $S_1 := G[\varphi_0^1, \ldots, \varphi_{n_3}^1, t_0^1, \ldots, t_{n_4}^1, v_0^1, \ldots, v_{n_5}^1] \rightarrow H[\varphi_0^2, \ldots, \varphi_{n_6}^2, t_0^2, \ldots, t_{n_7}^2, v_0^2, \ldots, v_{n_8}^2] \& \bigvee_{i \in I^1} \&_{j \in J_i^1} \sim^{k_j^{1,0}} C_{k_j^{1,1}}(A_i^1, t_{k_j^{1,2}}, v_{k_j^{1,3}})$.
4. We now proceed to see whether F and G can be unified by means of a substitution σ using the algorithm **A** and the previously developed methods. We assume that the conventions were already applied to these schemata.
5. Using the ideas of the previous proofs (with the suitable adaptations), it is not hard to see that one can test whether F and G have a common layout.
6. If they do not have, then reject and consider another pair of schemata.
7. If they have, find the common layout in such a way that unnecessary complexity is avoided (follow the ideas of the previous proofs).
8. Starting from the first entry of the common layout, do the following:
   - 8.1. Proceed with the quantifiers and negation signs as before (for instance, like in the proof of lemma 3.13).
   - 8.2. One should create a system for the cases where in both entries one has something of the form $X_0 \cdots X_n r = s$, with r and s term-structures. Act accordingly with the quantifiers and the negation signs. If it is not possible, then reject and consider another pair of proof-skeletons.
   - 8.3. One should run algorithm **A** (for example algorithm 3.22) for the needed equalities of term-structures that emerge.
   - 8.4. If **A** is not successful, then reject and consider another pair of schemata.
   - 8.5. Assume for the rest of the procedure that the algorithm **A** is successful. Take note of all solutions.
   - 8.6. Consider the case where one has in one entry $X_0^0 \cdots X_{\ell_0}^0(\varphi_0)_{\vec{s}_0}^{\vec{x}_0}$ and in the other one has $X_0^1 \cdots X_{\ell_1}^1 r = s$. Firstly, act accordingly with the quantifiers and negation signs (see, for example, 4.1.1 of the

proof of lemma 3.13). If it is not possible, then reject and consider another pair of schemata.

8.7. If any of the $\vec{x_0}$ appears in $r = s$, then then reject and consider another pair of schemata. If φ_0 was already assigned to something of the form $X_0^2 \cdots X_{\ell_2}^2 r' = s'$, then apply the considered replacement (this yields a case analysis). Apply the conventions and force it to be $X_0^1 \cdots X_{\ell_1}^1 r = s$; this will give rise to another system that one should solve using algorithm **A**. If the system is not successful, then reject and consider another pair of schemata. If it is, save the solutions. If φ_0 was not yet assigned to such a structure, then consider φ_0 as being $t = t'$, with t and t' fresh term-variables. Run the algorithm **A** to solve the systems that emerge and proceed only in the case where the algorithm is successful (all the case analyses should include the suitable information about the conditions that are needed at each stage).

8.8. One now considers all entries were ones has something of the form $X_0 \cdots X_n(\varphi_0)_{\vec{s_0}}^{\vec{x_0}}$ in both entries.

8.9. We assume, due to our conventions, that in the considered case we do not have the same formula-variable occurring. Act accordingly with the quantifiers and the negation signs. If it is not possible, then reject and consider another pair of schemata.

8.10. If both formula-variables were already assigned to something of the form $X_0 \cdots X_{\ell_0} r = s$, then apply the replacements, apply the conventions, and, for each case of the conventions, solve the obtained system in the previously mentioned ways. If any of the systems is not successful, reject and consider another pair of schemata.

8.11. If only one of them was mapped to the mentioned structure, adapt step 8.6.

8.12. Suppose now that non of the formula-variables was assigned. Then, one should unify them like what was done for the case of algorithm 3.22 for term-variables. If one needs to satisfy an equality where in both members of the equality one has something of the form $\varphi_{\vec{s}}^{\vec{x}}$, then reject and consider another pair of schemata; otherwise proceed as before by substituting the already attributed values. For example, if one has the equations $\varphi_0 = \varphi_1$, $\varphi_1 = \varphi_2$, then one assigns all those formula-variables to a common fresh formula-variable, say φ; if one has $\varphi_0 = \varphi_s^x$ and $\varphi_2 = \varphi_0$, then one assigns φ_2 to φ_s^x.

8.13. Apply the previous steps to all entries.

9. If we do not get a rejection for the considered schemata in the previous procedure, then apply to each of the final results the conventions and

then add the all resulting provable schemata to the list $\mathfrak{L}_{\mathbf{A},k+1}$:

$$H[\sigma(\varphi_0^2),\dots,\sigma(\varphi_{n_6}^2),\sigma(t_0^2),\dots,\sigma(t_{n_7}^2),\sigma(v_0^2),\dots,\sigma(v_{n_8}^2)]\&$$
$$\bigvee_{i\in I^0}\&_{j\in J_i^0}\sim^{k_j^{0,0}}C_{k_j^{0,1}}(\sigma(A_i^0),\sigma(t_{k_j^{0,2}}),\sigma(v_{k_j^{0,3}}))\&$$
$$\bigvee_{i\in I^1}\&_{j\in J_i^1}\sim^{k_j^{1,0}}C_{k_j^{1,1}}(\sigma(A_i^1),\sigma(t_{k_j^{1,2}}),\sigma(v_{k_j^{1,3}}))\left(\&\, C\right),$$

where C are, possibly, the conditions that appear from the conventions and the solutions of the considered systems.

10. Consider $\mathcal{S}_0$ a proof-skeleton in $\mathfrak{L}_{\mathbf{A},k_0}$ and $\mathcal{S}_1$ a proof-skeleton in $\mathfrak{L}_{\mathbf{A},k_1}$.
11. If for all schemata S_0 in $\mathcal{L}_{\mathbf{A},k_0}$ with skeleton $\mathcal{S}_0$ and S_1 in $\mathcal{L}_{\mathbf{A},k_1}$ with skeleton $\mathcal{S}_1$, the previous procedure does not yield a rejection, then one should add $\mathrm{MP}(\mathcal{S}_0,\mathcal{S}_1)$ to $\mathfrak{L}_{\mathbf{A},k+1}$; if for any of them one rejects, then one should consider another pair of proof-skeletons and do the same move.

It is important to observe that the previous constructions does not contradict the undecidability of second-order unification problem (see, for instance, [8]): we are considering different types of terms, namely term-structures; we are considering a different type of substitutions σ; and in algorithm 3.22 we do not allow the occurrence of a term-variable in both sides of an equation, something that is indispensable in the proof of the undecidability of second-order unification in [8]. One should keep in mind that the algorithms that we are considering are all necessarily partial—they cannot solve successfully all systems.

Observe that for each k and $\mathbf{A}$, the lists $\mathcal{L}_{k,\mathbf{A}}$ and $\mathfrak{L}_{k,\mathbf{A}}$ are finite—this follows by construction, in particular from the fact that for each convention there is a finite number of ways to apply it, and from the fact that the systems that are successful have a finite number of (most general) solutions for the considered algorithm. Although we presented an inductive construction of the lists, they are not necessarily computable uniformly in k; nevertheless, for each fixed k, $\mathcal{L}_{\mathbf{A},k}$ and $\mathfrak{L}_{\mathbf{A},k}$ are computable due to the fact that they are finite (and all finite lists are computable). The computable uniformity of the lists would entail that, for small values of k, one could, for the considered algorithm $\mathbf{A}$, computationally decide if a given system is successful for $\mathbf{A}$ (a feature that fails for most algorithms).

Definition 3.25. We say that a proof-skeleton $\mathcal{S}$ is *grounded for* $\mathbf{A}$ if $\mathcal{S}$ is in $\mathfrak{L}_{\mathbf{A},k}$, where k is the number of steps of $\mathcal{S}$. We say that k is a *grounded number for* $\mathbf{A}$ if all proof-skeleton whose number of steps is at most k are grounded for $\mathbf{A}$. We will consider $\mathcal{L}_{\mathbf{A}} := \cup_k \mathcal{L}_{\mathbf{A},k}$ and $\mathfrak{L}_{\mathbf{A}} := \cup_k \mathfrak{L}_{\mathbf{A},k}$.

If one makes some assumptions about the way the lists were generated—if one assumes that there is a general way to create them, if one assumes that in the construction one does not include unnecessary complexity, etc—, for most cases, the stable proof-skeletons are also grounded for some algorithm $\mathbf{A}$.

The intuition behind grounded proof-skeletons is that, to such skeletons, one can apply the intuitive reasoning made in the beginning of section 3.1

for the analysis of the proofs whose general structure is given by the skeleton $\mathrm{MP}([L]), \mathrm{MP}([L]), [|rmL)]))$. Furthermore, a grounded number is a number such that all proof-skeletons of proofs that have that very same number as the maximum number of steps are grounded.

Theorem 3.26. *Given $\mathcal{S}$ a grounded proof-skeleton for $\mathbf{A}$ in $\mathfrak{L}_{\mathbf{A},k}$, if φ has a proof whose skeleton is $\mathcal{S}$, then there are a schema S in $\mathcal{L}_{\mathbf{A},k}$ with skeleton $\mathcal{S}$, generated by the algorithms, and a concrete-substitution Σ such that $\Sigma(S) = \varphi$.*

Proof. Let us prove the result by induction on k. If $\mathcal{S}$ is in $\mathfrak{L}_{\mathbf{A},0}$, then $\mathcal{S}$ is the number of an axiom; thus, if φ has a proof whose skeleton is $\mathcal{S}$, then there are a schema S in $\mathcal{L}_{\mathbf{A},0}$ with skeleton $\mathcal{S}$, generated by the algorithms, and a concrete-substitution Σ such that $\Sigma(S) = \varphi$. Suppose, by induction hypothesis, that the result holds for $s \leq k$. Suppose that $\mathcal{S}$ is in $\mathfrak{L}_{\mathbf{A},k+1}$ and that φ has a proof whose skeleton is $\mathcal{S}$. Consider the following cases:

$\mathcal{S} = \mathrm{Gen}(\mathcal{S}_0)$ In this case, one must have $\varphi = \forall x.\psi$ and ψ should have a proof whose skeleton is $\mathcal{S}_0$. By construction, $\mathcal{S}_0$ must be in $\mathfrak{L}_{\mathbf{A},k}$. By induction hypothesis, there are a schema S_0 in $\mathcal{L}_{\mathbf{A},k}$ with skeleton $\mathcal{S}_0$, generated by the algorithms, and a concrete-substitution Σ such that $\Sigma(S_0) = \psi$. It is clear that the schema $\forall v.S_0$, obtained by S_0 using the generalisation rule, is in $\mathcal{L}_{\mathbf{A},k+1}$; furthermore, $\forall v.S_0$ has skeleton $\mathcal{S}$. Moreover, one can extend Σ in such a way that $\Sigma(\forall v.S_0) = \forall x.\Sigma(S_0) = \forall x.\psi = \varphi$.

$\mathcal{S} = \mathrm{MP}(\mathcal{S}_0, \mathcal{S}_1)$ In this case, there must be ψ such that ψ has a proof whose skeleton is $\mathcal{S}_0$ and $\psi \to \varphi$ has a proof whose skeleton is $\mathcal{S}_1$. So, one has $\mathcal{S}_0$ in $\mathfrak{L}_{\mathbf{A},k_0}$ and $\mathcal{S}_1$ in $\mathfrak{L}_{\mathbf{A},k_1}$, with $k = k_0 + k_1$. By induction hypothesis, there are schemata S_0 in $\mathcal{L}_{\mathbf{A},k_0}$ with skeleton $\mathcal{S}_0$ and S_1 in $\mathcal{L}_{\mathbf{A},k_1}$ with skeleton $\mathcal{S}_1$, and concrete-substitutions Σ_0 and Σ_1 such that $\Sigma_0(S_0) = \psi$ and $\Sigma_1(S_1) = \psi \to \varphi$. Lemma 3.13 guarantees that there is a substitution that unifies F and G; furthermore, guided by the concrete-substitutions Σ_0 and Σ_1, one can use the previous algorithms to obtain a minimal unifier σ for the algorithm $\mathbf{A}$ (the concrete-substitutions can be used to see what choices should be done while running the algorithm)—this follows from the **Sub. property** of convention 3.23 (and from the fact that a version of **Sub. Property** holds for formula-variables); moreover, the algorithm will be successful because $\mathcal{S}$ is grounded. Thus, using σ from the algorithm, $\sigma(F) = \sigma(G)$. Clearly, the schema $\sigma(H)$—this schema includes the suitable conventions—is in $\mathcal{L}_{\mathbf{A},k+1}$. Moreover, using the reasoning of the proof of lemma 3.13 and the **Sub. property**, one can guarantee the existence a concrete-substitution Σ such that $\Sigma(\sigma(H)) = \varphi$.

The result follows by induction. □

Corollary 3.27. *Given $\mathcal{S}$ a grounded proof-skeleton for $\mathbf{A}$, for each formula φ, it is decidable whether there is a proof of φ whose skeleton is $\mathcal{S}$.*

Proof. Consider $\mathcal{S}$ a grounded proof-skeleton for $\mathbf{A}$. Generate, using the previous procedures for the algorithm $\mathbf{A}$, the schemata that have $\mathcal{S}$ as proof-skeleton, let us call this finite list $\mathcal{R}$. These obtained schemata are minimal in the sense that there was added no unnecessary complexity to the substitutions. Let us prove that φ has a proof whose skeleton is $\mathcal{S}$ if, and only if, there are a concrete-substitution Σ and S in $\mathcal{R}$ such that $\Sigma(S) = \varphi$. It is clear that if there are a concrete-substitution Σ and S in $\mathcal{R}$ such that $\Sigma(R) = \varphi$, then φ has a proof whose skeleton is $\mathcal{S}$ (this can be proved by induction on the definition of proof-skeleton). Let us prove the other direction. Suppose that φ has a proof whose skeleton is $\mathcal{S}$. Then, by the previous theorem, there are a schema S with skeleton $\mathcal{S}$, generated by the algorithms, and a concrete-substitution Σ such that $\Sigma(S) = \varphi$. Clearly S is in $\mathcal{R}$, so desired result holds.

The decidability follows from the fact the the list $\mathcal{R}$ is finite and from theorem 3.17. □

From the proof of the previous result we can conclude that every grounded skeleton is stable. Thus, every grounded number is stable.

Theorem 3.28. *For k a grounded number for $\mathbf{A}$, $T \vdash_{k\,steps} \varphi$ if, and only if, there are a schema S in $\mathcal{L}_{\mathbf{A},k}$ and a concrete-substitution Σ such that $\Sigma(S) = \varphi$.*

Proof. Clearly, if there is a schema S in $\mathcal{L}_{\mathbf{A},k}$ and a concrete-substitution Σ such that $\Sigma(S) = \varphi$, then $T \vdash_{k\text{steps}} \varphi$(this follows by a simple induction argument). Let us prove the other direction by induction on k. It is clear that the result holds for $k = 0$, the axiom case. Suppose, by induction hypothesis, that the result holds for all $s \leq k$. Suppose that $k+1$ is a grounded number for $\mathbf{A}$ and consider φ such that $T \vdash_{k+1\text{steps}} \varphi$. We have two cases:

Last step uses Gen In this case, we have that $\varphi = \forall x.\psi$ and $T \vdash_{k\text{steps}} \psi$. It follows from the definition that k is a grounded number for $\mathbf{A}$. By induction hypothesis, there are a schema S in $\mathcal{L}_{\mathbf{A},k}$ and a concrete-substitution Σ such that $\Sigma(S) = \psi$. Consider $\forall v.S$ the schema that is obtained from S by the generalisation rule. Clearly, $\forall v.S$ is in $\mathcal{L}_{\mathbf{A},k+1}$. Furthermore, we can extend Σ in such a way that $\Sigma(\forall v.S) = \forall x.\psi = \varphi$.

Last step uses MP In this case, there is a formula ψ such that $T \vdash_{k_0\text{steps}} \psi$ and $T \vdash_{k_1\text{steps}} \psi \to \varphi$, with $k = k_0 + k_1$. It follows that k_0 and k_1 are grounded numbers for $\mathbf{A}$. By induction hypothesis, there are schemata F in $\mathcal{L}_{\mathbf{A},k_0}$ and $G \to H$ in $\mathcal{L}_{\mathbf{A},k_1}$, and concrete-substitutions Σ_0 and Σ_1 such that $\Sigma_0(F) = \psi$ and $\Sigma_1(G \to H) = \psi \to \varphi$. By the reasoning of the proof of theorem 3.26, we can guarantee the existence of a suitable substitution σ delivered by the algorithm such that $\sigma(F) = \sigma(G)$. As $k+1$ is a grounded number for $\mathbf{A}$, the algorithm must be successful and thus $\sigma(H)$ is in $\mathcal{L}_{\mathbf{A},k+1}$. As σ is minimal in the sense of

introduction of unnecessary complexity and by the reasoning of the proof of theorem 3.26 (the fact that σ was chosen using the concrete-substitutions), there must be a concrete-substitution Σ such that $\Sigma(\sigma(H)) = \varphi$.

The result follows by induction. One could also prove the result using the proof of corollary 3.27. □

Corollary 3.29. *Given k a grounded number for $\mathbf{A}$, it is decidable whether $T \vdash_{ksteps} \varphi$ or not.*

Proof. One considers the finite list $\mathcal{L}_{\mathbf{A},k}$. By the previous result, it is enough to see whether there is a concrete-substitution Σ and S in $\mathcal{L}_{\mathbf{A},k}$ such that $\Sigma(S) = \varphi$, something that is decidable from the fact that $\mathcal{L}_{\mathbf{A},k}$ is finite and from theorem 3.17. This also follows from theorem 3.20. □

Corollary 3.30. *Given k a grounded number for $\mathbf{A}$ in PA′, it is decidable whether $PA' \vdash_{ksteps} \varphi$ or not.*

Proof. Follows from the previous result when one has in mind that PA′ is one of the considered theories. □

Corollary 3.31. *Given k a grounded number for $\mathbf{A}$ in PA′, it is decidable whether $PA \vdash_{ksteps} \varphi$ or not.*

Proof. Follows from the previous corollary and theorem 2.4. □

Theorem 3.32. *Given $\mathbf{A}$, there is an* infinity *of schemata in $\mathcal{L}_{\mathbf{A}}$.*

Proof. It is not hard to see that all schemata that are constructed using only the propositional logic axioms, L1)–L3) in the initial list, are in $\mathcal{L}_{\mathbf{A}}$. Furthermore, besides these propositional schemata, there are much more schemata in $\mathcal{L}_{\mathbf{A}}$: the only restriction that ones has is that they do not yield the **problem** cases in their construction and the algorithm halts without rejecting. □

Theorem 3.33. *Given an algorithm $\mathbf{A}$, there is an algorithm $\mathbf{H}_{\mathbf{A}}$ such that, for every grounded proof-skeleton $\mathcal{S}$ and every formula φ, the algorithm halts and accepts for $\mathcal{S}$ and φ if, and only if, φ has a proof whose skeleton is $\mathcal{S}$.*

Proof. Fix an algorithm $\mathbf{A}$. For a proof-skeleton $\mathcal{S}$, the algorithm $\mathbf{H}_{\mathbf{A}}$—using the construction of the schemata in $\mathcal{L}_{\mathbf{A}}$—tries to generate all the provable schemata in $\mathcal{L}_{\mathbf{A}}$ whose skeleton is $\mathcal{S}$. Observe that in the construction of the lists $\mathcal{L}_{\mathbf{A},k}$ one needed to make assumptions about the successfulness of the algorithm $\mathbf{A}$ for certain systems, but in this algorithm we do not make those (possibly non-computable) assumptions; the rest of the process remains the same, but one only focus on the construction of the schemata that potentially have skeleton $\mathcal{S}$. This yields no problem because the construction of the lists is computable with possible exception of the successfulness conditions. For non-grounded proof-skeletons (not in $\mathcal{L}_{\mathbf{A}}$) the algorithm might not halt. Suppose that $\mathcal{S}$ is a grounded proof-skeleton. Then, the algorithm $\mathbf{H}_{\mathbf{A}}$ can successfully construct the lists $\mathcal{R}$ from the proof of corollary 3.27. Thus, using theorem 3.17, the algorithm can decide whether φ has a proof whose skeleton is $\mathcal{S}$. □

The algorithms $\mathbf{H_A}$ have, in a sense, implemented the idea of the analysis made to the skeleton $\mathrm{MP}([L]), \mathrm{MP}([L]), [|rmL)])$ in the beginning of section 3.1.

Theorem 3.34. *If T is such that $T \vdash_{ksteps} \varphi$ is uniformly decidable in k, then there is a recursive function $f(k, \varphi)$ such that*

$$T \vdash_{ksteps} \varphi \implies T \vdash_{f(k,\varphi)\ symbols} \varphi.$$

Proof. Assume $T \vdash_{\mathrm{ksteps}} \varphi$ is uniformly decidable in k. Consider the partial-recursive functions

$$(k, \varphi) := \mu n[n \text{ is the code of a proof of } \varphi \text{ in } T \text{ with at most } k \text{ steps}],$$

and

$$sym(s) := \begin{cases} \text{number of symbols in} \\ \text{the proof of code is } s, & s \text{ is the code of a proof in } T \\ 0, & \text{otherwise.} \end{cases}$$

By hypothesis, we can decide uniformly in k if $T \vdash_{\mathrm{ksteps}} \varphi$ holds or not. Thus, the function

$$f(k, \varphi) := \begin{cases} sym(c(k, \varphi)), & T \vdash_{\mathrm{ksteps}} \varphi \\ 0, & \text{otherwise.} \end{cases}$$

is, by construction, a total recursive-function that satisfies the desired property. □

Theorem 3.35. *If T is such that there is a recursive function $f(k, \varphi)$ such that*

$$T \vdash_{ksteps} \varphi \implies T \vdash_{k\ steps\ and\ f(k,\varphi)\ symbols} \varphi,$$

then $T \vdash_{ksteps} \varphi$ is uniformly decidable in k.

Proof. Assume there is a recursive-function $f(k, \varphi)$ satisfying the considered property. Let us consider the following algorithm.

1. Input: k and φ.
2. Compute $f(k, \varphi)$. If there is a proof of φ in k steps, then there is a proof of φ with at most $f(k, \varphi)$ symbols; such a proof would use at most $f(k, \varphi)$ variables, furthermore it does not matter the choice of variables that one makes in the sense that if one changes all the occurrences of a given variable in the proof one continues to have a sound proof. Take a finite list of at most $f(k, \varphi)$ variables.
3. Consider a finite list $\mathcal{I}_0$ of symbols consisting of: the logical symbols ('$\forall$', '$\neg$', and '$\rightarrow$'), '$=$', '(', ')', '$+$', '$\times$', '0' and the previously mentioned finite list of variables. Consider also a blank symbol '**B**' (just to separate the candidate formulas in a proof to be) not in $\mathcal{I}_0$.
4. Using only symbols from $\mathcal{I}_0$ and '**B**', generate a list $\mathcal{I}$ of all the (finitely many) possible lists of symbols which contain at most $f(k, \varphi)$ ones from $\mathcal{I}_0$ that have φ as the last element of the list.

5. Test if any element of $\mathcal{I}$ is a proof in T with k steps (clearly, this can be done in a computational manner): output 1 in affirmative case, and 0 in the negative case.

It is not hard to see that the previous algorithm decides uniformly in k the relation $T \vdash_{k\text{steps}} \varphi$. □

Consider PA^a as being any formulation of PA considered in [6] and proved to have a decidable k-provability. The next result is a solution to the problem 20 of [4] for these formulations, a problem proposed by Krajíček.

Corollary 3.36. *There is a recursive-function $f(k, \varphi)$ such that*

$$PA^a \vdash_{k steps} \varphi \implies PA^a \vdash_{f(k,\varphi)\ symbols} \varphi.$$

Proof. Follows from theorem 3.34 and the fact that $\mathrm{PA}^a \vdash_{k\text{steps}} \varphi$ was proved to be decidable uniformly in k (see [6]). □

Funding Open Access funding enabled and organized by Projekt DEAL.

References

[1] Baaz, M., Pudlák, P.: Kreisel's conjecture for L∃1. In: Clote, P., Krajićek, J. (eds.) Arithmetic, Proof Theory and Computational Complexity, pp. 29–59. Oxford University, Oxford (1993)

[2] Buss, S.R.: The undecidability of k-provability. Ann. Pure Appl. Logic **53**(1), 75–102 (1991)

[3] Cavagnetto, S.: The lengths of proofs: Kreisel's conjecture and Gödel's speed-up theorem. J. Math. Sci. **158**(5), 689–707 (2009)

[4] Clote, P., Krajíček, J.: Open problems. In: Clote, P., Krajíček, J. (eds.) Arithmetic, Proof Theory and Computational Complexity, pp. 1–19. Oxford University, Oxford (1993)

[5] Enderton, H.B.: A Mathematical Introduction to Logic. Elsevier, Amsterdam (2001)

[6] Farmer, W.M.: A unification-theoretic method for investigating the k-provability problem. Ann. Pure Appl. Logic **51**(3), 173–214 (1991)

[7] Farmer, W.M.: The Kreisel length-of-proof problem. Ann. Math. Artif. Intell. **6**(1–3), 27–55 (1992)

[8] Krajíček, J., Pavel, P.: The number of proof lines and the size of proofs in first order logic. Arch. Math. Logic **27**(1), 69–84 (1988)

[9] Miyatake, T.: On the length of proofs in a formal systems. Tsukuba J. Math. **4**(1), 115–125 (1980)

[10] Orevkov, V.P.: Complexity of Proofs and Their Transformations in Axiomatic Theories. Translations of Mathematical Monographs, vol. 128. American Mathematical Society, Providence (1993)

[11] Parikh, R.J.: Some results on the length of proofs. Trans. Am. Math. Soc. **177**, 29–36 (1973)

[12] Pavel, H.: Theories very close to PA where Kreisel's Conjecture is false. J. Symb. Log. **72**(1), 123–137 (2007)

Paulo Guilherme Santos and Reinhard Kahle
Centro de Matemática e Aplicações, NOVA School of Science and Technology
2829-516 Caparica
Portugal
e-mail: pgd.santos@campus.fct.unl.pt;
kahle@mat.uc.pt

Reinhard Kahle
Theorie und Geschichte der Wissenschaften, Universität Tübingen
Keplerstr. 2
72074 Tübingen
Germany

Received: September 20, 2020.
Accepted: May 1, 2021.