Size-based termination of higher-order rewrite systems

Frédéric Blanqui

Size-based termination of higher-order rewrite systems

Frédéric Blanqui* (INRIA, France)

3 January 2017

Abstract. This paper is concerned with the termination, in Church' simply-typed λ-calculus, of the combination of β-reduction and arbitrary user-defined rewrite rules fired using matching modulo α-congruence only.

Several authors have devised termination criteria for fixpoint-based function definitions using deduction rules for bounding the size of terms inhabiting inductively defined types, where the size of a term is (roughly speaking) the set-theoretical height of the tree representation of its normal form.

In the present paper, we extend this approach to rewriting-based function definitions and more general notions of size.

1 Introduction

Consider the rewrite system of Figure 1 defining the substraction and division functions on the sort N of natural numbers in unary notation, *i.e.* with the constructors $\mathsf{0} : \mathsf{N}$ for zero and $\mathsf{s} : \mathsf{N} \Rightarrow \mathsf{N}$ for the successor function.

Figure 1: Rewrite system defining substraction and division on natural numbers

$$\begin{array}{rcl}
\mathsf{sub}\ x\ \mathsf{0} & \to & x \\
\mathsf{sub}\ \mathsf{0}\ y & \to & \mathsf{0} \\
\mathsf{sub}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) & \to & \mathsf{sub}\ x\ y \\
 & & \\
\mathsf{div}\ \mathsf{0}\ (\mathsf{s}\ y) & \to & \mathsf{0} \\
\mathsf{div}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) & \to & \mathsf{s}\ (\mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y))
\end{array}$$

By termination, we mean the strong normalization property, that is, the absence of infinite rewrite sequences $t_0 \to t_1 \to \ldots$ A weaker property is the existence of a normal form or weak normalization.

In our example, the size of a terminating term t of sort N is the number of s symbols at the top of the normal form of t (this rewrite system is weakly orthogonal and thus confluent [vO94]).

*LSV, 61 avenue du Président Wilson, 94235 Cachan Cedex, France.

While the termination of sub (*i.e.* the absence of infinite reductions starting from a term of the form sub t u with t and u in normal form) is not very difficult to establish (the size of the first argument is strictly decreasing in recursive calls), proving the termination of div requires to observe that sub is not size-increasing, that is, the size of (sub t u) is less than or equal to the size of t.

The idea of sized types, introduced by Hughes, Pareto and Sabry in [HPS96] for fixpoint-based function definitions, is to consider an abstract interpretation of this notion of size into an algebra of symbolic size expressions, and turn the usual typing rules of simply-typed λ-calculus into deduction rules on the size of terms. This allows us to automatically deduce some information on the size of terms, and thus prove termination by checking that, for instance, the size of some given argument decreases in recursive calls. Hence, termination is reduced to checking typing and abstract size decreasingness.

In our example, this amounts to say: the 2nd rule of div does not jeopardize termination since, assuming that x is instantiated by a term t of abstract size α, and y is instantiated by a term u of abstract size β, then div (s t) (s u) terminates because its first argument is of size $\alpha + 1$ while, in the recursive call div (sub t u) (s u), the first argument has a size smaller than or equal to α.

The goal of this work is to automate this kind of (inductive) reasoning, and check the information given by the user (here, the fact that sub is not size-increasing).

However, when considering type constructors taking functions as arguments (*e.g.* Sellink's model of μCRL [Sel93], Howard's constructive ordinals in Example 4, continuations in Example 7), the size of a term is generally not a finite natural number but a transfinite ordinal number. Hopefully, (finite) abstract size expressions can also handle transfinite sizes.

Before explaining our contributions and detailing the outline of the paper, we give hereafter a short survey on the use of ordinals for proving termination since this is at the heart of our work though, in the end, we provide an ordinal-free termination criterion.

1.1 Ordinal-based termination

A natural (and trivially complete) method for proving the termination of a relation $\rightarrow$ consists in considering a well-founded domain $(\mathbb{D}, <_{\mathbb{D}})$, *e.g.* some ordinal $(\mathfrak{h}, <_{\mathfrak{h}})$, assigning a "size" $\|t\| \in \mathbb{D}$ to every term t, and checking that every rewrite step (including β-reduction) makes the "size" strictly decrease: $\|t\| >_{\mathbb{D}} \|u\|$ whenever $t \rightarrow u$.

In theory, it is enough to take $\mathbb{D} = \omega$ (the first infinite ordinal) when the rewrite relation is finitely branching. However, after Gödel's incompleteness theorem [Göd31], defining $\| \ \|$ and proving that $\|t\| >_{\mathbb{D}} \|u\|$ whenever $t \rightarrow u$, may require the use of much bigger ordinals. For instance, the termination of cut-elimination in Peano arithmetic (PA) requires induction up to the ordinal $\varepsilon_0 = \omega^{\omega^{\cdot^{\cdot^{\cdot}}}}$ but PA cannot prove the well-foundedness of ε_0 itself [Gen35]. Yet, there is a function $\| \ \|$ from the terms of Gödel' system T [Göd58] (which extends

PA) to ω such that $\|t\| >_{\mathbb{D}} \|u\|$ whenever $t \to u$ [Wei98].

An equivalent approach is finding a well-founded relation containing $\to$, like Dershowitz' recursive path ordering (RPO) [Der79b, Der82] or its extension to the higher-order case by Jouannaud and Rubio [JR99, JR07, BJR15] but, in this paper, we will focus on the explicit use of size functions. For a connection between RPO and ordinals, see for instance [DO88].

Early examples of this approach are given by Ackermann's proof of termination of second-order primitive recursive arithmetic functions using $\mathfrak{h} = \omega^{\omega^\omega}$ [Ack25], Gentzen's proof of termination of cut elimination in Peano arithmetic using $\mathfrak{h} = \varepsilon_0$ [Gen35, How70, WW12], Turing's proof of weak normalization of Church' simply-typed λ-calculus [Tur42], Howard's proof of termination of his system V, an extension of Gödel' system T with some higher-order inductive type, using Bachmann's ordinal [How72]. This approach developed into a whole area of research for measuring the logical strength of axiomatic theories, involving ever growing ordinals, that can hardly be automated. See for instance [Rat06] for some recent survey. Instead, Monin and Simonot developed an algorithm for trying to find size assignments in $\mathfrak{h} = \omega^\omega$ [MS01].

But, up to now, there has been no ordinal analysis for powerful theories like second-order arithmetic: the termination of cut elimination in such theories is based on another approach introduced by Girard [Gir72, GLT88], which consists in interpreting types by so-called computability predicates and typing by the membership relation.

In the first-order case, *i.e.* when there is no rule with abstraction or applied variables, size-decreasingness can be slightly relaxed by conducting a finer analysis of the possible sequences of function calls. This led to the notions of dependency pair in the theory of first-order rewrite systems [Art96, AG00, HM05, GTSKF06], and size-change principle for first-order functional programs [LJBA01]. These two notions are thoroughly compared in [TG05]. In both cases, it is sufficient to define a measure on the class of terms which are arguments of a function call only. Various extensions to the higher-order case have been developed [SWS01, Wah07, JB08, KISB09, Kop11], but no general unifying theory yet.

The present paper is not concerned with this problem but with defining a practical notion of size for simply-typed λ-terms inhabiting inductively defined types.

Note by the way that the derivational complexity of a rewrite system, *i.e.* the function mapping every term t to the maximum number of successive rewrite steps one can do from t [HL89], does not seem to be related, at least in a simple way, to the ordinal necessary to prove its termination: there are rewrite systems whose termination can be proved by induction up to ω only and yet have huge derivational complexities [Mos14], unless perhaps one bounds the growth rate of the size of terms (measured here as the number of symbols) [Sch14]. The notion of runtime complexity, *i.e.* the function mapping every $n \in \mathbb{N}$ to the maximum number of successive rewrite steps one can do from a term which subterms are in normal form and which size is smaller than n, seems to provide a better (Turing related) complexity model [AM10].

1.2 Model-based termination

In [MN70], Manna and Ness proposed to interpret every term whose free variables are $x_1, \ldots, x_n$ by a function from $\mathbb{E}^n$ to $\mathbb{E}$, where $(\mathbb{E}, <_{\mathbb{E}})$ is a well-founded domain. That is, $\mathbb{D}$ is the set of all the functions from some power of $\mathbb{E}$ to $\mathbb{E}$ and $<_{\mathbb{D}}$ is the pointwise extension of $<_{\mathbb{E}}$, *i.e.* $f : \mathbb{E}^n \to \mathbb{E} <_{\mathbb{D}} g : \mathbb{E}^n \to \mathbb{E}$ if, for all $x_1, \ldots, x_n \in \mathbb{E}$, $f(x_1, \ldots, x_n) <_{\mathbb{E}} g(x_1, \ldots, x_n)$.

In the first-order case, this can be done in a structured way by interpreting every function symbol f of arity n by a function $\mathsf{f}_{\mathbb{E}} : \mathbb{E}^n \to \mathbb{E}$ and every term by composing the interpretations of its symbols, *e.g.* $\|\mathsf{f}\,(\mathsf{g}\,x)\|$ is the function mapping x to $\mathsf{f}_{\mathbb{E}}(\mathsf{g}_{\mathbb{E}}(x))$. If moreover these interpretation functions are monotone in each argument, then checking that rewriting is size-decreasing can be reduced to checking that every rule is size-decreasing.

A natural domain for $(\mathbb{E}, <_{\mathbb{E}})$ is of course $(\mathbb{N}, <_{\mathbb{N}})$. In this case, both monotony and size-decreasingness can be reduced to absolute positivity. Indeed,

$f(x_1, \ldots, x_p) > g(x_1, \ldots, x_q)$ is equivalent to $f(x_1, \ldots, x_p) - g(x_1, \ldots, x_q) - 1 \geq 0$

and monotony is equivalent to check that, for all i, $f(\ldots, x_i+1, \ldots) - f(\ldots, x_i, \ldots) - 1 \geq 0$. By restricting the class of functions, *e.g.* to polynomials of bounded degree, one can develop heuristics for trying to automatically find monotone polynomial interpretation functions making rules size-decrease [CL87, Luc05, CMTU05, FGM$^+$07]. Unfortunately, polynomial absolute positivity is undecidable on $\mathbb{N}$ since it is equivalent to the solvability of Diophantine equations (Proposition 6.2.11 in [TeR03]), which is undecidable [Mat70, Mat93]. Yet, these tools get useful results in practice by restricting degrees and coefficients to small values, *e.g.* 2.

A similar approach can be developed for dense sets like $\mathbb{Q}^+$ or $\mathbb{R}^+$ by ordering them with the (not well-founded!) usual orderings on $\mathbb{Q}^+$ and $\mathbb{R}^+$ if one assumes moreover that the functions $\mathsf{f}_{\mathbb{E}}$ are strictly extensive (*i.e.* $\mathsf{f}_{\mathbb{E}}(x_1, \ldots, x_n) > x_i$ for all i) [Der79a], or with the well-founded relation $<_\delta$ where, for some fixed $\delta > 0$, $x <_\delta y$ if $x + \delta \leq y$ [Luc05, FNO$^+$08]. In the case of $\mathbb{R}^+$, polynomial absolute positivity is decidable but of exponential complexity [Tar48, Col75]. Useful heuristics have however been studied [HJ98].

These approaches have also been successfully extended to linear functions on domains like $\mathbb{E} = \mathbb{B}^n$ (vectors of dimension n) or $\mathbb{E} = \mathbb{B}^{n \times n}$ (square matrices of dimension n) [EWZ08, CGP10], where $\mathbb{B}$ is a well-founded domain.

Instead of polynomial functions, Cichoń considered the class of Hardy functions [Har04] indexed by ordinals smaller than ε_0 [CT96]. The properties of Hardy functions (composition is addition of indices, etc.) can be used to reduce the search of appropriate Hardy functions to solving inequalities on ordinals.

Manna and Ness' approach has also been extended to the higher-order case.

In [Gan80b], Gandy remarks that terms of the λI-calculus (*i.e.* when, in every abstraction λxt, x freely occurs at least once in t) can be interpreted in the set of hereditary strictly monotone functions on some well-founded set $(\mathbb{E}, <_{\mathbb{E}})$, that is, a closed term of base type B is interpreted in the set $[\![\mathsf{B}]\!] = \mathbb{E}$, a closed term of type $T \Rightarrow U$ is interpreted by a monotone function from $[\![T]\!]$

to $[\![U]\!]$, and $f : [\![T \Rightarrow U]\!] <_{[\![T \Rightarrow U]\!]} g : [\![T \Rightarrow U]\!]$ if, for all $x \in [\![T]\!]$, $f(x) <_{[\![U]\!]} g(x)$ (note that, in contrast with the first-order case, x itself may be a function). Then, by taking $\mathbb{E} = \mathbb{N}$ and extending the λ-calculus with constants $\mathsf{0} : o$, $\mathsf{s} : o \Rightarrow o$ and $+ : o \Rightarrow o \Rightarrow o$ for each base type o, he defines a size function that makes β-reduction size-decrease and provide an upper bound to the number of rewrite steps. An exact upper bound was later computed by de Vrijer in [dV87].

Gandy's approach was later extended by van de Pol [vdP93, vdP96] and Kahrs [Kah95] to arbitrary higher-order rewriting *à la* Nipkow [Nip91, MN98], that is, to rewriting on terms in β-normal η-long form and higher-order pattern-matching [Mil91]. But this approach has been implemented only recently [FK12].

Interestingly, van de Pol also showed that, in the simply-typed λ-calculus, Gandy's approach can be seen as a refinement of Girard's proof of termination based on computability predicates [vdP95, vdP96].

Finally, a general categorical framework has been developed by Hamana [Ham06], that is complete wrt. the termination of binding term rewrite systems, a formalism based on Fiore, Plotkin and Turi's binding algebra [FPT99] and close to a typed version of Klop's combinatory reduction systems [KvOvR93].

To the best of our knowledge, nobody seems to have studied the relations between Howard's approach based on ordinals [How70, WW12] and Gandy's approach based on interpretations [Gan80b, dV87, vdP96].

Note also that the existence of a quasi-interpretation, *i.e.* $\|t\| \geq_{\mathbb{D}} \|u\|$ whenever $t \to u$, not only may give useful information on the complexity of a rewrite system [BMP11] but, sometimes, may also simplify the search of a termination proof. Indeed, Zantema proved in [Zan95] that the termination of a first-order rewrite system $\mathcal{R}$ is equivalent to the termination of $\mathrm{lab}(\mathcal{R}) \cup >_{\mathbb{D}}$, where $\mathrm{lab}(\mathcal{R})$ are all the variants of $\mathcal{R}$ obtained by annotating function symbols by the interpretation of their arguments, a transformation called semantic labeling. Although usually infinite, the obtained labeled system may be simpler to prove terminating, and some heuristics have been developed to use this technique in automated termination tools [MOZ96, KZ06, SM08]. This result was later extended to the higher-order case by Hamana [Ham07].

1.3 Termination based on typing with size annotations

Finally, there is another approach based on the semantics of inductive types, that has been developed for functions defined with a fixpoint combinator and pattern-matching [BQS80].

The semantics of an inductive type B, $[\![\mathsf{B}]\!]$, is usually defined, following Hessenberg's theorem [Hes09], Knaster and Tarski's theorem [KT28] or Tarski's theorem [Tar55], as the smallest fixpoint of a monotone function $\mathbb{H}^{\mathsf{B}}$ on some complete lattice. Moreover, following Kuratowski [Kur22, CC79], such a fixpoint can be reached by transfinite iteration of $\mathbb{H}^{\mathsf{B}}$ from the smallest element of the lattice $\bot$. Hence, every element $t \in [\![\mathsf{B}]\!]$ can be given as size the smallest ordinal $\mathfrak{a}$ such that $t \in \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}$, where $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}$ is the set obtained after $\mathfrak{a}$ transfinite iterations of $\mathbb{H}^{\mathsf{B}}$ from $\bot$. In particular, terms of a first-order data type like the type of

Peano integers, lists, binary trees, ... always have a size smaller than ω.

Mendler used this notion of size to prove the termination of an extension of Gödel' system T [Göd58] and Howard' system V [How72] to functionals defined by recursion on higher-order inductive types, *i.e.* types with constructors taking functions as arguments [Men87, Men91], in which case the size of a term can be bigger than ω.

In [HPS96, Par00], Hughes, Pareto and Sabry proposed to internalize this notion of size by extending the type system with, for each data type B, new type constants B_0, B_1, ... $\mathsf{B}_\infty = \mathsf{B}$ for typing the terms of type B of size smaller than or equal to 0, 1, ..., ∞ respectively, and the subtyping relation induced by the fact that a term of size at most a is also of size at most b whenever $a \leq_{\mathbb{N}} b$ or $b = \infty$. More generally, to provide some information on how a function behaves wrt. sizes, they consider as size annotations not only $0, 1, \ldots$ but any first-order term built from the function symbols $\mathsf{0}$ for zero, s for successor and $+$ for addition, and arbitrary size variables $\alpha, \beta, \ldots$, that is the language of Presburger arithmetic [Pre29]. So, for instance, the usual list constructor cons gets the type $\mathsf{N} \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{L}_{\mathsf{s}\alpha}$, and the usual map function on lists can be typed by $(\mathsf{N} \Rightarrow \mathsf{N}) \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha$, where α is a free size variable that can be instantiated by any size expression in a way similar to type instantiation in ML-like programming languages [Mil78].

Hughes, Pareto and Sabry do not actually prove the termination of their calculus but provide a domain-theoretic model [Sco72]. However, following Plotkin [Plo77], a closed term of first-order data type terminates iff its interpretation is not $\bot$. The first termination proof for arbitrary terms seems to have been given by Amadio and Coupet-Grimal in [ACG97, ACG98], who independently developed a system similar to the one of Hughes, Pareto and Sabry, inspired by Giménez' work on the use of typing annotations for termination and productivity [Gim96]. Giménez himself later proposed a similar system in [Gim98] but provided no termination proof. Note that Plotkin's result was later extended to higher-order types and rewriting-based function definitions by Berger, and Coquand and Spiwack in [Ber05, CS07, Ber08].

Size annotations are an abstraction of the semantic notion of size that one can use to prove properties on the actual size of terms like termination (size-decreasingness) or the fact that a function is not size-increasing (*e.g.* map), which can in turn be used in a termination proof [Wal88, Gie97]. Following [Cou97], it could certainly be described as an abstract interpretation.

Hence, termination can be reduced to checking that a term has some given type in the system with size-annotated type constants and subtyping induced by the ordering on size annotations, the usual typing rules being indeed valid deduction rules wrt. the size of terms (*e.g.* if $t : \mathsf{N}_a \Rightarrow \mathsf{N}_b$ and $u : \mathsf{N}_a$, then $tu : \mathsf{N}_b$).

But, in such a system, a term can have infinitely many different types because of size instantiation or because of subtyping. As already mentioned, size instantiation is similar to type instantiation in Hindley-Milner's type system [Hin69, Mil78] where the set of types of a term has a smallest element wrt. the instantiation ordering if it is not empty [Hue76]. In this case, there is a

complete type-checking algorithm for (t, T) which consists in checking that T is an instance of the smallest type of t [Hin69]. Unfortunately, with subtyping, there is no smallest type wrt. the instantiation ordering (*e.g.* λxx has type $\alpha \Rightarrow \alpha$ for all α, and type $\mathsf{B} \Rightarrow \mathsf{C}$ if $\mathsf{B} < \mathsf{C}$, but $\mathsf{B} \Rightarrow \mathsf{C}$ is not an instance of $\alpha \Rightarrow \alpha$), or subtyping composed with instantiation (*e.g.* $\lambda f \lambda x f(fx)$) has type $(\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$ for all α, and type $(\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow (\mathsf{B} \Rightarrow \mathsf{C})$ if $\mathsf{B} < \mathsf{C}$, but no instance of $(\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$ is a subtype of $(\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow (\mathsf{B} \Rightarrow \mathsf{C})$) [FM90]. To recover a notion of smallest type and completeness, all the works we know on type inference with subtyping extend the notion of type to include subtyping constraints.

We will not follow this approach though. One reason is that we consider Church-style λ-terms (*i.e.* with type-annotated abstractions) instead of Curry-style λ-terms and, in this case, as we will prove it, there is a smallest type wrt. to subtyping composed with instantiation when size expressions are only built from variables, the successor symbol and an arbitrary number of constants (the "successor" size algebra). Note moreover that, although structural (function types and base types are incomparable), subtyping is not well-founded in this case since, for instance, $\mathsf{N}_\alpha \Rightarrow \mathsf{N} > \mathsf{N}_{\mathsf{s}\alpha} \Rightarrow \mathsf{N} > \ldots$ However, if we disregard how size annotations are related to the semantics of inductive types, our work has important connections with more general extensions of Hindley-Milner's type system with subtypes [Mit84, FM90, Pot01], indexed types [Zen97], DML(C) [Xi02], HM(X) [Sul00], or generalized algebraic data types (GADTs) [XCC03, Che03], which are all a restricted form of dependent types [dB70, ML75].

Hughes, Pareto and Sabry's approach was later extended to higher-order data types [BFG$^+$04], polymorphic types [Abe04, BGP05, Abe06, Abe08], rewriting-based function definitions in the calculus of constructions [Bla04, Bla05a], conditional rewriting [BR06], product types [BGR08], and fixpoint-based function definitions in the calculus of constructions [BGP06, GS10, Sac11].

It should be noted that, in contrast with the ordinal-based approach, not all terms are given a size, but only those of base type. Moreover, although ordinals are used to define the size of terms, no ordinal is actually used in the termination criterion since one considers an abstraction of them. Indeed, when comparing two terms, one does not need to actually know their size: it is enough the difference between their size. Hence, transfinite computations can be reduced to finite ones.

Finally, Roux and the author proved in [BR09] that size annotations provide a quasi-model, and thus can be used in a semantic labeling. Terms whose type is annotated by ∞ (unknown size) are interpreted by using a technique introduced by Hirokawa and Middeldorp in [HM06]. Interestingly, semantic labeling allows one to deal with function definitions using matching on defined symbols, like in rules for associativity (*e.g.* $(x + y) + z \to x + (y + z)$), while termination criteria based on types with size annotations are restricted to matching on constructor symbols.

Current implementations of termination checkers based on typing with size annotations include ATS [Xi03, ATS], MiniAgda [Abe10, Min14], Agda [Agd16], cicminus [Sac11, Sac15] or HOT [Bla12]. Most of these tools assume given the

annotated types of function symbols (*e.g.* to know whether the size of a function is bounded by the size of one of its arguments). Heuristics for inferring the annotations of function symbols have been proposed in [TT00, CK01]. They are both based on abstract interpretation techniques [Cou96].

1.4 Contributions

The first contribution of the present paper is to give a rigorous and detailed account, for the simply-typed λ-calculus, of the approach and results sketched in [Bla04, Bla05a], hence providing the first complete account of the extension of Hughes, Pareto and Sabry's approach to rewriting-based function definitions [DJ90, TeR03].

In all the works on size-annotated types, the size algebra is fixed. In those considering first-order data types only, the size algebra is usually the language of Presburger arithmetic, the first-order theory of which is decidable [Pre29, FR74]. In those considering higher-order data types, the successor symbol $\mathtt{s}$ is usually the only symbol allowed, except in [BGR08] which allows addition too. Yet, there are various examples showing that, within a richer size algebra, more functions can be proved terminating since types are more precise.

The second contribution of the present paper is therefore to provide a type-checking algorithm for a general formulation of Hughes, Pareto and Sabry's calculus parametrized, for size annotations, by a quasi-ordered first-order term algebra $(\mathtt{A}, \leq_{\mathtt{A}})$ interpreted in ordinals. In particular, we prove that this algorithm is complete whenever size function symbols are monotone, the existential fragment of $(\mathtt{A}, \leq_{\mathtt{A}})$ is decidable and every satisfiable set of size constraints admits a smallest solution.

Third, in all the previous works, the notion of size is also fixed: the size of $t \in [\![\mathsf{B}]\!]$ is the smallest ordinal $\mathfrak{a}$ such that $t \in \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}$. When the calculus is confluent, this corresponds to the height of the tree representation of the normal form of t (an abstraction being represented as an infinite set of trees). But this notion of size is clearly incomplete wrt. termination.

The third contribution of the paper is to introduce a more general notion of size (though still incomplete) based on the stratification of the interpretation of inductive types, and to prove that one can build such a stratification in the domain of Girard's computability predicates [Gir72, GLT88], by using any monotone and extensive size function on ordinals for each inductive type constructor.

The fourth contribution is the proof that, in the successor algebra, the satisfiability of a finite set of constraints is decidable in polynomial time, and every satisfiable finite set of constraints has a smallest solution that can be computed in polynomial time too.

In contrast with [Bla04, Bla05a], the present paper:

- includes a short survey on the use of ordinals in termination proofs;
- develops a stratification-based notion of size for inhabitants of inductive types;

- introduces the notion of constructor size function;
- shows how to define a stratification from constructor size functions that are monotone and strictly extensive on recursive arguments;
- proves the existence and polynomial complexity of the computation of a smallest solution for a solvable set of constraints in the successor algebra, using max-plus algebra techniques instead of pure linear algebra techniques.

1.5 Organization of the paper

In Section 2, we define the set of terms that we consider, the relation the termination of which we are interested in, and the interpretation of types as Girard's computability predicates [GLT88] that we use to prove termination.

In Section 3, we introduce the notions of stratification, size wrt. a stratification, stratification wrt. constructor size functions, and prove general properties on the size of computable terms.

In Section 4, we introduce a general termination criterion based on a type system with size-annotated type constants and subtyping which is parametrized, for size expressions, by an arbitrary quasi-ordered first-order term algebra interpreted in ordinals. This criterion is modular: each rewrite rule must satisfy three conditions, one of them being very easy to check (accessibility), the other two being a minimality property of size annotations used to abstract the sizes of the arguments of the left-hand side, and a property of the right-hand side mixing type-checking (for subject reduction) and size-decreasingness (for termination). This local size-decreasingness property could certainly be replaced by a global termination analysis like in the dependency pair framework [AG00] or in the size-change principle [LJBA01]. Before proving the correctness of the criterion, we provide various examples of its expressive power.

In Section 5, we provide a complete algorithm for checking subject reduction and size-decreasingness, assuming that the size algebra is monotone, that the satisfiability of finite conjunctions of subtyping constraints is decidable, and that satisfiable finite conjunctions of subtyping constraints have a smallest solution wrt. subtyping composed with instantiation.

In Section 6, we show how subtyping problems can be reduced to ordering problems in the size algebra.

We then study the simplest possible algebra, the successor algebra. Although this algebra is simple, our termination criterion based on it surpasses the termination criterion currently implemented in the Coq or Matita proof assistants [Coq92, Gim94, Bou12].

In Section 7, we prove that the successor algebra fulfills the conditions described in Section 5 and, in Section 8, we provide a simple sufficient condition for the minimality condition to hold in this algebra.

2 Types, terms and computability

In this section, we define the set of terms that we consider (Church' simply-typed λ-calculus with constants [Chu40]), the operational semantics (the combination of β-reduction and user-defined rewrite rules [DJ90, TeR03]), and the notion of computability used to prove termination.

Given a set E, we denote by E^* the set of words or sequences over E (*i.e.* the free monoid containing E), the empty word by ε, the concatenation of words by juxtaposition, the length of a word w by $|w|$. We also use $\vec{e}$ to denote a (possibly empty) sequence $e_1, \ldots, e_{|\vec{e}|}$ of elements of E.

Given a partial function $f : A \to B$, $a \in A$ and $b \in B$, let $[a : b, f]$ be the function mapping a to b and every $x \in \mathrm{dom}(f) - \{a\}$ to $f(x)$.

2.1 Types

Following Church, we assume given a non-empty countable set $\mathbb{S}$ of *sorts* B, C, ... and define the set $\mathbb{T}$ of (simple) *types* as follows:

- sorts are types;
- if T and U are types, then $T \Rightarrow U$ is a type.

Implication associates to the right. So, $T \Rightarrow U \Rightarrow V$ is the same as $T \Rightarrow (U \Rightarrow V)$. Moreover, $\vec{T} \Rightarrow U$ is the same as $T_1 \Rightarrow T_2 \Rightarrow \ldots \Rightarrow T_n \Rightarrow U$ where $n = |\vec{T}|$.

The *arity* of a type T, $\mathrm{ar}(T)$, is defined as follows: $\mathrm{ar}(\mathsf{B}) = 0$ and $\mathrm{ar}(T \Rightarrow U) = 1 + \mathrm{ar}(U)$.

2.2 Terms

Given disjoint countable sets $\mathbb{V}$, $\mathbb{C}$ and $\mathbb{F}$, for variables, constructors and function symbols respectively, we define the set of *pre-terms* as follows:

- variables, constructors and function symbols are pre-terms;
- if x is a variable, T a type and u a pre-term, then $\lambda x^T u$ is a pre-term;
- if t and u are pre-terms, then tu is a pre-term.

Application associates to the left. So, tuv is the same as $(tu)v$. Moreover, $t\vec{u}$ is the same as $(\ldots((tu_1)u_2)\ldots u_{n-1})u_n$ where $n = |\vec{u}|$.

As usual, the set of *terms* $\mathbb{L}$ is obtained by quotienting pre-terms by α-equivalence, *i.e.* renaming of bound variables, assuming that $\mathbb{V}$ is infinite [CF58].

A *substitution* θ is a map from variables to terms whose *domain* $\mathrm{dom}(\theta) = \{x \in \mathbb{V} \mid \theta(x) \neq x\}$ is finite. In the following, any finite map θ from variables to terms will be implicitly extended into the substitution $\theta \cup \{(x, x) \mid x \notin \mathrm{dom}(\theta)\}$. Let $\mathrm{FV}(\theta) = \bigcup\{\mathrm{FV}(\theta(x)) \mid x \in \mathrm{dom}(\theta)\}$. The application of a substitution θ to a term t is written $t\theta$. We have $x\theta = \theta(x)$, $(tu)\theta = (t\theta)(u\theta)$ and $(\lambda x^T u)\theta = \lambda x^T (u\theta)$ if $x \notin \mathrm{dom}(\theta) \cup \mathrm{FV}(\theta)$, which can always be achieved by α-equivalence.

2.3 Typing

We assume given a map Θ assigning a type to every symbol $s \in \mathbb{C} \cup \mathbb{F}$, and will sometimes write $s : T$ instead of $(s, T) \in \Theta$ or $\Theta(s) = T$.

A *typing environment* is a finite map Γ from variables to types. The usual deduction rules assigning a type to a term in a typing environment are reminded in Figure 2. As mentioned at the beginning of the section, $[x : U, \Gamma]$ is the function mapping x to U and every $y \in \mathrm{dom}(\Gamma) - \{x\}$ to $\Gamma(y)$.

Given a symbol s, let $\mathrm{r}^s = \mathrm{ar}(\Theta(s))$ be the maximum number of terms s can be applied to. For all s, there are types $T_1, \ldots, T_{\mathrm{r}^s}$ and a sort B such that $\Theta(s) = T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^s} \Rightarrow \mathsf{B}$.

Figure 2: Typing rules

$$\frac{(s, T) \in \Theta \cup \Gamma}{\Gamma \vdash s : T} \qquad \frac{\Gamma \vdash t : U \Rightarrow V \quad \Gamma \vdash u : U}{\Gamma \vdash tu : V} \qquad \frac{[x : U, \Gamma] \vdash v : V}{\Gamma \vdash \lambda x^U v : U \Rightarrow V}$$

2.4 Rewriting

Given a relation on terms R, let $R(t) = \{t' \in \mathbb{L} \mid tRt'\}$ be the set of immediate reducts of a term t, R^* be the reflexive and transitive closure of R, and R^{-1} be its inverse ($xR^{-1}y$ if yRx). R is *finitely branching* if, for all t, $R(t)$ is finite. It is *monotone* (or congruent, stable by context, compatible with the structure of terms) if $tuRt'u$, $utRut'$ and $\lambda x^U tR\lambda x^U t'$ whenever tRt'. It is *stable* (by substitution) if $t\theta Rt'\theta$ whenever tRt'. Given two relations R and S, let RS (or $R \circ S$) be their composition ($tRSv$ if there is u such that tRu and uSv). A relation R is *locally confluent* if $R^{-1}R \subseteq R^*(R^{-1})^*$, and *confluent* if $(R^{-1})^*R^* \subseteq R^*(R^{-1})^*$.

The relation of *β-rewriting* $\rightarrow_\beta$ is the smallest monotone relation containing all the pairs $((\lambda x^U t)u, t\{(x, u)\})$.

A pair of terms (l, r), written $l \rightarrow r$, is a *rewrite rule* if there are $\mathsf{f} \in \mathbb{F}$ and $\vec{l}$ such that:

- $l = \mathsf{f}\vec{l}$;
- $|\vec{l}| \leq \mathrm{r}^{\mathsf{f}}$;
- $\mathrm{FV}(r) \subseteq \mathrm{FV}(l)$;
- $\Gamma \vdash r : T$ whenever $\Gamma \vdash l : T$.

Given a set $\mathcal{R}$ of rewrite rules, let $\rightarrow_{\mathcal{R}}$ denote the smallest monotone and stable relation containing $\mathcal{R}$. The last condition implies that $\rightarrow_{\mathcal{R}}$ preserves typing: if $\Delta \vdash t : U$ and $t \rightarrow_{\mathcal{R}} u$, then $\Delta \vdash u : U$ (subject reduction property).

It is satisfied if, for instance, l contains no abstraction and no subterm of the form $x\,t$ [BFG97].

All over the paper, we assume given a set $\mathcal{R}$ of rewrite rules and let SN be the set of terms strongly normalizing wrt.:

$$\to = \to_\beta \cup \to_{\mathcal{R}}$$

We will moreover assume that $\to$ is finitely branching, which is in particular the case if $\mathcal{R}$ is finite.

Given a relation R, let $\vec{x}\ R_{\text{prod}}\ \vec{y}$ if $|\vec{x}| = |\vec{y}|$ and there is i such that $x_i\,R\,y_i$ and, for all $j \neq i$, $x_j = y_j$. Given n relations $R_1, \ldots, R_n$, let $\vec{x}\ (R_1, \ldots, R_n)_{\text{lex}}\ \vec{y}$ if $|\vec{x}| \geq n$, $|\vec{y}| \geq n$ and there is i such that $x_i\,R_i\,y_i$ and, for all $j < i$, $x_j = y_j$. R_{prod} and $(R_1, \ldots, R_n)_{\text{lex}}$ are well-founded whenever $R, R_1, \ldots, R_n$ so are.

2.5 Computability

Following Tait [Tai67], Girard [Gir72, GLT88], Mendler [Men87], Okada [Oka89], Breazu-Tannen and Gallier [BTG89], Jouannaud and Okada [JO91, BJO02], ... termination of a rewrite relation on simply-typed λ-terms can be obtained by interpreting types by *computability predicates* and checking that function symbols are computable, that is, map computable terms to computable terms.

However, for handling matching on constructors taking functions as arguments (or matching on function symbols), one needs to modify Girard's definition of computability. In the following, we recall the definition that we will use and some of its basic properties, and refer the reader to [Bla15, Rib09] for more details on the theory of computability predicates with rewriting.

Definition 1 (Computability predicates) A term t is *neutral* if it is of the form $x\vec{v}$, $(\lambda x t)u\vec{v}$ or $\mathsf{f}\vec{t}$ with $|\vec{t}| \geq \max\{|\vec{l}| \mid \exists r, \mathsf{f}\vec{l} \to r \in \mathcal{R}\}$[1]. A *computability predicate* is a set of terms $\mathcal{S}$ satisfying the following properties:

- $\mathcal{S} \subseteq \text{SN}$;
- $\to(\mathcal{S}) \subseteq \mathcal{S}$;
- if t is neutral and $\to(t) \subseteq \mathcal{S}$, then $t \in \mathcal{S}$.

Let $\mathbb{P}$ be the set of all the computability predicates. An element of a computability predicate is said to be *computable*.

In our definition of neutral terms, not every redex is neutral as it is the case in Girard's definition. However, the following key property is preserved: application preserves neutrality: if t is neutral, then tu is neutral too. This definition also works with polymorphic and dependent types. It only excludes infinite rewrite systems where the number of arguments to which a function symbol is applied is unbounded (at the top of rule left-hand sides only, not in every term).

Computability predicates enjoy the following properties:

[1] The maximum exists since, by assumption, $|\vec{l}| \leq \mathrm{r}^{\mathsf{f}}$.

- the set $\mathbb{V}$ of variables is included in every computability predicate;
- given a computability predicate $\mathcal{S}$, $(\lambda x^U v)u \in \mathcal{S}$ iff $v\{(x,u)\} \in \mathcal{S}$ and $u \in \mathrm{SN}$;
- $\mathbb{P}$ is a complete lattice wrt. inclusion.

The greatest lower bound of a set $\mathbb{Q} \subseteq \mathbb{P}$ is $\bigcap \mathbb{Q}$ if $\mathbb{Q} \neq \emptyset$, and SN (the greatest element of $\mathbb{P}$) otherwise. Note however that the lowest upper bound of $\mathbb{Q}$, written $\mathrm{lub}(\mathbb{Q})$, is not necessarily the union. For instance, with the non-confluent rewrite system $\mathcal{R} = \{\mathsf{f} \to \mathsf{a}, \mathsf{f} \to \mathsf{b}\}$, if $\mathbb{P}(\mathcal{X})$ denotes the smallest computability predicate containing $\mathcal{X}$, then $\mathbb{P}(\{\mathsf{a}\}) \cup \mathbb{P}(\{\mathsf{b}\})$ is not a computability predicate since it does not contain f. There are a number of cases where the union of two computability predicates is known to be a computability predicate, but this is for a different notion of neutral term:

- In [Rib07, Rib08], Riba proves that his set of computability predicates is stable by union if $\mathcal{R}$ is an orthogonal constructor rewrite system.
- In [Wer94] (Lemma 4.14 p. 96), Werner proves that his set of computability predicates is stable by well-ordered union.

Luckily, Werner's proof does not depend on the definition of neutral terms:

Lemma 1 If $\to$ is finitely branching and $\mathbb{Q}$ is a set of computability predicates well-ordered wrt. inclusion, then $\bigcup \mathbb{Q}$ is a computability predicate.

Proof.

- Let $t \in \bigcup \mathbb{Q}$. Then, there is $\mathcal{S} \in \mathbb{Q}$ such that $t \in \mathcal{S}$. Since $\mathcal{S} \subseteq \mathrm{SN}$, we have $t \in \mathrm{SN}$.
- Let $t \in \bigcup \mathbb{Q}$ and u such that $t \to u$. Then, there is $\mathcal{S} \in \mathbb{Q}$ such that $t \in \mathcal{S}$. Since $\to(\mathcal{S}) \subseteq \mathcal{S}$, we have $u \in \mathcal{S}$ and thus $u \in \mathbb{Q}$.
- Let t be a neutral term such that $\to(t) \subseteq \mathbb{Q}$. If $\to(t) = \emptyset$, then t belongs to every element of $\mathbb{Q}$. Therefore, $t \in \mathbb{Q}$. Otherwise, since $\to$ is finitely branching, we have $\to(t) = \{t_1, \ldots, t_n\}$ with $n \geq 1$. For every $i \in \{1, \ldots, n\}$, there is $\mathcal{S}_i \in \mathbb{Q}$ such that $t_i \in \mathcal{S}_i$. Since $\mathbb{Q}$ is well-ordered wrt. inclusion, there is $k \in \{1, \ldots, n\}$ such that $\mathcal{S}_k$ is the biggest element of $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ wrt. inclusion. Hence, $\to(t) \subseteq \mathcal{S}_k$ and $t \in \mathcal{S}_k$. Therefore, $t \in \mathbb{Q}$. ■

The interpretation of arrow types is defined as usual, in order to ensure the termination of β-reduction:

Definition 2 (Interpretation of arrow types) A (partial) interpretation of sorts, that is, a (partial) function $\mathbb{I} : \mathbb{S} \to \wp(\mathbb{L})$ (powerset of $\mathbb{L}$), is extended into a (partial) interpretation of types $\overline{\mathbb{I}} : \mathbb{T} \to \wp(\mathbb{L})$ as follows:

- $\overline{\mathbb{I}}(\mathsf{B}) = \mathbb{I}(\mathsf{B})$;
- $\overline{\mathbb{I}}(U \Rightarrow V) = \overline{\mathbb{I}}(U) \Rightarrow \overline{\mathbb{I}}(V)$ where $\mathcal{U} \Rightarrow \mathcal{V} = \{t \in \mathbb{L} \mid \forall u \in \mathcal{U}, tu \in \mathcal{V}\}$.

Note that $\overline{\mathbb{I}}(T)$ is defined whenever $\mathbb{I}$ is defined on every sort occurring in T, and $\overline{\mathbb{I}}(T) = \overline{\mathbb{J}}(T)$ whenever $\mathbb{I}$ and $\mathbb{J}$ are defined and equal on every sort occurring in T.

Note also that $\mathcal{U} \Rightarrow \mathcal{V}$ is a computability predicate whenever $\mathcal{U}$ and $\mathcal{V}$ so are. Hence, $\overline{\mathbb{I}}(T)$ is a computability predicate whenever $\mathbb{I}(\mathsf{B})$ so is for every sort B occurring in T.

For interpreting sorts, one could take the computability predicate SN. But this interpretation does not allow one to prove the computability of functions defined by induction on types with constructors taking functions as arguments.

Moreover, a computable term may have non-computable subterms. Consider for instance $\mathsf{c} : (\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow \mathsf{B}$, $\mathsf{f} : \mathsf{B} \Rightarrow (\mathsf{B} \Rightarrow \mathsf{C})$, $\mathcal{R} = \{\mathsf{f}(\mathsf{c}\ x) \to x\}$ and $t = \lambda x^{\mathsf{B}} \mathsf{f} x x$. Then, assuming that $\mathbb{I}(\mathsf{B}) = \mathrm{SN}$, we have $(\mathsf{c}\ t) \in \mathbb{I}(\mathsf{B})$, but $t \notin \mathbb{I}(\mathsf{B}) \Rightarrow \mathbb{I}(\mathsf{C})$ since $t(\mathsf{c}\ t) \notin \mathrm{SN}$ because $t(\mathsf{c}\ t) \to_\beta \mathsf{f}(\mathsf{c}\ t)(\mathsf{c}\ t) \to_{\mathcal{R}} t(\mathsf{c}\ t)$. It is however possible to enforce that a direct subterm of type T of a computable term of sort B is computable if B occurs in T at positive positions only [Men87]:

Definition 3 (Positive and negative positions) The set of *positions* $\mathrm{Pos}(\mathsf{B}, T) \subseteq \{1, 2\}^*$ of a sort B in a type T is defined as follows:

- $\mathrm{Pos}(\mathsf{B}, \mathsf{C}) = \{\varepsilon\}$ if $\mathsf{B} = \mathsf{C}$;
- $\mathrm{Pos}(\mathsf{B}, \mathsf{C}) = \emptyset$ if $\mathsf{B} \neq \mathsf{C}$;
- $\mathrm{Pos}(\mathsf{B}, U \Rightarrow V) = \{1p \mid p \in \mathrm{Pos}(\mathsf{B}, U)\} \cup \{2p \mid p \in \mathrm{Pos}(\mathsf{B}, V)\}$.

The sets of *positive* and *negative positions* in a type T, written $\mathrm{Pos}^+(T)$ and $\mathrm{Pos}^-(T)$ respectively, are mutually defined as follows:

- $\mathrm{Pos}^+(\mathsf{B}) = \{\varepsilon\}$;
- $\mathrm{Pos}^-(\mathsf{B}) = \emptyset$;
- $\mathrm{Pos}^+(U \Rightarrow V) = \{1p \mid p \in \mathrm{Pos}^-(U)\} \cup \{2p \mid p \in \mathrm{Pos}^+(V)\}$;
- $\mathrm{Pos}^-(U \Rightarrow V) = \{1p \mid p \in \mathrm{Pos}^+(U)\} \cup \{2p \mid p \in \mathrm{Pos}^-(V)\}$.

Note that the positive and negative positions of a type are disjoint sets. However, a sort can have both positive and negative occurrences. For instance, $\mathrm{Pos}^+(\mathsf{T}, \mathsf{T} \Rightarrow \mathsf{T}) = \{2\}$ and $\mathrm{Pos}^-(\mathsf{T}, \mathsf{T} \Rightarrow \mathsf{T}) = \{1\}$.

Definition 4 (Accessible arguments) We assume given a well-founded *ordering* on sorts $<_{\mathbb{S}}$. The i-th argument of a constructor $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$ is:

- *recursive* if $\mathrm{Pos}(\mathsf{B}, T_i) \neq \emptyset$;
- *accessible* if T_i is *positive wrt.* B, that is:
  - every sort occurring in T_i is smaller than or equal to B:
    for all C, $\mathrm{Pos}(\mathsf{C}, T) = \emptyset$ or $\mathsf{C} \leq_{\mathbb{S}} \mathsf{B}$, where $\leq_{\mathbb{S}}$ is the reflexive closure of $<_{\mathbb{S}}$;
  - B occurs only positively in T_i: $\mathrm{Pos}(\mathsf{B}, T_i) \subseteq \mathrm{Pos}^+(T_i)$.

In the following, we will assume wlog[2] that there are $0 \leq \mathrm{p}^{\mathsf{c}} \leq \mathrm{q}^{\mathsf{c}}$ such that:

- for all $i \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, the i-th argument of c is recursive;
- for all $i \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, the i-th argument of c is accessible.

$$\Theta(\mathsf{c}) = \underbrace{T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{p}^{\mathsf{c}}}}_{\text{rec. acc. args}} \Rightarrow \underbrace{T_{\mathrm{p}^{\mathsf{c}}+1} \Rightarrow \ldots \Rightarrow T_{\mathrm{q}^{\mathsf{c}}}}_{\text{non-rec. acc. args}} \Rightarrow \underbrace{T_{\mathrm{q}^{\mathsf{c}}+1} \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^{\mathsf{c}}}}_{\text{non-acc. args}} \Rightarrow \mathsf{B}$$

For instance, for the sort N of natural numbers with the constructors $\mathsf{0} : \mathsf{N}$ and $\mathsf{s} : \mathsf{N} \Rightarrow \mathsf{N}$ (successor) [Pea89], we can take $\mathrm{p}^{\mathsf{0}} = \mathrm{q}^{\mathsf{0}} = 0$ and $\mathrm{p}^{\mathsf{s}} = \mathrm{q}^{\mathsf{s}} = 1$ since N occurs only positively in N. Similarly, for the sort O of Howard's constructive ordinals with the constructors $\mathsf{zero} : \mathsf{O}$, $\mathsf{succ} : \mathsf{O} \Rightarrow \mathsf{O}$ (successor) and $\mathsf{lim} : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow \mathsf{O}$ (limit) [How72], we can take $\mathrm{p}^{\mathsf{zero}} = \mathrm{q}^{\mathsf{zero}} = 0$, $\mathrm{p}^{\mathsf{succ}} = \mathrm{q}^{\mathsf{succ}} = 1$ since O occurs only positively in O, and $\mathrm{p}^{\mathsf{lim}} = \mathrm{q}^{\mathsf{lim}} = 1$ since O occurs only positively in $\mathsf{N} \Rightarrow \mathsf{O}$ if one takes $\mathsf{N} <_{\mathbb{S}} \mathsf{O}$.

Non-accessible arguments are usually forbidden by requiring all the arguments to be positive, or even strictly positive[3] as it is the case in the Coq the proof assistant [CPM88]. Here, we do not forbid non-positive arguments and do not require arguments to be strictly positive (Example 7 uses a non-strictly positive sort). Hence, one can have a sort D with the constructors $\mathsf{app} : \mathsf{D} \Rightarrow \mathsf{D} \Rightarrow \mathsf{D}$ and $\mathsf{lam} : (\mathsf{D} \Rightarrow \mathsf{D}) \Rightarrow \mathsf{D}$, in which case we must have $\mathrm{p}^{\mathsf{lam}} = 0$ since the first argument of lam is not positive. However, the termination conditions will enforce that, although one can use in a rule left-hand side ($\mathsf{lam}\ x$) as a pattern, x cannot be used in the corresponding rule right-hand side: in a rule, constructors with non-positive arguments can be pattern-matched in the left-hand side, but only their positive arguments can be used in the right-hand side.

For the sake of simplicity, we consider an ordering instead of a quasi-ordering, although a quasi-ordering might a priori be necessary for dealing with mutually defined inductive types (*e.g.* the types of trees and forests with the constructors $\mathsf{empty} : \mathsf{F}$, $\mathsf{add} : \mathsf{F} \Rightarrow \mathsf{T} \Rightarrow \mathsf{F}$ and $\mathsf{node} : \mathsf{F} \Rightarrow \mathsf{T}$). The results described in this paper can however still be applied if one identifies mutually defined inductive types, because a term typable with mutually defined inductive types is a fortiori typable in the type system where they are identified. This abstraction is correct but not necessarily complete since more terms get typable when two typed are identified (*e.g.* $\mathsf{add}\ \mathsf{empty}\ \mathsf{empty}$ is typable if $\mathsf{T} = \mathsf{F}$).

Since $<_{\mathbb{S}}$ is well-founded, we can can define an interpretation $\mathbb{I}$ for every sort by well-founded induction on it as follows. Let B be a sort and assume that $\mathbb{I}$ is defined for every sort smaller than B. Then, let $\mathbb{I}(\mathsf{B})$ be the least fixpoint of the monotone function $\mathbb{H}^{\mathsf{B}}$ on the complete lattice $\wp(\mathbb{L})$ such that:

$$\mathbb{H}^{\mathsf{B}}(\mathcal{X}) = \{t \in \mathrm{SN} \mid \forall (\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\rightarrow^*}(t), \forall k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}, t_k \in \overline{[\mathsf{B} : \mathcal{X}, \mathbb{I}]}(T_k)\}$$

[2] Arguments can be permuted if needed.

[3] The i-th argument of c is *strictly positive* if $\mathrm{Pos}(\mathsf{B}, T_i) = \emptyset$, or $T_i = \vec{U} \Rightarrow \mathsf{B}$ and $\mathrm{Pos}(\mathsf{B}, \vec{U}) = \emptyset$.

where $\mathbb{C}^{\mathsf{B}}_{\to^*}(t) = \{(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}} \mid t \to^* \mathsf{c}\,\vec{t}\}$, $\mathbb{C}^{\mathsf{B}} = \{(\mathsf{c}, \vec{t}, \vec{T}) \mid \mathsf{c} : \vec{T} \Rightarrow \mathsf{B}, |\vec{t}| = |\vec{T}|\}$ and $[\mathsf{B} : \mathcal{X}, \mathbb{I}]$ is the function mapping B to $\mathcal{X}$ and every $\mathsf{C} \in \mathrm{dom}(\mathbb{I}) - \{\mathsf{B}\}$ to $\mathbb{I}(\mathsf{C})$.

That such a least fixpoint exists follows from Knaster and Tarski's fixpoint theorem [KT28, Tar55] and the following fact:

Proposition 1 ([Bla05b]) Let B be a sort, $\mathbb{I}$ be an interpretation for every sort smaller than B, and T be a type positive wrt. B. Then, $\overline{[\mathsf{B} : \mathcal{X}, \mathbb{I}]}(T)$ is monotone wrt. $\mathcal{X}$.

Moreover, one can easily check that $\mathbb{H}^{\mathsf{B}}(\mathcal{X})$ is a computability predicate whenever $\mathcal{X}$ so is. Hence, for every type T, $\overline{\mathbb{I}}(T)$ is a computability predicate.

In the following, for the sake of simplicity, we will not mention $\mathbb{I}$ anymore and simply write $t \in T$ instead of $t \in \overline{\mathbb{I}}(T)$, and $t \in [\mathsf{B} : \mathcal{X}]T$ instead of $t \in \overline{[\mathsf{B} : \mathcal{X}, \mathbb{I}]}(T)$.

3 Size of computable terms

In this section, we study a general way of attributing an ordinal size to computable terms of base type by defining, for each sort, a stratification of computable terms of this sort using a size function for each constructor, and assuming that $\to$ is finitely branching.

By Hartogs' theorem [Har15], there is an ordinal the elements of which cannot be injected into $\wp(\mathbb{L})$, where $\mathbb{L}$ is the set of terms (note that this theorem does not require the axiom of choice). Let $\mathfrak{h}$ be the smallest such ordinal. Since $\mathbb{V}$ is countably infinite and $\mathbb{C}$ and $\mathbb{F}$ are countable, $\mathfrak{h}$ is the successor cardinal of $|\wp(\mathbb{L})| = 2^{\omega}$ [HJ99].

3.1 Stratifications

Definition 5 (Stratification of a sort) Given a family $(\mathcal{S}_{\mathfrak{a}})_{\mathfrak{a}<\mathfrak{h}}$ of computability predicates, let $\mathcal{S}_{\mathfrak{h}} = \mathrm{lub}\{\mathcal{S}_{\mathfrak{a}} \mid \mathfrak{b} < \mathfrak{a}\}$.

A *stratification* of a computability predicate $\mathcal{S}$ is a monotone sequence of computability predicates $(\mathcal{S}_{\mathfrak{a}})_{\mathfrak{a}<\mathfrak{h}}$ included in $\mathcal{S}$ and converging to $\mathcal{S}$, that is, such that $\mathcal{S}_{\mathfrak{h}} = \mathcal{S}$.

A stratification of a type T is a stratification of $\overline{\mathbb{I}}(T)$.

Given a stratification $\mathcal{S}$, the *size* of an element $t \in \mathcal{S}_{\mathfrak{h}}$, written $o_{\mathcal{S}}(t)$, is the smallest ordinal $\mathfrak{a} < \mathfrak{h}$ such that $t \in \mathcal{S}_{\mathfrak{a}}$.

A stratification is *continuous* if, for all limit ordinals $0 < \mathfrak{a} < \mathfrak{h}$, $\mathcal{S}_{\mathfrak{a}} = \mathrm{lub}(\{\mathcal{S}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$.

Since $\mathcal{S}$ is monotone, for all $\mathfrak{a} \leq \mathfrak{h}$, $\{\mathcal{S}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\}$ is well-ordered wrt. inclusion. Hence, since $\to$ is finitely branching, by Lemma 1, $\mathcal{S}_{\mathfrak{h}} = \bigcup(\{\mathcal{S}_{\mathfrak{a}} \mid \mathfrak{b} < \mathfrak{a}\})$. Moreover, a stratification is *continuous* iff, for all infinite limit ordinals $\mathfrak{a} < \mathfrak{h}$, $\mathcal{S}_{\mathfrak{a}} = \bigcup(\{\mathcal{S}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$.

We now prove some properties of $o_{\mathcal{S}}(t)$:

Lemma 2 Let $\mathcal{S}$ be a stratification and $t \in \mathcal{S}_{\mathfrak{h}}$.

1. If $t \to t'$, then $t' \in \mathcal{S}_{\mathfrak{h}}$ and $o_{\mathcal{S}}(t) \geq o_{\mathcal{S}}(t')$.
2. If $\mathcal{S}$ is continuous, then either $o_{\mathcal{S}}(t) = 0$ or $o_{\mathcal{S}}(t) = \mathfrak{b} + 1$ for some ordinal $\mathfrak{b}$.

Proof.

1. Since $\mathcal{S}_{o_{\mathcal{S}}(t)}$ is stable by reduction, $t' \in \mathcal{S}_{o_{\mathcal{S}}(t)}$. Therefore, $o_{\mathcal{S}}(t') \leq o_{\mathcal{S}}(t)$.
2. Assume that $o_{\mathcal{S}}(t)$ is a limit ordinal $\mathfrak{a} > 0$. Since $\mathcal{S}$ is continuous, we have $\mathcal{S}_{\mathfrak{a}} = \bigcup(\{\mathcal{S}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$. Therefore, $t \in \mathcal{S}_{\mathfrak{b}}$ for some $\mathfrak{b} < \mathfrak{a}$. Contradiction. ■

By Proposition 1, $[\mathsf{B} : \mathcal{X}](T)$ is monotone wrt. $\mathcal{X}$ whenever T is positive wrt. B. Hence, any stratification $\mathcal{S}$ of B provides a way to define a stratification of T:

Definition 6 (Stratification of a positive type) Given a stratification $\mathcal{S}$ of a sort B and a type T positive wrt. B, let $[\mathsf{B} : \mathcal{S}](T)$ denote the stratification $\mathcal{T}$ of T obtained by taking $\mathcal{T}_{\mathfrak{a}} = [\mathsf{B} : \mathcal{S}_{\mathfrak{a}}](T)$.

Note that $[\mathsf{B} : \mathcal{S}]T$ is not continuous in general (see Example 1).

Lemma 3 If $\mathcal{S}$ is a stratification of B, $v \in \vec{U} \Rightarrow \mathsf{B}$ and $\mathrm{Pos}(\mathsf{B}, \vec{U}) = \emptyset$, then $o_{[\mathsf{B}:\mathcal{S}](\vec{U} \Rightarrow \mathsf{B})}(v) = \sup\{o_{\mathcal{S}}(v\vec{u}) \mid \vec{u} \in \vec{U}\}$.

Proof. Let $\mathfrak{a} = o_{[\mathsf{B}:\mathcal{S}](\vec{U} \Rightarrow \mathsf{B})}(v)$ and $\mathfrak{b} = \sup\{o_{\mathcal{S}}(v\vec{u}) \mid \vec{u} \in \vec{U}\}$. By definition of $\mathfrak{a}$, we have $v \in \vec{U} \Rightarrow \mathcal{S}_{\mathfrak{a}}$. So, for all $\vec{u} \in \vec{U}$, $v\vec{u} \in \mathcal{S}_{\mathfrak{a}}$ and $o_{\mathcal{S}}(v\vec{u}) \leq \mathfrak{a}$. Thus, $\mathfrak{b} \leq \mathfrak{a}$. We now prove that $v \in \vec{U} \Rightarrow \mathcal{S}_{\mathfrak{b}}$. Let $\vec{u} \in \vec{U}$. By definition of $\mathfrak{b}$, $o_{\mathcal{S}}(v\vec{u}) \leq \mathfrak{b}$. So, $v\vec{u} \in \mathcal{S}_{\mathfrak{b}}$. ■

A continuous stratification of a sort B can be obtained by the transfinite iteration of $\mathbb{H}^{\mathsf{B}}$ from the smallest computability predicate $\bot$ [Kur22, CC79]:

- $\mathcal{D}^{\mathsf{B}}_0 = \bot$;
- $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}+1} = \mathbb{H}^{\mathsf{B}}(\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}})$;
- $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}(\{\mathcal{D}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$ if $\mathfrak{a}$ is an infinite limit ordinal.

The fact that $\mathcal{D}^{\mathsf{B}}$ is monotone follows from the facts that $\mathcal{D}^{\mathsf{B}}_0 \subseteq \mathcal{D}^{\mathsf{B}}_1$ and $\mathbb{H}^{\mathsf{B}}$ is monotone [CC79]. Now, by definition of $\mathfrak{h}$, $\mathcal{D}^{\mathsf{B}}$ is not injective. Therefore, there are $\mathfrak{c} < \mathfrak{d} < \mathfrak{h}$ such that $\mathcal{D}^{\mathsf{B}}_{\mathfrak{c}} = \mathcal{D}^{\mathsf{B}}_{\mathfrak{d}}$. Since $\mathcal{D}^{\mathsf{B}}$ is monotone, $\mathcal{D}^{\mathsf{B}}_{\mathfrak{c}+1} = \mathcal{D}^{\mathsf{B}}_{\mathfrak{c}}$ and $\mathcal{D}^{\mathsf{B}}_{\mathfrak{h}} = \mathsf{B}$ [RR63].

We call this stratification the *default* stratification. It is the one used in all the previous works on sized types, except in [Abe12] where, after [SD03], Abel uses a stratification having better properties, namely $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}(\{\mathbb{H}^{\mathsf{B}}(\mathcal{S}^{\mathsf{B}}_{\mathfrak{b}}) \mid \mathfrak{b} < \mathfrak{a}\})$.

The default stratification measures the height. If no constructor of B has accessible *functional* arguments and $\to$ is finitely branching, then every element of B has a size smaller than ω. Hence, when considering first-order data types only (*e.g.* natural numbers, lists, binary trees) and a finitely branching rewrite relation $\to$, one can in fact take $\mathfrak{h} = \omega$.

On the other hand, when one wants to consider constructors with accessible functional arguments, then one can get terms of size bigger than ω:

Example 1 Take the sort O of Howard's constructive ordinals mentioned in the previous section and let $\mathsf{inj} : \mathsf{N} \Rightarrow \mathsf{O}$ be the usual injection from N to O defined by the rules $\mathsf{inj}\ \mathsf{0} \to \mathsf{zero}$ and $\mathsf{inj}\ (\mathsf{s}\ x) \to \mathsf{succ}\ (\mathsf{inj}\ x)$. Then, $o_{\mathcal{D}^\circ}(\mathsf{lim}\ \mathsf{inj}) = \omega + 1$. Indeed, $o_{\mathcal{D}^\circ}(\mathsf{lim}\ \mathsf{inj})$ is the smallest ordinal $\mathfrak{a}$ such that $\mathsf{lim}\ \mathsf{inj} \in \mathcal{D}^{\mathsf{O}}_{\mathfrak{a}}$. By definition of $\mathcal{D}$, $\mathfrak{a} = \mathfrak{b} + 1$ where $\mathfrak{b}$ is the smallest ordinal such that $\mathsf{inj} \in \mathsf{N} \Rightarrow \mathcal{D}^{\mathsf{O}}_{\mathfrak{b}}$. By Lemma 3, $\mathfrak{b} = \sup\{o_{\mathcal{D}^\circ}(\mathsf{inj}\ t) \mid t \in \mathsf{N}\}$. Now, a term of the form $(\mathsf{inj}\ t)$ can only reduce to a term of the form $(\mathsf{inj}\ u)$, zero or $(\mathsf{succ}\ u)$. Hence, $o_{\mathcal{D}^\circ}(\mathsf{inj}\ t) < \omega$. Finally, one can easily prove that, for all $n < \omega$, $o_{\mathcal{D}^\circ}(\mathsf{inj}(\mathsf{s}^n\mathsf{0})) = n+1$. Therefore, $\mathfrak{b} = \omega$. ■

One can also get terms of size bigger than ω by considering infinitely branching and non-confluent rewrite relations. For instance, with $\mathcal{R} = \{\mathsf{f} \to \mathsf{s}^i\mathsf{0} \mid i \in \mathbb{N}\}$, one gets $o_{\mathcal{D}^{\mathsf{N}}}(\mathsf{f}) = \omega + 1$.

3.2 Stratifications based on size functions

We now introduce a general way of defining a stratification:

Definition 7 (Size function) A *size function* for $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$ is given by:

- a function $\Sigma^{\mathsf{c}} : \mathfrak{h}^{\mathrm{q}^{\mathsf{c}}} \to \mathfrak{h}$ for computing the size of a term of the form $\mathsf{c}\,\vec{t}$ from the sizes of its accessible arguments;
- for every non-recursive accessible argument $k \in \{\mathrm{p}^{\mathsf{c}}+1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, a sort $\Sigma^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$ occurring in T_k, only positively, and with respect to which we will measure the size of the k-th argument of c (in the following, we let $\Sigma^{\mathsf{c}}_k = \mathsf{B}$ if $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$).

For instance, consider the type T of labeled binary trees with the constructors $\mathsf{leaf} : \mathsf{B} \Rightarrow \mathsf{T}$ and $\mathsf{node} : \mathsf{T} \Rightarrow \mathsf{T} \Rightarrow \mathsf{B} \Rightarrow \mathsf{T}$, where $\mathsf{B} <_{\mathbb{S}} \mathsf{T}$ is a sort for labels. We can take $\mathrm{p}^{\mathsf{leaf}} = 0$, $\mathrm{q}^{\mathsf{leaf}} = 1$, $\mathrm{p}^{\mathsf{node}} = 2$, $\mathrm{q}^{\mathsf{node}} = 3$, $\Sigma^{\mathsf{leaf}}(\mathfrak{a}) = 0$ and $\Sigma^{\mathsf{node}}(\mathfrak{a}, \mathfrak{b}, \mathfrak{c}) = \mathfrak{a} + \mathfrak{b} + 1$, so that the size of a tree is not its height as in the default stratification but the number of its nodes.

Note that Σ^{c} may depend on all accessible arguments, including the non-recursive ones. That is why we need a sort $\Sigma^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$ wrt. which the size of the k-th argument will be measured. For instance, one can measure the size of a pair of natural numbers by the sum of their sizes: given a type P for pairs of natural numbers with the constructor $\mathsf{pair} : \mathsf{N} \Rightarrow \mathsf{N} \Rightarrow \mathsf{P}$, one can take $\mathrm{p}^{\mathsf{pair}} = 0$, $\mathrm{q}^{\mathsf{pair}} = 2$, $\Sigma^{\mathsf{pair}}_1 = \Sigma^{\mathsf{pair}}_2 = \mathsf{N}$ and $\Sigma^{\mathsf{pair}}(\mathfrak{a}, \mathfrak{b}) = \mathfrak{a} + \mathfrak{b}$.

Finally, the two can be mixed so that, for instance, the size of a list of natural numbers can be defined as the sum of the sizes of its components. With this notion of size, a list with only one big element can be greater than a list with many small elements.

Definition 8 (Stratification defined by a size function) Assume that $\to$ is finitely branching. Given a size function Σ^{c} for every constructor c, we define a continuous stratification $\mathcal{S}^{\mathsf{B}}$ for every sort B by induction on $>_{\mathbb{S}}$ as follows, where, given $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$, $o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})$ denotes the sequence $o_{\mathcal{S}^{\mathsf{c},1}}(t_1)$, $\ldots$, $o_{\mathcal{S}^{\mathsf{c},n}}(t_n)$ with $n = \mathrm{q}^{\mathsf{c}}$ and $\mathcal{S}^{\mathsf{c},k} = [\Sigma^{\mathsf{c}}_k : \mathcal{S}^{\Sigma^{\mathsf{c}}_k}]T_k$, that is, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k)$ is the size of t_k in T_k wrt. Σ^{c}_k (which is B if $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$):

- $\mathcal{S}^{\mathsf{B}}_0$ is the set of terms $t \in \mathrm{SN}$ such that, for all $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$:
  - $\mathrm{p}^{\mathsf{c}} = 0$ (*i.e.* c has no recursive argument),
  - $\forall k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, $t_k \in T_k$,
  - $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq 0$.
- $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}+1}$ is the set of terms $t \in \mathrm{SN}$ such that, for all $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$:
  - $\forall k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$
  - $\forall k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, $t_k \in T_k$
  - $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a} + 1$.
- $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}(\{\mathcal{S}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$ if $\mathfrak{a}$ is an infinite limit ordinal.

Note that $\mathcal{S}$ is well-defined because:

- in the case of $\mathcal{S}^{\mathsf{B}}_0$:
  - $\mathrm{p}^{\mathsf{c}} = 0$ and thus, for all $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) = o_{[\Sigma^{\mathsf{c}}_k : \mathcal{S}^{\Sigma^{\mathsf{c}}_k}]T_k}(t_k)$ is well-defined since $t_k \in T_k$ and $\Sigma^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$.
- in the case of $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}+1}$:
  - $\forall k \leq \mathrm{p}^{\mathsf{c}}$, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) = o_{[\mathsf{B}:\mathcal{S}^{\mathsf{B}}]T_k}(t_k)$ is well-defined and $\leq \mathfrak{a}$ since $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$;
  - $\forall k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k)$ is well-defined since $t_k \in T_k$ and $\Sigma^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$.

The definition of $\mathcal{S}^{\mathsf{B}}$ is similar to the definition of the default stratification except that the size functions Σ^{c} are used to enforce lower bounds on the size of terms. Hence, if one takes for every Σ^{c} the constant function equal to 0, then one almost gets the default stratification. To get the default stratification one has to slightly change the definition of $\mathcal{S}^{\mathsf{B}}$ by taking $\mathcal{S}^{\mathsf{B}}_0 = \bot$. The current definition has the advantage that both variables and nullary constructors have size 0 so that, for instance, if one takes $\Sigma_{\mathsf{0}} = \Sigma_{\mathsf{s}}(\mathfrak{a}) = 0$, then $o_{\mathcal{S}^{\mathsf{N}}}(\mathsf{s}^i x) = o_{\mathcal{S}^{\mathsf{N}}}(\mathsf{s}^i \mathsf{0}) = i$ while, in the default stratification, $o_{\mathcal{D}^{\mathsf{N}}}(\mathsf{s}^i x) = i$ and $o_{\mathcal{D}^{\mathsf{N}}}(\mathsf{s}^i \mathsf{0}) = i + 1$ (nullary constructors do not belong to $\bot$).

We now check that $\mathcal{S}^{\mathsf{B}}$ is indeed a stratification of B.

Lemma 4 For every sort B and ordinal $\mathfrak{a} < \mathfrak{h}$, $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B}$.

Proof. We proceed by induction on $<_{\mathbb{S}}$ and $\mathfrak{a}$.

- Let $t \in \mathcal{S}^{\mathsf{B}}_{0}$. Then, $t \in \mathrm{SN}$. Let now $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$. Then, $\mathrm{p}^{\mathsf{c}} = 0$ and $t_k \in T_k$. Hence, $t \in \mathsf{B}$ since $\mathsf{B} = \mathbb{H}^{\mathsf{B}}(\mathsf{B})$.

- Let $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}+1}$. Then, $t \in \mathrm{SN}$. Let now $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$. If $k \leq \mathrm{p}^{\mathsf{c}}$, then $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$. By induction hypothesis, $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B}$. Since B occurs only positively in T_k, Proposition 1 gives $[\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k \subseteq [\mathsf{B} : \mathsf{B}]T_k = T_k$. Therefore, $t_k \in T_k$. Now, if $k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, then $t_k \in T_k$ too. Therefore, $t \in \mathsf{B}$ since $\mathsf{B} = \mathbb{H}^{\mathsf{B}}(\mathsf{B})$.

- Let $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}$ with $\mathfrak{a}$ a limit ordinal. Then, $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{b}}$ for some $\mathfrak{b} < \mathfrak{a}$. Therefore, by induction hypothesis, $t \in \mathsf{B}$. ■

Lemma 5 For every sort B, $\mathcal{S}^{\mathsf{B}}$ is monotone.

Proof. We prove that, for all $\mathfrak{a}$, for all $\mathfrak{b}$, if $\mathfrak{b} < \mathfrak{a}$, then $\mathcal{S}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}$, by induction on $\mathfrak{a}$ (1). If $\mathfrak{a}$ is a limit ordinal, then this is immediate. Otherwise, $\mathfrak{a} = \mathfrak{a}' + 1$ and $\mathfrak{b} \leq \mathfrak{a}'$. If $\mathfrak{b} < \mathfrak{a}'$ then, by induction hypothesis (1), $\mathcal{S}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}'}$. Otherwise, $\mathfrak{b} = \mathfrak{a}'$. Hence, in both cases, we need to have $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}'} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}'+1}$ to conclude. To this end, we prove that, for all $\mathfrak{c} < \mathfrak{a}$, $\mathcal{S}^{\mathsf{B}}_{\mathfrak{c}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}+1}$, by induction on $\mathfrak{c}$ (2).

- Let $t \in \mathcal{S}^{\mathsf{B}}_{0}$. Then, $t \in \mathrm{SN}$. Let now $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$. Then, $\mathrm{p}^{\mathsf{c}} = 0$; for all $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, $t_k \in T_k$; and $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq 0 \leq 1$. Therefore, $t \in \mathcal{S}^{\mathsf{B}}_{1}$.

- Let $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}+1}$. Then, $t \in \mathrm{SN}$. Let now $\mathsf{c}\,\vec{t} \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$. Then $o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}) \leq \mathfrak{c} \leq \mathfrak{c} + 1$; if $k \leq \mathrm{p}^{\mathsf{c}}$, then $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k$; and if $k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, then $t_k \in T_k$. Assume that $k \leq \mathrm{p}^{\mathsf{c}}$. By induction hypothesis (2), $\mathcal{S}^{\mathsf{B}}_{\mathfrak{c}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}+1}$. Since B occurs only positively in T_k, $[\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k \subseteq [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}+1}]T_k$. Therefore, $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}+2}$.

- Let $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}$ with $\mathfrak{c}$ an infinite limit ordinal. Then, $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}}$ for some $\mathfrak{d} < \mathfrak{c}$. Thus $t \in \mathrm{SN}$ and, by induction hypothesis (2), $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}+1}$. Let now $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$. Then, $o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}) \leq \mathfrak{d} + 1 < \mathfrak{c} + 1$; if $k \leq \mathrm{p}^{\mathsf{c}}$, then $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}}]T_k$; and if $k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, then $t_k \in T_k$. Assume that $k \leq \mathrm{p}^{\mathsf{c}}$. Since $\mathfrak{d} < \mathfrak{c} < \mathfrak{a}$, by induction hypothesis (1), $\mathcal{S}^{\mathsf{B}}_{\mathfrak{d}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}$. Since B occurs only positively in T_k, $[\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}}]T_k \subseteq [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k$. Therefore, $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}+1}$. ■

Lemma 6 For every sort B and ordinal $\mathfrak{a} < \mathfrak{h}$, $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}$ is a computability predicate.

Proof. We proceed by induction on $<_{\mathbb{S}}$ and $\mathfrak{a}$. If $\mathfrak{a}$ is an infinite limit ordinal, then $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}$ is a computability predicate by definition of lub since, by induction hypothesis, for all $\mathfrak{b} < \mathfrak{a}$, $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}$ is a computability predicate.

We are left with the cases of 0 and successor ordinals. Given a predicate P on triples $(\mathsf{c}, \vec{t}, \vec{T})$, let $\mathrm{SN}^{\mathsf{B}}(P) = \{t \in \mathrm{SN} \mid \mathbb{C}^{\mathsf{B}}_{\to^*}(t) \subseteq P\}$. We have $\mathcal{S}^{\mathsf{B}}_{0} =$

$\mathrm{SN}^{\mathsf{B}}(P_0)$ for some predicate P_0, and $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}+1} = \mathrm{SN}^{\mathsf{B}}(P_{\mathfrak{a}+1})$ for some predicate $P_{\mathfrak{a}+1}$. However, for all predicate P, $\mathrm{SN}^{\mathsf{B}}(P)$ is a computability predicate:

- $\mathrm{SN}^{\mathsf{B}}(P) \subseteq \mathrm{SN}$ by definition.
- If $t \in \mathrm{SN}^{\mathsf{B}}(P)$ and $t \to t'$, then $t' \in \mathrm{SN}^{\mathsf{B}}(P)$ since $t' \in \mathrm{SN}$ and $\mathbb{C}^{\mathsf{B}}_{\to^*}(t') \subseteq \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$.
- Assume now that t is neutral and $\to(t) \subseteq \mathrm{SN}^{\mathsf{B}}(P)$. Then, $t \in \mathrm{SN}$. Assume moreover that $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$. Since t is neutral, there is t' such that $t \to t'$ and $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t')$. Therefore, $(\mathsf{c}, \vec{t}, \vec{T}) \in P$ and $t \in \mathrm{SN}^{\mathsf{B}}(P)$. ■

Lemma 7 For every sort B, $\mathcal{S}^{\mathsf{B}}_{\mathfrak{h}} = \mathsf{B}$.

Proof. Since $\mathcal{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B} = \mathcal{D}^{\mathsf{B}}_{\mathfrak{h}}$, it suffices to prove that, for all $\mathfrak{a}$, $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{h}}$, that is, for all $\mathfrak{a}$, there is $\mathfrak{b} < \mathfrak{h}$ such that $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{b}}$. We proceed by induction on $<_{\mathbb{S}}$ and $\mathfrak{a}$.

- $\mathcal{D}^{\mathsf{B}}_0 = \bot \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{h}}$.
- Let $\mathfrak{a}$ be an infinite limit ordinal smaller than $\mathfrak{h}$. By induction hypothesis, for all $\mathfrak{b} < \mathfrak{a}$, $\mathcal{D}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{h}}$. Therefore, $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{h}}$.
- Let now $\mathfrak{a} + 1 < \mathfrak{h}$. By induction hypothesis, $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{h}}$.

  First note that, since $\mathfrak{h}$ is a successor cardinal, it is regular, that is, it is equal to its cofinality. And since it is uncountable, it is ω-complete, that is, every countable subset of $\mathfrak{h}$ has a least upper bound in $\mathfrak{h}$.

  Let $X = \{o_{\mathcal{S}^{\mathsf{B}}}(t) \mid t \in \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}\}$ and $\mathfrak{c} = \sup(X)$. Since $|X| \leq |\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}| \leq |\mathbb{L}| \leq \omega$, we have $\mathfrak{c} < \mathfrak{h}$ and $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}$.

  Let now $\mathfrak{d} = \sup(X \cup Y)$ where Y is the set of the ordinals $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$ such that there are $t \in \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}$ and $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$. Since $|Y| \leq \omega$ ($\to$ is finitely branching), we have $\sup(Y) < \mathfrak{h}$ and $\mathfrak{d} + 1 < \mathfrak{h}$.

  We now prove that $\mathcal{D}^{\mathsf{B}}_{\mathfrak{a}+1} \subseteq \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}+1}$. Let $t \in \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}+1}$. Then, $t \in \mathrm{SN}$. Let now $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$. If $k \in \{\mathrm{p}^{\mathsf{c}} + 1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, then $t_k \in T_k$. Otherwise, $t_k \in [\mathsf{B} : \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}]T_k$. Since B occurs only positively in T_k, we have $[\mathsf{B} : \mathcal{D}^{\mathsf{B}}_{\mathfrak{a}}]T_k \subseteq [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k$. Since $\mathfrak{c} \leq \mathfrak{d}$ and $\mathcal{S}^{\mathsf{B}}$ is monotone, we have $[\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k \subseteq [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}}]T_k$. Finally, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{d}$. Therefore, $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{d}+1}$. ■

Let us now see some properties of $\mathcal{S}$:

Lemma 8

1. $t \in \mathcal{S}^{\mathsf{B}}_0$ iff $t \in \mathsf{B}$ and, for all $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) = \mathrm{p}^{\mathsf{c}} = 0$.
2. $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}+1}$ iff $t \in \mathsf{B}$ and, for all $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a} + 1$ and, for all $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}$.

Proof.

1. Immediate.

2. Assume that $t \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}+1}$. Then, $t \in \mathsf{B}$. Assume moreover that $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\rightarrow^*}(t)$. Then, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}+1$ and, for all $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k = \mathcal{S}^{\mathsf{c},k}_{\mathfrak{a}}$. Hence, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}$. Conversely, if $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}$, then $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$. ■

Lemma 9 If $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $\mathsf{c}\,\vec{t} \in \mathsf{B}$, then:

1. $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) \geq \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$.

2. $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) > o_{\mathcal{S}^{\mathsf{c},k}}(t_k)$ for all $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$.

Proof. Let $\mathfrak{a} = o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t})$.

1. If $\mathfrak{a} = 0$, then $\mathsf{c}\,\vec{t} \in \mathcal{S}^{\mathsf{B}}_0$ and $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) = 0 \leq \mathfrak{a}$. Otherwise, since $\mathcal{S}^{\mathsf{B}}$ is continuous, $\mathfrak{a} = \mathfrak{b}+1$ for some $\mathfrak{b}$. Hence, $\mathsf{c}\,\vec{t} \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{b}+1}$ and $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{b}+1 = \mathfrak{a}$ too.

2. If $\mathfrak{a} = 0$, then $\mathsf{c}\,\vec{t} \in \mathcal{S}^{\mathsf{B}}_0$ and $\mathrm{p}^{\mathsf{c}} = 0$. Otherwise, since $\mathcal{S}^{\mathsf{B}}$ is continuous, $\mathfrak{a} = \mathfrak{b}+1$ for some $\mathfrak{b}$. Hence, $\mathsf{c}\,\vec{t} \in \mathcal{S}^{\mathsf{B}}_{\mathfrak{b}+1}$ and $t_k \in [\mathsf{B} : \mathcal{S}^{\mathsf{B}}_{\mathfrak{b}}]T_k$. Thus, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{b} < \mathfrak{a}$.■

Theorem 1 If $t \in \mathsf{B}$, then $o_{\mathcal{S}^{\mathsf{B}}}(t) = \delta \sup(R \cup S \cup T)$[4] where:

- $\delta\mathfrak{a} = \mathfrak{a}+1$ if $\mathfrak{a}$ is an infinite limit ordinal, and $\delta\mathfrak{a} = \mathfrak{a}$ otherwise;
- $R = \{o_{\mathcal{S}^{\mathsf{B}}}(t') \mid t \rightarrow t'\}$;
- $S = \{o_{\mathcal{S}^{\mathsf{c},k}}(t_k)+1 \mid (\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}},\ t = \mathsf{c}\,\vec{t},\ k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}\}$;
- $T = \{\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \mid (\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}},\ t = \mathsf{c}\,\vec{t}\}$.

Proof. Let $\mathfrak{a} = \sup(R \cup S \cup T\})$ and $\mathfrak{b} = o_{\mathcal{S}^{\mathsf{B}}}(t)$.

We first prove that $\mathfrak{b} \geq \delta\mathfrak{a}$. First, we have $\mathfrak{b} \geq 0$. Let now t' such that $t \rightarrow t'$. Then, $\mathfrak{b} \geq o_{\mathcal{S}^{\mathsf{B}}}(t')$ by Lemma 2 (1). Assume now that $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $t = \mathsf{c}\,\vec{t}$. Then, $\mathfrak{b} \geq \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$ and, if $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, then $\mathfrak{b} > o_{\mathcal{S}^{\mathsf{c},k}}(t_k)$. Therefore, $\mathfrak{b} \geq \mathfrak{a}$.

Since $\mathcal{S}^{\mathsf{B}}$ is continuous, $\mathfrak{b}$ cannot be an infinite limit ordinal. So, if $\mathfrak{a}$ is an infinite limit ordinal, then $\mathfrak{b} > \mathfrak{a}$ and thus $\mathfrak{b} \geq \delta\mathfrak{a}$. Otherwise, $\delta\mathfrak{a} = \mathfrak{a}$ and thus $\mathfrak{b} \geq \delta\mathfrak{a}$.

Now, to have $\mathfrak{b} \leq \delta\mathfrak{a}$, we prove that $t \in \mathcal{S}^{\mathsf{B}}_{\delta\mathfrak{a}}$, by case on $\delta\mathfrak{a}$.

- $\delta\mathfrak{a} = 0$. Then, $\mathfrak{a} = 0$. Let $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\rightarrow^*}(t)$.
  - Case $t = \mathsf{c}\,\vec{t}$. Then, $S = \emptyset$, $\mathrm{p}^{\mathsf{c}} = 0$ and $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) = 0$. Thus, by Lemma 8, $t \in \mathcal{S}^{\mathsf{B}}_0$.
  - Case $t \rightarrow t' \rightarrow^* \mathsf{c}\,\vec{t}$. Then, $o_{\mathcal{S}^{\mathsf{B}}}(t') = 0$. So, $\mathrm{p}^{\mathsf{c}} = 0$, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) = 0$ and $t \in \mathcal{S}^{\mathsf{B}}_0$.

[4] Note that $\sup(\emptyset) = 0$ by definition of sup.

- $\delta\mathfrak{a} = \mathfrak{a}' + 1$. Let $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$.
  - Case $t = \mathsf{c}\,\vec{t}$. If $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, then $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}'$ since, by Lemma 9 (2), $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) < \mathfrak{a} \leq \delta\mathfrak{a} = \mathfrak{a}' + 1$. Moreover, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}' + 1$ since, by Lemma 9 (1), $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a} \leq \delta\mathfrak{a} = \mathfrak{a}' + 1$. Therefore, by Lemma 8, $t \in \mathcal{S}_{\mathfrak{a}'+1}$.
  - Case $t \to t' \to^* \mathsf{c}\,\vec{t}$. If $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, then $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}'$ since $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) < o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) \leq o_{\mathcal{S}^{\mathsf{B}}}(t') \leq \mathfrak{a} \leq \delta\mathfrak{a} = \mathfrak{a}' + 1$. Moreover, $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}' + 1$ since $\Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) \leq o_{\mathcal{S}^{\mathsf{B}}}(t') \leq \mathfrak{a} \leq \delta\mathfrak{a} = \mathfrak{a}' + 1$. Therefore, $t \in \mathcal{S}_{\mathfrak{a}'+1}$. ■

Note that T has 0 or 1 element, and S has a finite number of elements. So, $\sup(R \cup S \cup T\})$ is an infinite limit ordinal only when R is infinite, that is, only when $\to$ is infinitely branching.

Note also that taking $\Sigma^{\mathsf{c}}(\vec{\mathfrak{a}}) \leq \sup\{\mathfrak{a}_k + 1 \mid k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}\}$ gives the same notion of size as taking $\Sigma^{\mathsf{c}}(\vec{\mathfrak{a}}) = 0$. On the other hand, if $\Sigma^{\mathsf{c}}(\vec{\mathfrak{a}}) \geq \sup\{\mathfrak{a}_k + 1 \mid k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}\}$, then Σ^{c} determines the size of irreducible terms of the form $\mathsf{c}\,\vec{t}$:

Corollary 1 Assume that Σ^{c} is strictly extensive wrt. recursive arguments (*i.e.* $\mathfrak{a}_k < \Sigma^{\mathsf{c}}(\vec{\mathfrak{a}})$ if $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$) and $\Sigma^{\mathsf{c}}(\vec{\mathfrak{a}})$ is not an infinite limit ordinal. Then, for all $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ such that $\mathsf{c}\,\vec{t} \in \mathsf{B}$ and $\mathsf{c}\,\vec{t}$ is irreducible, we have $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$.

Proof. Since $\mathsf{c}\,\vec{t}$ is irreducible, $R = \emptyset$. Let $\mathfrak{a} = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$. Since $\mathfrak{a} > o_{\mathcal{S}^{\mathsf{c},k}}(t_k)$ whenever $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) = \delta\mathfrak{a}$. Since $\mathfrak{a}$ is not an infinite limit, $\delta\mathfrak{a} = \mathfrak{a}$. ■

Note that all the previous results hold even if $\mathbb{C}$ and $\mathbb{F}$ are not disjoint: up to now, we used no assumption on the rewrite relation $\to$. In the following, we will use the fact that there is no rule of the form $\mathsf{c}\,\vec{l} \to r$ with $\mathsf{c} \in \mathbb{C}$:

Corollary 2 Assume that Σ^{c} is monotone wrt. every argument, strictly extensive wrt. recursive arguments and not an infinite limit ordinal. Then, for all $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ such that $\mathsf{c}\,\vec{t} \in \mathsf{B}$, we have $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\,\vec{t}) = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$.

Proof. We proceed by induction on $\vec{t}$ with $\leftarrow_{\mathrm{prod}}$ as well-founded relation. Assume that $\mathsf{c}\,\vec{t} \to u$. Then, there are $\vec{u}$ such that $u = \mathsf{c}\,\vec{u}$ and $\vec{t} \to_{\mathrm{prod}} \vec{u}$. Hence, $o_{\mathcal{S}^{\mathsf{c}}}(\vec{u}) \leq_{\mathrm{prod}} o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})$ and, by induction hypothesis, $o(\mathsf{c}\,\vec{u}) = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{u}))$. So, $o(\mathsf{c}\,\vec{u}) \leq \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$ since Σ^{c} is monotone. Therefore, $o(\mathsf{c}\,\vec{t}) = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$. ■

Finally, we are going to prove that, if $\to$ is locally confluent, hence confluent on strongly normalizing terms [New42], then the size of a term is equal to the size of its normal form when its type is a strictly positive sort:

Definition 9 (Strictly positive sorts) A sort B is *strictly positive* if, for every constructor $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$ and argument $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, T_k is positive wrt. B

and either T_k is a strictly positive sort[5] $\mathsf{C} <_{\mathbb{S}} \mathsf{B}$ or T_k is of the form $\vec{U} \Rightarrow \mathsf{B}$ with $\mathrm{Pos}(\mathsf{B}, \vec{U}) = \emptyset$.

Examples of strictly positive sorts are Peano natural numbers and Howard constructive ordinals.

Lemma 10 Assume that $\rightarrow$ is locally confluent and, for every constructor c, Σ^{c} is monotone wrt. every argument, strictly extensive wrt. recursive arguments and not an infinite limit ordinal. Then, for every strictly positive sort B and term $t \in \mathsf{B}$, $o_{\mathcal{S}^{\mathsf{B}}}(t) = o_{\mathcal{S}^{\mathsf{B}}}(t{\downarrow})$, where $t{\downarrow}$ is the normal form of t.

Proof. First note that $o_{\mathcal{S}^{\mathsf{B}}}(t{\downarrow}) \leq o_{\mathcal{S}^{\mathsf{B}}}(t)$ since $t \rightarrow^* t{\downarrow}$. We now prove that, for all B, for all $t \in \mathsf{B}$, $o_{\mathcal{S}^{\mathsf{B}}}(t) \leq o_{\mathcal{S}^{\mathsf{B}}}(t{\downarrow})$, by induction on $(\mathsf{B}, o_{\mathcal{S}^{\mathsf{B}}}(t), t)$ with $(<_{\mathbb{S}}, <, \leftarrow)_{\mathrm{lex}}$ as well-founded relation. If $t \rightarrow u$ then $o_{\mathcal{S}^{\mathsf{B}}}(u) \leq o_{\mathcal{S}^{\mathsf{B}}}(t)$. Hence, by induction hypothesis on the 2nd or 3rd component, $o_{\mathcal{S}^{\mathsf{B}}}(u) = o_{\mathcal{S}^{\mathsf{B}}}(u{\downarrow}) = o_{\mathcal{S}^{\mathsf{B}}}(t{\downarrow})$. Assume now that $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $t = \mathsf{c}\,\vec{t}$. Then, by Corollary 2, $o_{\mathcal{S}^{\mathsf{B}}}(t) = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}))$ and $o_{\mathcal{S}^{\mathsf{B}}}(t{\downarrow}) = \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t}{\downarrow}))$. Since Σ^{c} is monotone, it suffices to prove that, for all $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) \leq o_{\mathcal{S}^{\mathsf{c},k}}(t_k{\downarrow})$. Since B is strictly positive, there are two cases:

- T_k is a strictly positive sort $\mathsf{C} <_{\mathbb{S}} \mathsf{B}$. Then, by induction hypothesis on the 1st component, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) = o_{\mathcal{S}^{\mathsf{C}}}(t_k) = o_{\mathcal{S}^{\mathsf{C}}}(t_k{\downarrow}) = o_{\mathcal{S}^{\mathsf{c},k}}(t_k{\downarrow})$.
- There is $\vec{U}$ such that $T_k = \vec{U} \Rightarrow \mathsf{B}$ and $\mathrm{Pos}(\mathsf{B}, \vec{U}) = \emptyset$. Then, by Lemma 3 (3), $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) = \sup\{o_{\mathcal{S}^{\mathsf{B}}}(t_k\vec{u}) \mid \vec{u} \in \vec{U}\}$ and $o_{\mathcal{S}^{\mathsf{c},k}}(t_k{\downarrow}) = \sup\{o_{\mathcal{S}^{\mathsf{B}}}(t_k{\downarrow}\vec{u}) \mid \vec{u} \in \vec{U}\}$. Since $o_{\mathcal{S}^{\mathsf{B}}}(t_k{\downarrow}\vec{u}) \leq o_{\mathcal{S}^{\mathsf{B}}}(t_k\vec{u}) < o_{\mathcal{S}^{\mathsf{B}}}(t)$, by induction hypothesis on the 2nd component, $o_{\mathcal{S}^{\mathsf{B}}}(t_k\vec{u}) = o_{\mathcal{S}^{\mathsf{B}}}(t_k{\downarrow}\vec{u})$. Therefore, $o_{\mathcal{S}^{\mathsf{c},k}}(t_k) = o_{\mathcal{S}^{\mathsf{c},k}}(t_k{\downarrow})$. ■

4 Termination criterion

In this section, we describe a termination criterion that capitalizes on the fact that some terms can be assigned an ordinal size. The idea is simple: if for every rewrite step $\mathsf{f}l \rightarrow r$ and every function call $\mathsf{g}m$ in r, the size of m is strictly smaller than the size of l, then there cannot be any infinite reduction.

The idea, dating back to Hughes, Pareto and Sabry [HPS96], consists in introducing symbolic expressions representing ordinals and logical rules for deducing information about the size of terms, namely, that it is bounded by some expression. Hence, termination is reduced to checking the decreasingness of symbolic size expressions.

Following these authors, we replace every sort B by a pair (B, a), written B_a, where a is a symbolic expression from an algebra interpretable in ordinals, so that a term is of size-annotated type B_a if it is of type B and of size *smaller than or equal to* the interpretation of a. The typing rules of Figure 2 are then easily turned into valid deduction rules on size annotations. Moreover, the monotony

[5] This is a restriction wrt. the definition given in [CPM88] where T_k can be any type where B does not occur.

of stratifications naturally induces a notion of subtyping on size-annotated types: a term of type B_a is also of type B_b if $a \leq b$.

4.1 Size-annotated types

In the previously mentioned works, only two particular algebras have been considered so far. First, the successor algebra, that is the algebra generated by a unique size function symbol $\mathtt{s}$ interpreted as the successor function on ordinals (see Definition 21). Second, when $\mathfrak{h}$ is restricted to ω (*e.g.* when inductive types are restricted to first-order data types), the algebra of Presburger arithmetic generated from the symbols $\mathtt{0}$, $\mathtt{s}$ and $+$ interpreted by zero, the successor function and the addition on natural numbers respectively, the first-order theory of which is decidable [Pre29].

Other algebras are however interesting as we shall see in some examples. For instance, the max-successor algebra, that is, the successor algebra extended by a $\mathtt{max}$ operator, and the max-plus algebra, that is, the algebra generated by the symbols $\mathtt{0}$, $\mathtt{1}$, $+$ and $\mathtt{max}$.

So, in the following, we consider an arbitrary size algebra and prove general results under some conditions on it. Then, in Section 7, we prove that these conditions are in particular satisfied by the successor algebra.

Definition 10 (Size algebra) A *size algebra* is given by:

- a first-order term algebra $\mathtt{A}$ built from a set $\mathtt{V}$ of size variables $\alpha, \beta, \ldots$ and a set $\mathtt{F}$ of size function symbols $\mathtt{f}, \mathtt{g}, \ldots$ of fixed arity, disjoint from $\mathtt{V}$;
- a quasi-ordering $\leq_{\mathtt{A}}$ on $\mathtt{A}$ that is stable by size substitution: $a\varphi <_{\mathtt{A}} b\varphi$ ($a\varphi \simeq_{\mathtt{A}} b\varphi$ resp.) whenever $a <_{\mathtt{A}} b$ ($a \simeq_{\mathtt{A}} b$ resp.) and $\varphi : \mathtt{V} \to \mathtt{A}$;
- for each size function symbol $\mathtt{f} \in \mathtt{F}$ of arity n, a function $\mathtt{f}_{\mathfrak{h}} : \mathfrak{h}^n \to \mathfrak{h}$ so that, for every valuation $\mu : \mathtt{V} \to \mathfrak{h}$, $a\mu < b\mu$ ($a\mu = b\mu$ resp.) whenever $a <_{\mathtt{A}} b$ ($a \simeq_{\mathtt{A}} b$ resp.) where, as usual, $\alpha\mu = \mu(\alpha)$ and $(\mathtt{f}a_1 \ldots a_n)\mu = \mathtt{f}_{\mathfrak{h}}(a_1\mu, \ldots, a_n\mu)$.

A size algebra is *monotone* if every size function symbol is monotone in every argument, that is, $\mathtt{f}\vec{a} \leq_{\mathtt{A}} \mathtt{f}\vec{b}$ whenever $\vec{a}\,(\leq_{\mathtt{A}})_{\mathrm{prod}}\,\vec{b}$.

Let $a \leq_{\mathrm{ext}} b$ iff, for all μ, $a\mu \leq b\mu$. Note that $\leq_{\mathrm{ext}}$ satisfies the above conditions, and every ordering $\leq_{\mathtt{A}}$ satisfying the above conditions is included in $\leq_{\mathrm{ext}}$. So, one can always take $\leq_{\mathrm{ext}}$ for $\leq_{\mathtt{A}}$ but, as this ordering may be undecidable, one may prefer to use a decidable subordering instead.

Definition 11 (Size-annotated types) The set $\mathbb{T}_{\mathtt{A}}$ of (possibly) annotated types is defined as follows:

- if T is a type, then $T \in \mathbb{T}_{\mathtt{A}}$;
- if B is a sort and a a size expression, then $\mathsf{B}_a \in \mathbb{T}_{\mathtt{A}}$;
- if U and V belong to $\mathbb{T}_{\mathtt{A}}$, then $U \Rightarrow V \in \mathbb{T}_{\mathtt{A}}$.

Let $\mathrm{Var}(T)$ be the set of size variables occurring in T.

Given an annotated type T, let $|T|$ be the type obtained by removing in T every size annotation.

Given a sort B, a size variable α and a type T, let $\mathrm{Annot}(T, \mathsf{B}, a)$ be the annotated type obtained by annotating in T every occurrence of B by a.

The set of positions of a size variable α in an annotated type T, $\mathrm{Pos}(\alpha, T)$, is defined as follows:

- $\mathrm{Pos}(\alpha, U \Rightarrow V) = \{1p \mid p \in \mathrm{Pos}(\alpha, U)\} \cup \{2p \mid p \in \mathrm{Pos}(\alpha, V)\}$;
- $\mathrm{Pos}(\alpha, \mathsf{B}) = \emptyset$;
- $\mathrm{Pos}(\alpha, \mathsf{B}_b) = \{0p \mid p \in \mathrm{Pos}(\alpha, b)\}$;
- $\mathrm{Pos}(\alpha, \beta) = \{\varepsilon\}$ if $\alpha = \beta$;
- $\mathrm{Pos}(\alpha, \beta) = \emptyset$ if $\alpha \neq \beta$;
- $\mathrm{Pos}(\alpha, \mathtt{f}\,\vec{b}) = \{ip \mid p \in \mathrm{Pos}(\alpha, b_i)\}$;

The sets of *positive* and *negative positions* of an annotated type are defined as follows:

- $\mathrm{Pos}^+(U \Rightarrow V) = \{1p \mid p \in \mathrm{Pos}^-(U)\} \cup \{2p \mid p \in \mathrm{Pos}^+(V)\}$;
- $\mathrm{Pos}^-(U \Rightarrow V) = \{1p \mid p \in \mathrm{Pos}^+(U)\} \cup \{2p \mid p \in \mathrm{Pos}^-(V)\}$;
- $\mathrm{Pos}^+(\mathsf{B}) = \{\varepsilon\}$;
- $\mathrm{Pos}^-(\mathsf{B}) = \emptyset$;
- $\mathrm{Pos}^+(\mathsf{B}_b) = \{0p \mid p \in \mathrm{Pos}^+(b)\}$;
- $\mathrm{Pos}^-(\mathsf{B}_b) = \{0p \mid p \in \mathrm{Pos}^-(b)\}$;
- $\mathrm{Pos}^+(\mathtt{f}\,\vec{b}) = \{ip \mid i \in \mathrm{Mon}^+(\mathtt{f}), p \in \mathrm{Pos}^+(b_i)\} \cup \{ip \mid i \in \mathrm{Mon}^-(\mathtt{f}), p \in \mathrm{Pos}^-(b_i)\}$;
- $\mathrm{Pos}^-(\mathtt{f}\,\vec{b}) = \{ip \mid i \in \mathrm{Mon}^+(\mathtt{f}), p \in \mathrm{Pos}^-(b_i)\} \cup \{ip \mid i \in \mathrm{Mon}^-(\mathtt{f}), p \in \mathrm{Pos}^+(b_i)\}$;

where $\mathrm{Mon}^+(\mathtt{f})$ ($\mathrm{Mon}^-(\mathtt{f})$ resp.) is the set of arguments in which $\mathtt{f}$ is monotone (anti-monotone resp.) wrt. $\leq_{\mathtt{A}}$.

In order to combine terms with annotated and unannotated types, we extend $\mathtt{A}$ by a greatest element ∞ and identify B_∞ with B:

Definition 12 (Top-extension of a size algebra) The *top-extension* of a size algebra $\mathtt{A}$ is a set $\mathtt{A} \cup \{\infty\}$ with $\infty \notin \mathtt{A}$. Given $\mathsf{B} \in \mathbb{S}$, let $\mathsf{B}_\infty = \mathsf{B}$ (we identify a sort B annotated by ∞ with B). Given extended size expressions $a, b \in \mathtt{A} \cup \{\infty\}$, let $a \leq^\infty_{\mathtt{A}} b$ if $a \leq_{\mathtt{A}} b$ or $b = \infty$. Given $\varphi : \mathtt{V} \to \mathtt{A} \cup \{\infty\}$, let $a\varphi = \infty$ if a contains a variable α such that $\varphi(\alpha) = \infty$, and $a\varphi$ be the usual substitution otherwise.

We now propose to users a syntactic way to specify their own notions of size through the annotation of constructor types. We assume that every constructor type is annotated in a way that allow us to define a size function, hence a stratification for every sort, and thus an interpretation of every annotated type in computability predicates.

Definition 13 (Annotated types of constructors) We assume that every constructor c with $\Theta(\mathsf{c}) = T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^\mathsf{c}} \Rightarrow \mathsf{B}$ is equipped with an annotated type

$$\overline{\Theta}(\mathsf{c}) = \mathrm{Annot}(T_1, \Sigma_1^\mathsf{c}, \alpha_1^\mathsf{c}) \Rightarrow \ldots \Rightarrow \mathrm{Annot}(T_{\mathrm{q}^\mathsf{c}}, \Sigma_{\mathrm{q}^\mathsf{c}}^\mathsf{c}, \alpha_{\mathrm{q}^\mathsf{c}}^\mathsf{c}) \Rightarrow T_{\mathrm{q}^\mathsf{c}+1} \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^\mathsf{c}} \Rightarrow \mathsf{B}_{\sigma^\mathsf{c}}$$

where:

- $\alpha_1^\mathsf{c}, \ldots, \alpha_{\mathrm{p}^\mathsf{c}}^\mathsf{c} \in \mathtt{V}$;
- $\alpha_{\mathrm{p}^\mathsf{c}+1}^\mathsf{c}, \ldots, \alpha_{\mathrm{q}^\mathsf{c}}^\mathsf{c} \in \mathtt{V} \cup \{\infty\}$;
- the variables of $\{\alpha_1^\mathsf{c}, \ldots, \alpha_{\mathrm{q}^\mathsf{c}}^\mathsf{c}\}$ are either pairwise equal or pairwise distinct, in which case we say that the annotated type of c is *functional*;
- for all $i \in \{1, \ldots, \mathrm{p}^\mathsf{c}\}$, $\Sigma_i^\mathsf{c} = \mathsf{B}$;
- for all $i \in \{\mathrm{p}^\mathsf{c} + 1, \ldots, \mathrm{p}^\mathsf{c}\}$ with $\alpha_i^\mathsf{c} \in \mathtt{V}$, Σ_i^c occurs in T_i;
- for all $i \in \{\mathrm{p}^\mathsf{c} + 1, \ldots, \mathrm{p}^\mathsf{c}\}$ with $\alpha_i^\mathsf{c} \in \mathtt{V}$, $\mathrm{Pos}(\Sigma_i^\mathsf{c}, T_i) \subseteq \mathrm{Pos}^+(T_i)$;
- $\sigma^\mathsf{c} \in \mathtt{A} \cup \{\infty\}$;
- $\mathrm{Var}(\sigma^\mathsf{c}) = \{\alpha_i^\mathsf{c} \in \mathtt{V}\}$;
- σ^c is monotone wrt. every argument: for all $i \in \{1, \ldots, \mathrm{q}^\mathsf{c}\}$, $\mathrm{Pos}(\alpha_i^\mathsf{c}, \sigma^\mathsf{c}) \subseteq \mathrm{Pos}^+(\sigma^\mathsf{c})$;
- σ^c is strictly extensive wrt. recursive arguments: for all $i \in \{1, \ldots, \mathrm{p}^\mathsf{c}\}$, $\alpha_i^\mathsf{c} <_\mathtt{A} \sigma^\mathsf{c}$.

We say that $\overline{\Theta}$ is functional if every constructor has a functional annotated type.

The precise semantics of these annotations will be given in the next definition. The intuition is that the size of a term of the form $\mathsf{c}\vec{t}$ will be given by the interpretation in ordinals of σ^c with each α_i^c, the abstract size of the i-th argument of c, interpreted by the actual size of t_i in $[\Sigma_i^\mathsf{c} : \mathcal{S}^{\Sigma_i^\mathsf{c}}]T_i$.

Note that a constructor c can always be annotated under the above conditions in any algebra extending the successor algebra by taking:

- $\alpha_1^\mathsf{c} = \ldots = \alpha_{\mathrm{p}^\mathsf{c}}^\mathsf{c}$;
- $\alpha_{\mathrm{p}^\mathsf{c}+1}^\mathsf{c} = \ldots = \alpha_{\mathrm{q}^\mathsf{c}}^\mathsf{c} = \infty$;
- $\sigma^\mathsf{c} = \infty$ if $\mathrm{p}^\mathsf{c} = 0$, and $\sigma^\mathsf{c} = \mathsf{s}\,\alpha_1^\mathsf{c}$ otherwise.

For instance, for the constructors of the sort T of labeled binary trees with labels in some sort $\mathsf{B} <_{\mathbb{S}} \mathsf{T}$, one can take in the successor algebra $\mathsf{leaf} : \mathsf{B} \Rightarrow \mathsf{T}$ and $\mathsf{node} : \mathsf{T}_\alpha \Rightarrow \mathsf{T}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{T}_{\mathsf{s}\alpha}$. So, we have $\Sigma_1^{\mathsf{leaf}} = \mathsf{B}$, $\alpha_1^{\mathsf{leaf}} = \infty$, $\sigma^{\mathsf{leaf}} = \infty$; and $\Sigma_1^{\mathsf{node}} = \Sigma_2^{\mathsf{node}} = \mathsf{T}$, $\alpha_1^{\mathsf{node}} = \alpha_2^{\mathsf{node}} = \alpha$ and $\sigma^{\mathsf{node}} = \mathsf{s}\,\alpha$. Because leaf is not given a definite size, every tree containing a leaf has no definite size too. In this case, the size information is useless for proving the termination of a rule like $\mathsf{f}\,(\mathsf{node}\,\mathsf{leaf}\,x) \to \mathsf{f}\,x$.

As for the constructors of the sort O of Howard's constructive ordinals, we can take $\mathsf{zero} : \mathsf{O}$, $\mathsf{succ} : \mathsf{O}_\alpha \Rightarrow \mathsf{O}_{\mathsf{s}\alpha}$ and $\mathsf{lim} : (\mathsf{N} \Rightarrow \mathsf{O}_\alpha) \Rightarrow \mathsf{O}_{\mathsf{s}\alpha}$, that is, $\sigma^{\mathsf{zero}} = \infty$; $\Sigma_1^{\mathsf{succ}} = \mathsf{O}$, $\alpha_1^{\mathsf{succ}} = \alpha$ and $\sigma^{\mathsf{succ}} = \mathsf{s}\,\alpha$; and $\Sigma_1^{\mathsf{lim}} = \mathsf{O}$, $\alpha_1^{\mathsf{lim}} = \alpha$ and $\sigma^{\mathsf{lim}} = \mathsf{s}\,\alpha$.

However, in the successor algebra, constructors with at least two accessible arguments cannot have functional annotated types (because there is only one non-nullary symbol).

In contrast, in any algebra extending the max-successor algebra, a constructor c can always be equipped with a functional type satisfying the above conditions by taking:

- $\alpha_1^{\mathsf{c}}, \ldots, \alpha_{\mathrm{p}^{\mathsf{c}}}^{\mathsf{c}}$ pairwise distinct;
- $\alpha_{\mathrm{p}^{\mathsf{c}}+1}^{\mathsf{c}} = \ldots = \alpha_{\mathrm{q}^{\mathsf{c}}}^{\mathsf{c}} = \infty$;
- $\sigma^{\mathsf{c}} = \infty$ if $\mathrm{p}^{\mathsf{c}} = 0$, and $\sigma^{\mathsf{c}} = \mathsf{s}(\max \alpha_1^{\mathsf{c}} \ldots \alpha_{\mathrm{p}^{\mathsf{c}}}^{\mathsf{c}})$ otherwise.

For instance, for the constructors of the sort T of labeled binary trees, we can take $\mathsf{leaf} : \mathsf{B} \Rightarrow \mathsf{T}$ and $\mathsf{node} : \mathsf{T}_\alpha \Rightarrow \mathsf{T}_\beta \Rightarrow \mathsf{B} \Rightarrow \mathsf{T}_{\mathsf{s}(\alpha+\beta)}$.

We now extend the interpretation of types in computability predicates to annotated types, by defining a size function Σ^{c} for each constructor c:

Definition 14 (Interpretation of size-annotated types) First, for each constructor c with $\overline{\Theta}(c)$ as in Definition 13, we define a size function Σ^{c} as follows:

- $\Sigma^{\mathsf{c}}(\vec{\mathfrak{a}}) = 0$ if $\sigma^{\mathsf{c}} = \infty$,
- otherwise $\Sigma^{\mathsf{c}}(\vec{\mathfrak{a}}) = \sigma^{\mathsf{c}}\nu$ where $\nu : \mathrm{Var}(\sigma^{\mathsf{c}}) \to \mathfrak{h}$ is defined as follows:
  - if all the $\alpha_i^{\mathsf{c}} \in \mathtt{V}$ are pairwise distinct, then $\forall i \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$ with $\alpha_i^{\mathsf{c}} \in \mathtt{V}$, $\alpha_i^{\mathsf{c}}\nu = \mathfrak{a}_i$;
  - if all the $\alpha_i^{\mathsf{c}} \in \mathtt{V}$ are equal to some α, then $\alpha\nu = \sup\{\mathfrak{a}_i \mid i \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}, \alpha_i^{\mathsf{c}} \in \mathtt{V}\}$.

Then, given a valuation $\mu : \mathtt{V} \to \mathfrak{h}$, we interpret annotated types as follows:

- $\mathsf{B}\mu = \mathsf{B}$,
- $\mathsf{B}_a\mu = \mathcal{S}_{a\mu}^{\mathsf{B}}$ if $a \in \mathtt{A}$, where $\mathcal{S}$ is the stratification defined by Σ as in Definition 8,
- $(U \Rightarrow V)\mu = U\mu \Rightarrow V\mu$.

Note that the size function Σ^{c} is monotone wrt. every argument and strictly extensive wrt. recursive arguments, since σ^{c} is monotone wrt. every argument and strictly extensive wrt. recursive arguments.

We end this section by introducing some reflexive and transitive closure of the notion of accessible argument and prove some properties about it. In order to keep track of the sort with respect to which the size is measured, we consider a relation on triples (t, T, B) made of a term t, its type T and the sort B used to measure the size of t in $[\mathsf{B} : \mathcal{S}^{\mathsf{B}}]T$.

Definition 15 (Accessible subterm) We say that (u, U, C) is *accessible* in (t, T, B), written $(u, U, \mathsf{C}) \trianglelefteq_{\mathrm{a}} (t, T, \mathsf{B})$, if $(u, U, \mathsf{C}) = (t, T, \mathsf{B})$ or there are $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$ such that $t = \mathsf{c}\,\vec{t}$, $T = \mathsf{B}$ and $(u, U, \mathsf{C}) \trianglelefteq_{\mathrm{a}} (t_k, T_k, \Sigma^{\mathsf{c}}_k)$.

For example:

- $(x, \mathsf{N}, \mathsf{N})$ is accessible in $(\mathsf{s}\,x, \mathsf{N}, \mathsf{N})$ if $\mathsf{s} : \mathsf{N} \Rightarrow \mathsf{N}$;
- $(f, \mathsf{N} \Rightarrow \mathsf{O}, \mathsf{O})$ is accessible in $(\mathsf{lim}\,f, \mathsf{O}, \mathsf{O})$ if $\mathsf{lim} : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow \mathsf{O}$;
- $(x, \mathsf{N}, \mathsf{N})$ is accessible in $(\mathsf{pair}\,(\mathsf{s}\,x)\,y, \mathsf{P}, \mathsf{P})$ if $\mathsf{pair} : \mathsf{N} \Rightarrow \mathsf{N} \Rightarrow \mathsf{P}$ and $\mathsf{s} : \mathsf{N} \Rightarrow \mathsf{N}$.
- $(x, \mathsf{B} \Rightarrow \mathsf{C}, \mathsf{B})$ is not accessible in $(\mathsf{c}\,x, \mathsf{B}, \mathsf{B})$ if $\mathsf{c} : (\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow \mathsf{B}$.

Note that $\trianglelefteq_{\mathrm{a}}$ is stable by substitution, and that C occurs only positively in U whenever $(u, U, \mathsf{C}) \triangleleft_{\mathrm{a}} (t, T, \mathsf{B})$, where $\triangleleft_{\mathrm{a}}$ is the strict part of $\trianglelefteq_{\mathrm{a}}$.

Lemma 11

1. Accessibility preserves computability: if $(u, U, \mathsf{C}) \trianglelefteq_{\mathrm{a}} (t, T, \mathsf{B})$ and $t \in T$, then $u \in U$.
2. Accessibility makes size decrease: if for every constructor c, Σ^{c} is (strictly resp.) extensive wrt. every argument and $(u, U, \mathsf{C}) \trianglelefteq_{\mathrm{a}} (t, T, \mathsf{B})$ ($(u, U, \mathsf{C}) \triangleleft_{\mathrm{a}} (t, T, \mathsf{B})$ resp.), then $o_{[\mathsf{C}:\mathcal{S}^{\mathsf{C}}]U}(u) \leq o_{[\mathsf{B}:\mathcal{S}^{\mathsf{B}}]T}(t)$ ($o_{[\mathsf{C}:\mathcal{S}^{\mathsf{C}}]U}(u) < o_{[\mathsf{B}:\mathcal{S}^{\mathsf{B}}]T}(t)$ resp.).

Proof. By induction on $\trianglelefteq_{\mathrm{a}}$:

- $(u, U, \mathsf{C}) = (t, T, \mathsf{B})$. Immediate.
- There are $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$ such that $t = \mathsf{c}\,\vec{t}$, $T = \mathsf{B}$ and $(u, U, \mathsf{C}) \trianglelefteq_{\mathrm{a}} (t_k, T_k, \Sigma^{\mathsf{c}}_k)$. Then, $t_k \in T_k$ and, by induction hypothesis, $u \in U$ and $o_{[\mathsf{C}:\mathcal{S}^{\mathsf{C}}]U}(u) \leq^e o_{[\Sigma^{\mathsf{c}}_k:\mathcal{S}^{\Sigma^{\mathsf{c}}_k}]T_k}(t_k)$, where $\leq^e = \leq$ ($<$ resp.) if Σ is (strictly resp.) extensive. Since Σ^{c} is extensive wrt. every argument, $o_{[\Sigma^{\mathsf{c}}_k:\mathcal{S}^{\Sigma^{\mathsf{c}}_k}]T_k}(t_k) \leq^e \Sigma^{\mathsf{c}}(o_{\mathcal{S}^{\mathsf{c}}}(\vec{t})) \leq o_{\mathcal{S}^{\mathsf{B}}}(t) = o_{[\mathsf{B}:\mathcal{S}^{\mathsf{B}}]T}(t)$. Therefore, $o_{[\mathsf{C}:\mathcal{S}^{\mathsf{C}}]U}(u) \leq^e o_{[\mathsf{B}:\mathcal{S}^{\mathsf{B}}]T}(t)$. ■

4.2 Termination conditions

An important ingredient of the termination criterion is the way the sizes of function arguments are compared. In frameworks where functions are defined by fixpoint and case analysis, exactly one argument must decrease at a time. Here, we allow the comparison of various arguments simultaneously, possibly through some interpretation functions ζ.

Since not every term can be assigned a notion of size, and since two function calls can have different numbers of arguments, we first need to specify what arguments have to be taken into account and how their size are compared:

Definition 16 (Order on function calls) We assume given:

- a well-founded quasi-ordering $\leq_{\mathbb{F}}$ on $\mathbb{F}$ (precedence) that we extend into a well-founded quasi-ordering on $\mathbb{V} \cup \mathbb{C} \cup \mathbb{F}$ by taking $s <_{\mathbb{F}} \mathsf{f}$ whenever $s \in \mathbb{V} \cup \mathbb{C}$ and $\mathsf{f} \in \mathbb{F}$;
- for each $\mathsf{f} : \vec{T} \Rightarrow \mathsf{B}$:
  - a number q^{f} such that, for all $i \in \{1, \ldots, \mathrm{q}^{\mathsf{f}}\}$, T_i is a sort Σ_i^{f} (the first q^{f} arguments of f are the arguments that will be taken into account for proving termination);
  - an annotated type

    $$\overline{\Theta}(\mathsf{f}) = \mathrm{Annot}(T_1, \Sigma_1^{\mathsf{f}}, \alpha_1^{\mathsf{f}}) \Rightarrow \ldots \Rightarrow \mathrm{Annot}(T_{\mathrm{q}^{\mathsf{f}}}, \Sigma_{\mathrm{q}^{\mathsf{f}}}^{\mathsf{f}}, \alpha_{\mathrm{q}^{\mathsf{f}}}^{\mathsf{f}}) \Rightarrow T_{\mathrm{q}^{\mathsf{f}}+1} \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^{\mathsf{f}}} \Rightarrow \mathsf{B}_{\sigma^{\mathsf{f}}}$$

    with $\vec{\alpha}^{\mathsf{f}}$ distinct variables, $\sigma^{\mathsf{f}} \in \mathtt{A} \cup \{\infty\}$ and $\mathrm{Var}(\sigma^{\mathsf{f}}) \subseteq \{\vec{\alpha}^{\mathsf{f}}\}$;
  - for each $X \in \{\mathtt{A}, \mathfrak{h}\}$, a well-founded quasi-order $(\mathbb{D}_X^{\mathsf{f}}, \leq_X^{\mathsf{f}})$ and a map $\zeta_X^{\mathsf{f}} : X^{\mathrm{q}^{\mathsf{f}}} \to \mathbb{D}_X^{\mathsf{f}}$ such that:
    - $(\mathbb{D}_X^{\mathsf{f}}, \leq_X^{\mathsf{f}}) = (\mathbb{D}_X^{\mathsf{g}}, \leq_X^{\mathsf{g}})$ whenever $\mathsf{f} \simeq_{\mathbb{F}} \mathsf{g}$;
    - $\vec{a}\mu <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{b}\mu$ whenever $\vec{a} <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{b}$ and $\mu : \mathtt{V} \to \mathfrak{h}$;
    - $\vec{a}\mu \simeq_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{b}\mu$ whenever $\vec{a} \simeq_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{b}$ and $\mu : \mathtt{V} \to \mathfrak{h}$;
    - $\vec{a} <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{c}$ whenever $\vec{a} \leq_{\mathtt{A}}^{\infty} \vec{b}$ and $\vec{b} <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{c}$;
    - $\vec{\mathfrak{a}} <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{\mathfrak{c}}$ whenever $\vec{\mathfrak{a}} <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{\mathfrak{b}}$ and $\vec{\mathfrak{b}} \leq \vec{\mathfrak{c}}$;

    where $(x_1, .., x_{\mathrm{q}^{\mathsf{g}}}) \leq_X^{\mathsf{g},\mathsf{f}} (y_1, .., y_{\mathrm{q}^{\mathsf{f}}})$ iff $\mathsf{g} \simeq_{\mathbb{F}} \mathsf{f}$ and $\zeta_X^{\mathsf{g}}(x_1, .., x_{\mathrm{q}^{\mathsf{g}}}) \leq_X^{\mathsf{f}} \zeta_X^{\mathsf{f}}(y_1, .., y_{\mathrm{q}^{\mathsf{f}}})$.

A function call $\mathsf{f}\vec{t}$ gives rise to a pair (f, φ) where $\varphi : \{\vec{\alpha}^{\mathsf{f}}\} \to \mathtt{A}$ maps α_i^{f} to the size of t_i.

We then define the quasi-ordering on function calls, $\leq_{\mathtt{A}}$, as follows. Given $h \in \mathbb{V} \cup \mathbb{C} \cup \mathbb{F}$, $\psi : \{\vec{\alpha}^h\} \to \mathtt{A} \cup \{\infty\}$ with $\{\vec{\alpha}^h\} = \emptyset$ if $h \in \mathbb{V}$, $\mathsf{f} \in \mathbb{F}$, $\varphi : \{\vec{\alpha}^{\mathsf{f}}\} \to \mathtt{A} \cup \{\infty\}$, let

$$(h, \psi) \leq_{\mathtt{A}} (\mathsf{f}, \varphi) \text{ if } h <_{\mathbb{F}} \mathsf{f} \text{ or } \vec{\alpha}^h\psi <_{\mathtt{A}}^{h,\mathsf{f}} \vec{\alpha}^{\mathsf{f}}\varphi.$$

Its counterpart in $\mathfrak{h}$, $\leq_{\mathfrak{h}}$, is defined similarly as follows. Given $\mathsf{g} \in \mathbb{V} \cup \mathbb{C} \cup \mathbb{F}$, $\nu : \{\vec{\alpha}^{\mathsf{g}}\} \to \mathfrak{h}$, $\mathsf{f} \in \mathbb{F}$, $\mu : \{\vec{\alpha}^{\mathsf{f}}\} \to \mathfrak{h}$, let $(\mathsf{g}, \nu) <_{\mathfrak{h}} (\mathsf{f}, \mu)$ if $\mathsf{g} <_{\mathbb{F}} \mathsf{f}$ or $\vec{\alpha}^{\mathsf{g}}\nu <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{\alpha}^{\mathsf{f}}\mu$.

For the sake of simplicity, we assume that termination arguments come first. This is not a real restriction since arguments can always be permuted if needed.

For ζ^{f}_X, one can often take the identity (assuming that $\mathrm{q}^{\mathsf{f}} = \mathrm{q}^{\mathsf{g}}$ whenever $\mathsf{f} \simeq_{\mathbb{F}} \mathsf{g}$). In Example 6, we use a different function. When ζ^{f}_X is the identity, one can for instance take for $\leq^{\mathsf{f}}_{\mathtt{A}}$ ($\leq^{\mathsf{f}}_{\mathfrak{h}}$ resp.) the lexicographic or multiset extension [DM79] of $\leq_{\mathtt{A}}$ ($\leq$ resp.), or some combination thereof, for which one can easily prove the compatibility of $\leq^{\mathsf{f}}_{\mathtt{A}}$ ($\leq^{\mathsf{f}}_{\mathfrak{h}}$ resp.) with $\leq^{\infty}_{\mathtt{A}}$ ($\leq$ resp.). Indeed, we have $(\leq^{\infty}_{\mathtt{A}})_{\mathrm{prod}} \circ (<_{\mathtt{A}})_{\mathrm{lex}} \subseteq (<_{\mathtt{A}})_{\mathrm{lex}}$ (where $\circ$ is relation composition) since $\leq^{\infty}_{\mathtt{A}} \circ \leq_{\mathtt{A}} \subseteq \leq_{\mathtt{A}}$, and $<_{\mathrm{lex}} \circ \leq_{\mathrm{prod}} \subseteq <_{\mathrm{lex}}$.

We can now state our general termination theorem under abstract conditions on size annotations. We will then discuss these conditions in turn.

Theorem 2 Assume that constructor types are annotated as in Definition 13. The relation $\to = \to_\beta \cup \to_{\mathcal{R}}$ terminates on the set $\mathbb{T}$ of well-typed terms if $\to$ is finitely branching and, for each rule $l \to r \in \mathcal{R} \subseteq \mathbb{T}^2$, the left-hand side l is of the form $\mathsf{f}\vec{l}$, the annotated type of f is as in Definition 16, $|\vec{l}| \geq \mathrm{q}^{\mathsf{f}}$ and there are:

- a typing environment $\Gamma : \mathrm{FV}(r) \to \mathbb{T}_{\mathtt{A}}$ with, for every $(x, U) \in \Gamma$:
  - an integer $k^x \in \{1, \ldots, |\vec{l}|\}$ such that x occurs in l_{k^x} and,
  - if $k^x \leq \mathrm{q}^{\mathsf{f}}$, a sort Σ^x and a size variable α^x such that $U = \mathrm{Annot}(|U|, \Sigma^x, \alpha^x)$, indicating how to measure the size of x;
- symbolic size upper bounds $\varphi : \{\vec{\alpha}^{\mathsf{f}}\} \to \mathtt{A} \cup \{\infty\}$ for $l_1, \ldots, l_{\mathrm{q}^{\mathsf{f}}}$;

such that:

- **Accessibility.** For every $(x, U) \in \Gamma$:
  - either $k^x > \mathrm{q}^{\mathsf{f}}$, $l_{k^x} = x$ and $T_{k^x} = U$,
  - or $k^x \leq \mathrm{q}^{\mathsf{f}}$ and $(x, |U|, \Sigma^x) \trianglelefteq_{\mathrm{a}} (l_{k^x}, T_{k^x}, \Sigma^{\mathsf{f}}_{k^x})$;
- **Subject reduction and decreasingness.**
  $\Gamma \vdash^{\mathsf{f}}_{\varphi} r : (T_{|\vec{l}|+1} \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^{\mathsf{f}}} \Rightarrow \mathsf{B}_{\sigma^{\mathsf{f}}})\varphi$, where $\vdash^{\mathsf{f}}_{\varphi}$ is defined in Figures 3 and 4;
- **Minimality.** For all substitutions θ with $\vec{l}\theta \in \vec{T}$, there exists a valuation ν such that:
  - for all $i \in \{1, \ldots, \mathrm{q}^{\mathsf{f}}\}$, $\alpha^{\mathsf{f}}_i \varphi \nu = o_{\mathcal{S}^{\Sigma^{\mathsf{f}}_i}}(l_i\theta)$;
  - for all $(x, U) \in \Gamma$ with $k^x \leq \mathrm{q}^{\mathsf{f}}$, $o_{[\Sigma^x : \mathcal{S}^{\Sigma^x}]|U|}(x\theta) \leq \alpha^x \nu$,

  where $\mathcal{S}$ is the stratification defined in Definition 8 using the size function Σ defined in Definition 14;
- **Monotony.** For all $i \in \{1, \ldots, \mathrm{q}^{\mathsf{f}}\}$, $\mathrm{Pos}(\alpha^{\mathsf{f}}_i, \sigma^{\mathsf{f}}) \subseteq \mathrm{Pos}^+(\sigma^{\mathsf{f}})$;

Figure 3: Computability closure of (f, φ)

$$\text{(app-decr)}\quad \frac{\begin{array}{c}(h, \vec{V} \Rightarrow V) \in \Gamma \cup \overline{\Theta} \\ h <_{\mathbb{F}} \mathsf{f} \vee (h \simeq_{\mathbb{F}} \mathsf{f} \wedge |\vec{V}| \geq \mathrm{q}^h) \\ \psi : \{\vec{\alpha}^h\} \to \mathtt{A} \cup \{\infty\} \\ (h, \psi) <_{\mathtt{A}} (\mathsf{f}, \varphi) \\ (\forall i) \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi\end{array}}{\Gamma \vdash^{\mathsf{f}}_{\varphi} h\vec{u} : V\psi}$$

$$\text{(lam)}\quad \frac{\Gamma, x : U \vdash^{\mathsf{f}}_{\varphi} v : V}{\Gamma \vdash^{\mathsf{f}}_{\varphi} \lambda x^U v : U \Rightarrow V} \qquad \text{(sub)}\quad \frac{\Gamma \vdash^{\mathsf{f}}_{\varphi} t : U \quad U \leq V}{\Gamma \vdash^{\mathsf{f}}_{\varphi} t : V}$$

Figure 4: Subtyping rules

$$\text{(size)}\quad \frac{a \leq^{\infty}_{\mathtt{A}} b}{\mathsf{B}_a \leq \mathsf{B}_b} \qquad \text{(prod)}\quad \frac{U' \leq U \quad V \leq V'}{U \Rightarrow V \leq U' \Rightarrow V'}$$

$$\text{(refl)}\quad \frac{}{T \leq T} \qquad \text{(trans)}\quad \frac{T \leq U \quad U \leq V}{T \leq V}$$

In (app-decr), ψ is any size substitution of the size variables of $\vec{V}$ (after Definitions 13 and 16). This rule works like the rule for type instantiation in Hindley-Milner type system [Hin69, Mil78] except that, here, ψ is not a type substitution but a size substitution. Hence, if s is declared of type $\mathsf{N}_\alpha \Rightarrow \mathsf{N}_{\mathsf{s}\alpha}$ then, by (app-decr), $\vdash^{\mathsf{f}}_{\varphi} \mathsf{s} : \mathsf{N}_a \Rightarrow \mathsf{N}_{\mathsf{s}a}$ for any size expression a (in annotated types, size variables are implicitly universally quantified).

The rule (app-decr) is a compact formulation that subsumes in a single rule the usual rules of simply-typed λ-calculus for variables ($\Gamma \vdash^{\mathsf{f}}_{\varphi} x : T$ if $(x, T) \in \Gamma$), constructors and function symbols ($\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{c} : T\psi$ if $(\mathsf{c}, T) \in \overline{\Theta}$ and ψ is any size substitution), and application ($\Gamma \vdash^{\mathsf{f}}_{\varphi} tu : V$ if $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : U \Rightarrow V$ and $\Gamma \vdash^{\mathsf{f}}_{\varphi} u : U$), with the following restrictions on application. First, the head of an application cannot be an abstraction: $\vdash^{\mathsf{f}}_{\varphi}$ only accepts terms in β-normal form since rule right-hand sides usually so are. Second, if an application is headed by a function symbol g, then $\mathsf{g} <_{\mathbb{F}} \mathsf{f}$ (note that $h <_{\mathbb{F}} \mathsf{f}$ whenever $h \in \mathbb{V} \cup \mathbb{C}$), or we have: $\mathsf{g} \simeq_{\mathbb{F}} \mathsf{f}$, g applied to at least q^{g} arguments, and the sizes of the arguments of g, represented by ψ, smaller than φ in $<_{\mathtt{A}}$.

Hence, in (app-decr), h is either a variable of Γ, in which case $\vec{V} \Rightarrow V$ is the type of h declared in Γ, or a constructor or function symbol, in which case

$\vec{V} \Rightarrow V$ is the annotated type of h declared in $\overline{\Theta}$. In addition, if h is a variable, a constructor symbol or a function symbol strictly smaller than f, then h can be applied to any number of arguments compatible with its type. On the other hand, if h is a function symbol equivalent to f, then it must be applied to at least q^h arguments, and the abstract sizes of these arguments, given by the size substitution ψ, must be strictly smaller than φ in $<_{\mathtt{A}}$.

An equivalent less compact formulation of (app-decr) is given in Figure 5.

Figure 5: Typing rules equivalent to (app-decr)

$$\text{(var)}\quad \frac{(x,U)\in\Gamma}{\Gamma\vdash^{\mathsf{f}}_{\varphi} x : U} \qquad \text{(cons)}\quad \frac{\begin{array}{c}(\mathsf{c},\vec{V}\Rightarrow V)\in\overline{\Theta}\\ \psi:\{\vec{\alpha}^{\mathsf{c}}\}\to \mathtt{A}\cup\{\infty\}\\ (\forall i)\Gamma\vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi\end{array}}{\Gamma\vdash^{\mathsf{f}}_{\varphi} \mathsf{c}\vec{u} : V\psi}$$

$$\text{(prec)}\quad \frac{\begin{array}{c}(\mathsf{g},\vec{V}\Rightarrow V)\in\overline{\Theta}\\ \mathsf{g}<_{\mathbb{F}}\mathsf{f}\\ \psi:\{\vec{\alpha}^{\mathsf{g}}\}\to \mathtt{A}\cup\{\infty\}\\ (\forall i)\Gamma\vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi\end{array}}{\Gamma\vdash^{\mathsf{f}}_{\varphi} \mathsf{g}\vec{u} : V\psi} \qquad \text{(rec-call)}\quad \frac{\begin{array}{c}(\mathsf{g},\vec{V}\Rightarrow V)\in\overline{\Theta}\\ \mathsf{g}\simeq_{\mathbb{F}}\mathsf{f}\wedge|\vec{V}|\geq \mathrm{q}^{\mathsf{g}}\\ \psi:\{\vec{\alpha}^{\mathsf{g}}\}\to \mathtt{A}\cup\{\infty\}\\ (\mathsf{g},\psi)<_{\mathtt{A}}(\mathsf{f},\varphi)\\ (\forall i)\Gamma\vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi\end{array}}{\Gamma\vdash^{\mathsf{f}}_{\varphi} \mathsf{g}\vec{u} : V\psi}$$

Before proving this theorem, let us discuss the conditions and give some examples.

Accessibility. As explained in Section 2.5, not every subterm of a computable term is computable. The definition of computability ensures that all accessible subterms so are. The accessibility condition ensures that each free variable x of the right hand-side is either a parameter or an accessible subterm of a termination argument. Hence, every instance of x is computable if the arguments of f so are. Now, by definition of accessibility, there must be a sort Σ^x with respect to which the size of instances of x are measured. Since x can be instantiated by terms of any size, the type of x should be of the form $\mathrm{Annot}(|U|,\Sigma^x,\alpha^x)$, that is, every occurrence of Σ^x should be annotated by some size variable α^x, and no other sort should be annotated.

Subject reduction and decreasingness. This condition enforces two properties at once. First, the right hand-side has the same type as the left hand-side (subject reduction). Indeed, since the interpretation of a type has to be stable by reduction, there cannot be a rule $\mathsf{f}\vec{l}\to r$ such that the size of r is strictly bigger than the size of $\mathsf{f}\vec{l}$: rewriting cannot increase the size. Second, by rule (app-decr), in every function call $h\vec{t}$, the symbolic size upper bounds ψ of the actual sizes of the termination arguments of h are strictly smaller than

those of $\mathsf{f}\vec{l}$ given by φ.

The relation $\vdash^{\mathsf{f}}_{\varphi}$ is similar to the notion of computability closure introduced in [BJO02, Bla15] except that, when comparing function arguments, it uses the sizes given by the type system instead of the structure of terms. As already mentioned in the introduction, using the size information instead of the structure of terms relates our termination technique to well-founded monotone algebras [MN70, vdP96, Ham06], semantic labeling [Zan95, Ham07] or the notion of size-change principle [LJBA01, Hyv14]. Now, as remarked in [Bla06a, KS07], the notion of computability closure itelf has strong connections with the notion of dependency pair [AG00]. It is also a tool for defining and strengthening the higher-order recursive path ordering [Bla06b, JR07, BJR15]. Finally, all these notions are known to be related in some way: size-change principle and dependency pairs [TG05], semantic labeling and recursive path ordering [KL80], dependency pairs and recursive path ordering [Der13], and size-based termination and semantic labeling [BR09].

Minimality. Since φ provides symbolic *upper bounds* only, this does not suffice for getting termination. We also need φ to be minimal. Indeed, consider the rule $\mathsf{f}\ x \to \mathsf{f}\ x$ with $\mathsf{f} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}_\alpha$ and $\Gamma = [x : \mathsf{N}_x]$. By taking, $\alpha\varphi = \mathsf{s}\, x$, one has $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{f}\ x : \mathsf{N}_x$ since $\Gamma \vdash^{\mathsf{f}}_{\varphi} x : \mathsf{N}_x$ and $x <_{\mathtt{A}} \mathsf{s}\ x$.

Monotony. Finally, the monotony condition requires the size of terms generated by f to be monotone wrt. the sizes of its termination arguments. It can always be satisfied by taking $\sigma^{\mathsf{f}} = \infty$. This condition also appears in [Abe04, BFG$^+$04]. It is necessary because, in the rule (app-decr), ψ is not necessarily minimal: it may be set to a *strict* upper bound by using the rule (sub) beforehand. This could lead to invalid deductions wrt. sizes. Take for instance the subtraction on natural numbers $\mathsf{sub} : \mathsf{N} \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}$ defined by the rules of Figure 1. Because of the monotony condition, we cannot take $\mathsf{sub} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}_\beta \Rightarrow \mathsf{N}_{\alpha-\beta}$. Otherwise, given $\mathsf{f} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}$ and $\mathsf{0} : \mathsf{N}_\mathsf{0}$, the rule $\mathsf{f}\ (\mathsf{s}\ \mathsf{0}) \to \mathsf{f}\ (\mathsf{sub}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{0})$ would satisfy the other conditions. Indeed, by taking $\mathsf{sub} <_{\mathbb{F}} \mathsf{f}$ and $\psi = \{(\alpha, \mathsf{s}\,\mathsf{0}), (\beta, \mathsf{s}\,\mathsf{0})\}$, one gets $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{sub}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{0} : \mathsf{N}_\mathsf{0}$ (while $o_{\mathcal{D}^{\mathsf{N}}}(\mathsf{sub}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{0}) = 1$) and $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{f}\ (\mathsf{sub}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{0}) : \mathsf{N}$ since $\mathsf{0} <_{\mathtt{A}} \mathsf{s}\,\mathsf{0}$, but the system does not terminate since $\mathsf{f}\ (\mathsf{sub}\ (\mathsf{s}\ \mathsf{0})\ \mathsf{0}) \to \mathsf{f}\ (\mathsf{s}\ \mathsf{0})$.

The accessibility and monotony conditions can be easily checked. In the next section, we study the decidability of $\vdash^{\mathsf{f}}_{\varphi}$ and, in Section 8, we provide a syntactic condition for the minimality condition to hold in the successor algebra.

To end this section, note that the termination conditions do not require l itself to be typable in $\vdash^{\mathsf{f}}_{\varphi}$. Hence, for instance, assuming that B has two constructors $\mathsf{c} : \mathsf{B}_\alpha \Rightarrow \mathsf{B}_{\mathsf{s}\alpha}$ and $\mathsf{b} : \mathsf{B}_\alpha \Rightarrow \mathsf{B}_\alpha \Rightarrow \mathsf{B}_{\mathsf{s}\alpha}$, we can handle the rule $\mathsf{f}\,(\mathsf{b}\,x_1\,(\mathsf{c}\,x_2)) \to \mathsf{f}\,x_2$ by taking $\Gamma = [x_2 : \mathsf{B}_{\alpha^{x_2}}]$ and $\alpha^{\mathsf{f}}_1\varphi = \mathsf{s}\alpha^{x_2}$. On the other hand, we cannot handle the rule $\mathsf{f}\,(\mathsf{b}\,x_1\,(\mathsf{c}\,x_2)) \to \mathsf{f}\,(\mathsf{b}\,x_1\,x_2)$. Indeed, in this case, we can have $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{b}\,x_1\theta\,x_2\theta) = o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{b}\,x_1\theta\,(\mathsf{c}\,x_2\theta))$ if $o(x_2\theta) < o(x_1\theta)$: the height is not a decreasing measure in this case.

4.3 Examples

We now show, for various examples, how to check these conditions, except the minimality condition whose verification is postponed to Section 8.

We will use the following sorts and constructors:

- B: the sort of booleans with the constructors $\mathsf{true} : \mathsf{B}$ and $\mathsf{false} : \mathsf{B}$;
- N: the sort of natural numbers with the constructors $\mathsf{0} : \mathsf{N}$ and $\mathsf{s} : \mathsf{N} \Rightarrow \mathsf{N}$;
- O: the sort of Howard's constructive ordinals with the constructors $\mathsf{zero} : \mathsf{O}$, $\mathsf{succ} : \mathsf{O} \Rightarrow \mathsf{O}$ and $\mathsf{lim} : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow \mathsf{O}$;
- L: the sort of lists with the constructors $\mathsf{nil} : \mathsf{L}$ and $\mathsf{cons} : \mathsf{L} \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}$[6];

with $\mathsf{N} <_{\mathbb{S}} \mathsf{L}$ and $\mathsf{N} <_{\mathbb{S}} \mathsf{O}$.

Example 2 (Division) Consider the function symbols sub (substraction) and div (division) both of type $\mathsf{N} \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}$ defined by the rules of Figure 1.

One can easily check the accessibility condition.

For constructors, take in the successor algebra, $\mathsf{0} : \mathsf{N}$ and $\mathsf{s} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}_{\mathsf{s}\alpha}$. For sub and div, take $\mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}_\alpha$ with $\mathrm{q}^{\mathsf{sub}} = \mathrm{q}^{\mathsf{div}} = 1$ (and thus $\Sigma_1^{\mathsf{sub}} = \Sigma_1^{\mathsf{div}} = \mathsf{N}$ and $\alpha_1^{\mathsf{sub}} = \alpha_1^{\mathsf{div}} = \alpha$), which expresses the fact that these functions are not size-increasing. One can easily check the monotony condition.

For the subject reduction and decreasingness condition, we only detail the case of the last rule of div by taking $\Gamma = [x : \mathsf{N}_x, y : \mathsf{N}]$, $\varphi = \{(\alpha, \mathsf{s}\, x)\}$, $\mathsf{sub} <_{\mathbb{F}} \mathsf{div}$ and the identity for ζ^{div}. Let $\vdash = \vdash_\varphi^{\mathsf{div}}$. By (var), $\Gamma \vdash x : \mathsf{N}_x$ and $\Gamma \vdash y : \mathsf{N}$. By (prec), $\Gamma \vdash \mathsf{sub}\ x\ y : \mathsf{N}_x$ (by taking $\psi = \{(\alpha, x)\}$). By (cons), $\Gamma \vdash \mathsf{s}\ y : \mathsf{N}$ (by taking $\psi = \{(\alpha, \infty)\}$). By (app-decr), $\Gamma \vdash \mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y) : \mathsf{N}_x$ since $x <_{\mathtt{A}} \mathsf{s}\, x$. Finally, by (cons), $\Gamma \vdash \mathsf{s}\ (\mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y)) : \mathsf{N}_{\mathsf{s}x} = \mathsf{N}_\alpha\varphi$. ■

Example 3 (Map and filter) Consider the function symbols $\mathsf{map} : \mathsf{L} \Rightarrow (\mathsf{N} \Rightarrow \mathsf{N}) \Rightarrow \mathsf{L}$,[7] $\mathsf{if} : \mathsf{L} \Rightarrow \mathsf{L} \Rightarrow \mathsf{B} \Rightarrow \mathsf{L}$ and $\mathsf{filter} : \mathsf{L} \Rightarrow (\mathsf{N} \Rightarrow \mathsf{B}) \Rightarrow \mathsf{L}$ defined by the rules:

$$\begin{array}{rcl}
\mathsf{map}\ \mathsf{nil}\ f & \to & \mathsf{nil} \\
\mathsf{map}\ (\mathsf{cons}\ l\ x)\ f & \to & \mathsf{cons}\ (\mathsf{map}\ l\ f)\ (f\ x) \\
\mathsf{if}\ x\ y\ \mathsf{true} & \to & x \\
\mathsf{if}\ x\ y\ \mathsf{false} & \to & y \\
\mathsf{filter}\ \mathsf{nil}\ f & \to & \mathsf{nil} \\
\mathsf{filter}\ (\mathsf{cons}\ l\ x)\ f & \to & \mathsf{if}\ (\mathsf{cons}\ (\mathsf{filter}\ l\ f)\ x)\ (\mathsf{filter}\ l\ f)\ (f\ x)
\end{array}$$

For annotated types, we could take in the successor algebra, $\mathsf{true} : \mathsf{B}$, $\mathsf{false} : \mathsf{B}$, $\mathsf{nil} : \mathsf{L}$, $\mathsf{cons} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}_{\mathsf{s}\alpha}$, $\mathsf{map} : \mathsf{L}_\alpha \Rightarrow (\mathsf{N} \Rightarrow \mathsf{N}) \Rightarrow \mathsf{L}_\alpha$ with $\mathrm{q}^{\mathsf{map}} = 1$ (and thus $\Sigma_1^{\mathsf{map}} = \mathsf{L}$ and $\alpha_1^{\mathsf{map}} = \alpha$), $\mathsf{if} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{L}_\alpha$ with $\mathrm{q}^{\mathsf{if}} = 2$ (and thus $\Sigma_1^{\mathsf{if}} = \Sigma_2^{\mathsf{if}} = \mathsf{L}$ and $\alpha_1^{\mathsf{f}} = \alpha_2^{\mathsf{f}} = \alpha$), and $\mathsf{filter} : \mathsf{L}_\alpha \Rightarrow (\mathsf{N} \Rightarrow \mathsf{B}) \Rightarrow \mathsf{L}_\alpha$ with

[6] We permuted the arguments of cons so that the typing conforms to Definition 4.

[7] We permuted the arguments so that the typing conforms to Definition 16.

$\mathrm{q}^{\mathsf{filter}} = 1$, expressing the fact that these functions are not size-increasing. One can easily check the monotony condition.

Unfortunately, the annotated type of if does not satisfy the conditions of Definition 16 because $\alpha_1^{\mathsf{if}} = \alpha_2^{\mathsf{if}}$ (the variables α_i^{if} should be distinct). There are however two solutions to get around this problem:

- Annotate if in the max-successor algebra by taking $\mathsf{if} : \mathsf{B} \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\beta \Rightarrow \mathsf{L}_{\mathtt{max}\alpha\beta}$.
- Introduce a new type $\mathsf{C} >_{\mathbb{S}} \mathsf{L}$ with constructor $\mathsf{cond} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}_\alpha$, a new function symbol $\mathsf{newif} : \mathsf{C}_\alpha \Rightarrow \mathsf{L}_\alpha$ with $\mathrm{q}^{\mathsf{newif}} = 1$, and define newif and filter by the following rules instead:

$$\begin{array}{rcl}
\mathsf{newif}\ (\mathsf{cond}\ x\ y\ \mathsf{true}) & \to & x \\
\mathsf{newif}\ (\mathsf{cond}\ x\ y\ \mathsf{false}) & \to & y \\
\mathsf{filter}\ \mathsf{nil}\ f & \to & \mathsf{nil} \\
\mathsf{filter}\ (\mathsf{cons}\ l\ x)\ f & \to & \mathsf{newif}\ (\mathsf{cond}\ (\mathsf{cons}\ (\mathsf{filter}\ l\ f)\ x)\ (\mathsf{filter}\ l\ f)\ (f\ x))
\end{array}$$

For subject reduction and decreasingness, we only detail the case of the last rule of filter by taking $\Gamma = [f : \mathsf{N} \Rightarrow \mathsf{B}, x : \mathsf{N}, l : \mathsf{L}_l]$, $\varphi = \{(\alpha, \mathtt{s}\, l)\}$, $\mathsf{newif} <_{\mathbb{F}} \mathsf{filter}$, $\mathsf{cond} <_{\mathbb{F}} \mathsf{filter}$ and the identity for ζ^{filter}. Let $\vdash = \vdash_\varphi^{\mathsf{filter}}$. By (var), $\Gamma \vdash x : \mathsf{N}$, $\Gamma \vdash l : \mathsf{L}_l$ and $\Gamma \vdash f\ x : \mathsf{B}$. By (app-decr), $\Gamma \vdash \mathsf{filter}\ l\ f : \mathsf{L}_l$ since $l <_{\mathtt{A}} \mathtt{s}\, l$. By (cons), $\Gamma \vdash \mathsf{cons}\ (\mathsf{filter}\ l\ f)\ x : \mathsf{L}_{\mathtt{s}l}$. By (sub), $\Gamma \vdash \mathsf{filter}\ l\ f : \mathsf{L}_{\mathtt{s}l}$ since $l \leq_{\mathtt{A}}^{\infty} \mathtt{s}\, l$. By (prec), $\Gamma \vdash \mathsf{cond}\ (\mathsf{cons}\ (\mathsf{filter}\ l\ f)\ x)\ (\mathsf{filter}\ l\ f)\ (f\ x) : \mathsf{L}_{\mathtt{s}l}$. So, by (prec), $\Gamma \vdash \mathsf{newif}\ (\mathsf{cond}\ (\mathsf{cons}\ (\mathsf{filter}\ l\ f)\ x)\ (\mathsf{filter}\ l\ f)\ (f\ x)) : \mathsf{L}_{\mathtt{s}l} = \mathsf{L}_\alpha\varphi$. ■

Example 4 (Gödel' system T and Howard' system V) Consider the recursor on natural numbers $\mathsf{rec}_T^{\mathsf{N}} : \mathsf{N} \Rightarrow T \Rightarrow (\mathsf{N} \Rightarrow T \Rightarrow T) \Rightarrow T$ from Gödel' system T [Göd58], and the recursor on ordinals $\mathsf{rec}_T^{\mathsf{O}} : \mathsf{O} \Rightarrow T \Rightarrow (\mathsf{O} \Rightarrow T \Rightarrow T) \Rightarrow ((\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T) \Rightarrow T$ from Howard' system V [How72] defined by the following rules:

$$\begin{array}{rcl}
\mathsf{rec}_T^{\mathsf{N}}\ 0\ u\ v & \to & u \\
\mathsf{rec}_T^{\mathsf{N}}\ (\mathsf{s}\ x)\ u\ v & \to & v\ x\ (\mathsf{rec}_T^{\mathsf{N}}\ x\ u\ v)
\end{array}$$

$$\begin{array}{rcl}
\mathsf{rec}_T^{\mathsf{O}}\ 0\ u\ v\ w & \to & u \\
\mathsf{rec}_T^{\mathsf{O}}\ (\mathsf{succ}\ x)\ u\ v\ w & \to & v\ x\ (\mathsf{rec}_T^{\mathsf{O}}\ x\ u\ v\ w) \\
\mathsf{rec}_T^{\mathsf{O}}\ (\mathsf{lim}\ f)\ u\ v\ w & \to & w\ f\ (\lambda n^{\mathsf{N}}.\mathsf{rec}_T^{\mathsf{O}}\ (f\ n)\ u\ v\ w)
\end{array}$$

One can easily check the accessibility condition.

For annotated types, take in the successor algebra, $0 : \mathsf{N}$, $\mathsf{s} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}_{\mathsf{s}\alpha}$, $\mathsf{zero} : \mathsf{O}$, $\mathsf{succ} : \mathsf{O}_\alpha \Rightarrow \mathsf{o}_{\mathsf{s}\alpha}$, $\mathsf{lim} : (\mathsf{N} \Rightarrow \mathsf{O}_\alpha) \Rightarrow \mathsf{O}_{\mathsf{s}\alpha}$, $\mathsf{rec}_T^{\mathsf{N}} : \mathsf{N}_\alpha \Rightarrow T \Rightarrow (\mathsf{N} \Rightarrow T \Rightarrow T) \Rightarrow T$ and $\mathsf{rec}_T^{\mathsf{O}} : \mathsf{O}_\alpha \Rightarrow T \Rightarrow (\mathsf{O} \Rightarrow T \Rightarrow T) \Rightarrow ((\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T) \Rightarrow T$. One can easily check the monotony condition.

For subject reduction and decreasingness, we only detail the case of the last rule of $\mathsf{f} = \mathsf{rec}_T^{\mathsf{O}}$ by taking $\Gamma = [f : \mathsf{N} \Rightarrow \mathsf{O}_\beta, u : T, v : \mathsf{O} \Rightarrow T \Rightarrow T, w : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T]$, $\varphi = \{(\alpha, \mathsf{s}\, \beta)\}$ and the identity for ζ^{f}. Let $\vdash = \vdash_\varphi^{\mathsf{f}}$ and $\Delta = [n : \mathsf{N}]\Gamma$. By (var), $\Gamma \vdash f : \mathsf{N} \Rightarrow \mathsf{O}_\beta$ and $\Delta \vdash f : \mathsf{N} \Rightarrow \mathsf{O}_\beta$,

$\Gamma \vdash u : T$ and $\Delta \vdash u : T$, $\Gamma \vdash v : \mathsf{O} \Rightarrow T \Rightarrow T$ $\Delta \vdash v : \mathsf{O} \Rightarrow T \Rightarrow T$, $\Gamma \vdash w : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T$ and $\Delta \vdash w : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T$. By (var) again, $\Delta \vdash f\ n : \mathsf{O}_\beta$. By (app-decr), $\Delta \vdash \mathsf{rec}^{\mathsf{O}}_T\ (f\ n)\ u\ v\ w : T$ since $\beta <_{\mathtt{A}} \mathtt{s}\,\beta$. By (lam), $\Gamma \vdash \lambda n^{\mathsf{N}}.\mathsf{rec}^{\mathsf{O}}_T\ (f\ n)\ u\ v\ w : \mathsf{N} \Rightarrow T$. By (sub), $\Gamma \vdash f : \mathsf{N} \Rightarrow \mathsf{O}$ since $\mathsf{N} \Rightarrow \mathsf{O}_\beta \leq \mathsf{N} \Rightarrow \mathsf{O}$. Finally, by (var), $\Delta \vdash w\ f\ (\lambda n^{\mathsf{N}}.\mathsf{rec}^{\mathsf{O}}_T\ (f\ n)\ u\ v\ w) : T$. ■

Follows a similar example showing that we cannot capture all non-increasing functions as one can easily expect:

Example 5 (Quicksort) Let P be the sort of pairs of lists with the constructor $\mathsf{pair} : \mathsf{L} \Rightarrow \mathsf{L} \Rightarrow \mathsf{P}$, and C be the sort with the constructor $\mathsf{cond} : \mathsf{P} \Rightarrow \mathsf{P} \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}$. Then, let the functions $\mathsf{fst}, \mathsf{snd} : \mathsf{P} \Rightarrow \mathsf{L}$, $\mathsf{le} : \mathsf{N} \Rightarrow \mathsf{N} \Rightarrow \mathsf{B}$, $\mathsf{if} : \mathsf{C} \Rightarrow \mathsf{P}$, $\mathsf{pivot} : \mathsf{L} \Rightarrow \mathsf{N} \Rightarrow \mathsf{P}$, $\mathsf{qs} : \mathsf{L} \Rightarrow \mathsf{L} \Rightarrow \mathsf{L}$ and $\mathsf{qsort} : \mathsf{L} \Rightarrow \mathsf{L}$ be defined by the rules:

$$\begin{array}{rcl} \mathsf{fst}\ (\mathsf{pair}\ l\ m) & \to & l \\ \mathsf{snd}\ (\mathsf{pair}\ l\ m) & \to & m \\ \mathsf{if}\ (\mathsf{cond\ true}\ p\ q) & \to & p \\ \mathsf{if}\ (\mathsf{cond\ false}\ p\ q) & \to & q \end{array} \qquad \begin{array}{rcl} \mathsf{le}\ 0\ y & \to & \mathsf{true} \\ \mathsf{le}\ (\mathsf{s}\ x)\ 0 & \to & \mathsf{false} \\ \mathsf{le}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) & \to & \mathsf{le}\ x\ y \end{array}$$

$$\begin{array}{rcl} \mathsf{pivot\ nil}\ x & \to & \mathsf{pair\ nil\ nil} \\ \mathsf{pivot}\ (\mathsf{cons}\ l\ y)\ x & \to & \mathsf{if}\ (\mathsf{cond}\ (\mathsf{pair}\ (\mathsf{cons}\ p_1\ y)\ p_2)\ (\mathsf{pair}\ p_1\ (\mathsf{cons}\ p_2\ y))\ (\mathsf{le}\ y\ x)) \\ & & \text{where } p_1 = \mathsf{fst}\ p, p_2 = \mathsf{snd}\ p, p = \mathsf{pivot}\ l\ x \\ \mathsf{qs\ nil}\ l & \to & l \\ \mathsf{qs}\ (\mathsf{cons}\ l\ x)\ m & \to & \mathsf{qs}\ p_1\ (\mathsf{cons}\ (\mathsf{qs}\ p_2\ m)\ x) \\ & & \text{where } p_1 = \mathsf{fst}\ p, p_2 = \mathsf{snd}\ p, p = \mathsf{pivot}\ x\ l \\ \mathsf{qsort}\ l & \to & \mathsf{qs}\ l\ \mathsf{nil} \end{array}$$

Again, one can easily check the accessibility condition.

Now, for the annotated types of constructors, we take in the successor algebra $\mathsf{true} : \mathsf{B}$, $\mathsf{false} : \mathsf{B}$, $0 : \mathsf{N}$, $\mathsf{s} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}_{\mathtt{s}\alpha}$, $\mathsf{nil} : \mathsf{L}$, $\mathsf{cons} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}_{\mathtt{s}\alpha}$, $\mathsf{pair} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{P}_\alpha$ and $\mathsf{cond} : \mathsf{P}_\alpha \Rightarrow \mathsf{P}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{C}_\alpha$. Hence, a term of type P_α is a pair of lists of length smaller than or equal to α.

Now, for function symbols, we take $\mathsf{fst}, \mathsf{snd} : \mathsf{P}_\alpha \Rightarrow \mathsf{L}_\alpha$, $\mathsf{if} : \mathsf{C}_\alpha \Rightarrow \mathsf{P}_\alpha$, $\mathsf{le} : \mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{B}$, $\mathsf{pivot} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{P}_\alpha$, $\mathsf{qs} : \mathsf{L}_\alpha \Rightarrow \mathsf{L} \Rightarrow \mathsf{L}$ and $\mathsf{qsort} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}$, which expresses the fact that fst, snd, if and pivot are not size-increasing. Again, one can easily check the monotony condition.

For the subject reduction and decreasingness condition, we only detail the case of the last rule of qs by taking $\Gamma = [x : \mathsf{N}, l : \mathsf{L}_l, m : \mathsf{L}]$, $\varphi = \{(\alpha, \mathtt{s}\,l)\}$, $\mathsf{fst}, \mathsf{snd}, \mathsf{pivot} <_{\mathbb{F}} \mathsf{qs}$ and the identity for ζ^{qs}. Let $\vdash = \vdash^{\mathsf{qs}}_{\varphi}$. By (var), $\Gamma \vdash x : \mathsf{N}$, $\Gamma \vdash l : \mathsf{L}_l$ and $\Gamma \vdash m : \mathsf{L}$. By (prec), $\Gamma \vdash p : \mathsf{P}_l$, $\Gamma \vdash p_1 : \mathsf{L}_l$ and $\Gamma \vdash p_2 : \mathsf{L}_l$. Since $l <_{\mathtt{A}} \mathtt{s}\,l$, by (app-decr), $\Gamma \vdash \mathsf{qs}\ p_2\ m : \mathsf{L}$. By (cons), $\Gamma \vdash \mathsf{cons}\ (\mathsf{qs}\ v\ m)\ x : \mathsf{L}$. Finally, since $l <_{\mathtt{A}} \mathtt{s}\,l$, by (app-decr) again, $\Gamma \vdash \mathsf{qs}\ u\ (\mathsf{cons}\ (\mathsf{qs}\ v\ m)\ x) : \mathsf{L}$.

Note however that we cannot express that qsort is non-increasing, that is, take $\mathsf{qsort} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha$. To do so, we need a more precise type system with existential quantifiers and constraints on size variables where pivot can be given

the type:

$$(\forall\alpha)\mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow (\exists\beta)(\exists\gamma)(\alpha = \beta + \gamma)\mathsf{L}_\beta \times \mathsf{L}_\gamma \text{ [BR06].}$$ ■

We now give an example using interpretation functions ζ^{f}_X different from the identity:

Example 6 (Reverse) The reversal of a list can be defined as follows [HH82]:

$$\begin{array}{rcl}
\mathsf{last}\ \mathsf{nil}\ x & \to & x \\
\mathsf{last}\ (\mathsf{cons}\ l\ y)\ x & \to & \mathsf{last}\ l\ y \\
\mathsf{revremlast}\ \mathsf{nil}\ x & \to & \mathsf{nil} \\
\mathsf{revremlast}\ (\mathsf{cons}\ l\ y)\ x & \to & \mathsf{rev}\ (\mathsf{cons}\ (\mathsf{rev}\ (\mathsf{revremlast}\ l\ y))\ x) \\
\mathsf{rev}\ \mathsf{nil} & \to & \mathsf{nil} \\
\mathsf{rev}\ (\mathsf{cons}\ l\ x) & \to & \mathsf{cons}\ (\mathsf{revremlast}\ l\ x)\ (\mathsf{last}\ l\ x)
\end{array}$$

where $\mathsf{rev} : \mathsf{L} \Rightarrow \mathsf{L}$, $\mathsf{revremlast} : \mathsf{L} \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}$ and $\mathsf{last} : \mathsf{L} \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}$.

Consider the 4th rule. Take $\mathsf{cons} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}_{\alpha+1}$, $\mathsf{rev} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha$, $\mathsf{revremlast} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}_\alpha$, $\Gamma = [x : \mathsf{N}, y : \mathsf{N}, l : \mathsf{L}_l]$ and $\varphi = \{(\alpha, l+1)\}$. One can easily check the accessibility and monotony conditions. We now check the subject reduction and decreasingness condition. Let $\vdash = \vdash^{\mathsf{revremlast}}_{\varphi}$. For comparing termination arguments, take $\leq_{\mathtt{A}} = \leq^{\mathrm{ext}}_{\mathtt{A}}$, $\mathsf{rev} \simeq_{\mathbb{F}} \mathsf{revremlast}$, $\zeta^{\mathsf{rev}}(a) = 2a$ and $\zeta^{\mathsf{revremlast}}(a) = 2a + 1$. By (var), $\Gamma \vdash x : \mathsf{N}$, $\Gamma \vdash y : \mathsf{N}$ and $\Gamma \vdash l : \mathsf{L}_l$. By (app-decr), $\Gamma \vdash \mathsf{revremlast}\ l\ y : \mathsf{N}_l$ since $\zeta^{\mathsf{revremlast}}(l) = 2l+1 <_{\mathtt{A}} \zeta^{\mathsf{revremlast}}(l+1) = 2(l+1)+1 = 2l+3$. By (app-decr), $\Gamma \vdash \mathsf{rev}\ (\mathsf{revremlast}\ l\ y) : \mathsf{N}_l$ since $\zeta^{\mathsf{rev}}(l) = 2l < 2l+3$. By (cons), $\Gamma \vdash \mathsf{cons}\ (\mathsf{rev}\ (\mathsf{revremlast}\ l\ y))\ x : \mathsf{L}_{l+1}$. Finally, by (app-decr) again, $\Gamma \vdash \mathsf{rev}\ (\mathsf{cons}\ (\mathsf{rev}\ (\mathsf{revremlast}\ l\ y))\ x) : \mathsf{L}_{l+1}$ since $\zeta^{\mathsf{rev}}(l+1) = 2l+2 < 2l+3$. ■

Here is an example of a recursor for a non-strictly positive type the termination of which can be proved neither by HORPO [JR99] nor by CPO [BJR15]:

Example 7 (Continuations) One can implement a breadth-first search algorithm on leaf-labeled binary trees by using a non-strictly positive type of continuations C whose constructors are $\mathsf{b} : \mathsf{C}$ and $\mathsf{c} : \neg\neg\mathsf{C} \Rightarrow \mathsf{C}$, where $\neg T = T \Rightarrow \mathsf{L}$ and L is the sort for lists of labels [Hof95]. Its recursor for type T, $\mathsf{rec}^{\mathsf{C}}_T : \mathsf{C} \Rightarrow T \Rightarrow (\neg\neg\mathsf{C} \Rightarrow \neg\neg T \Rightarrow T) \Rightarrow T$, can be defined by the rules:

$$\begin{array}{rcl}
\mathsf{rec}^{\mathsf{C}}_T\ \mathsf{b}\ u\ v & \to & u \\
\mathsf{rec}^{\mathsf{C}}_T\ (\mathsf{c}\ x)\ u\ v & \to & v\ x\ (\lambda y^{\neg A} x\ (\lambda z^{\mathsf{C}} y\ (\mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v)))
\end{array}$$

Again, one can easily check the accessibility condition.

Now, take $\mathsf{c} : \neg\neg\mathsf{C}_\alpha \Rightarrow \mathsf{C}_{\mathsf{s}\alpha}$ and $\mathsf{f} = \mathsf{rec}^{\mathsf{C}}_T : \mathsf{C}_\alpha \Rightarrow A \Rightarrow (\neg\neg\mathsf{C} \Rightarrow \neg\neg A \Rightarrow A) \Rightarrow A$. One can easily check the monotony condition.

For the subject reduction and decreasingness condition, we only detail the case of the second rule by taking $\Gamma = [x : \neg\neg\mathsf{C}_x, u : A, v : \neg\neg\mathsf{C} \Rightarrow \neg\neg A \Rightarrow A]$ and $\varphi = \{(\alpha, \mathsf{s}\,x)\}$. Let $\Gamma_y = [y : \neg A]\Gamma$, $\Gamma_{yz} = [z : \mathsf{C}_x]\Gamma_y$ and $\vdash = \vdash^{\mathsf{f}}_{\varphi}$. By (var), $\Gamma \vdash x : \neg\neg\mathsf{C}_x$, $\Gamma \vdash u : A$, $\Gamma \vdash v : \neg\neg\mathsf{C} \Rightarrow \neg\neg A \Rightarrow A$, $\Gamma_{yz} \vdash y : \neg A$ and

$\Gamma_{yz} \vdash z : \mathsf{C}_x$. Since $x <_{\mathtt{A}} \mathsf{s}\, x$, by (app-decr), $\Gamma_{yz} \vdash \mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v : A$. By (var), $\Gamma_{yz} \vdash y\ (\mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v) : \mathsf{L}$. By (lam), $\Gamma_y \vdash \lambda zy\ (\mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v) : \neg\mathsf{C}_x$. By (var), $\Gamma_y \vdash x\ (\lambda zy\ (\mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v)) : \mathsf{L}$. By (lam), $\Gamma \vdash \lambda y^{\neg A} x\ (\lambda zy\ (\mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v)) : \neg\neg A$. By (sub), $\Gamma \vdash x : \neg\neg\mathsf{C}$. Thus, by (var), $\Gamma \vdash v\ x\ (\lambda y^{\neg A} x\ (\lambda zy\ (\mathsf{rec}^{\mathsf{C}}_T\ z\ u\ v))) : A$. ■

We end this series of examples with one using non-standard constructor size annotations:

Example 8 (Normalization of conditionals) Let C be the sort of conditional expressions with the constructors $\mathsf{at} : \mathsf{C}$ and $\mathsf{if} : \mathsf{C}^3 \Rightarrow \mathsf{C}$. Following [BM79], one can define a normalization function $\mathsf{nm} : \mathsf{C} \Rightarrow \mathsf{C}$ as follows:

$$\begin{array}{rcl} \mathsf{nm}\ \mathsf{at} & \to & \mathsf{at} \\ \mathsf{nm}\ (\mathsf{if}\ \mathsf{at}\ y\ z) & \to & \mathsf{if}\ \mathsf{at}\ (\mathsf{nm}\ y)\ (\mathsf{nm}\ z) \\ \mathsf{nm}\ (\mathsf{if}\ (\mathsf{if}\ u\ v\ w)\ y\ z) & \to & \mathsf{nm}\ (\mathsf{if}\ u\ (\mathsf{nm}\ (\mathsf{if}\ v\ y\ z))\ (\mathsf{nm}\ (\mathsf{if}\ w\ y\ z))) \end{array}$$

In [Pau86] is given a measure on terms due to Shostak that is decreasing in recursive calls. Hence, we can prove the termination of nm by using the following annotated types: $\mathsf{at} : \mathsf{C}$, $\mathsf{if} : \mathsf{C}_\alpha \Rightarrow \mathsf{C}_\beta \Rightarrow \mathsf{C}_\gamma \Rightarrow \mathsf{C}_{(\alpha+1)(\beta+\gamma+3)}$ and $\mathsf{nm} : \mathsf{C}_\alpha \Rightarrow \mathsf{C}_\alpha$. One can easily check the monotony condition.

Now, for the 3rd rule, let $\Gamma = [u : \mathsf{C}_u, v : \mathsf{C}_v, w : \mathsf{C}_w, y : \mathsf{C}_y, z : \mathsf{C}_z]$, $\varphi = \{(\alpha, a)\}$ where $a = ((u+1)(v+w+3)+1)(y+z+3)) = uvy+uvz+uwy+uwz+3uv+3uw+3uy+3uz+vy+wy+vz+wz+9u+3v+3w+4y+4z+12$, and ζ^{nm} be the identity. One can easily check the accessibility condition. We now check the subject reduction and decreasingness condition. Let $\vdash = \vdash^{\mathsf{nm}}_{\varphi}$. By (cons), $\Gamma \vdash \mathsf{if}\, v\, y\, z : \mathsf{C}_{(v+1)(y+z+3)}$ and $\Gamma \vdash \mathsf{if}\, w\, y\, z : \mathsf{C}_{(w+1)(y+z+3)}$. By (app-decr), $\Gamma \vdash \mathsf{nm}\,(\mathsf{if}\, v\, y\, z) : \mathsf{C}_{(v+1)(y+z+3)}$ since $(v+1)(y+z+3) = vy+vz+y+z+3 <_{\mathtt{A}} a$, and $\Gamma \vdash \mathsf{nm}\,(\mathsf{if}\, w\, y\, z) : \mathsf{C}_{(w+1)(y+z+3)}$ since $(w+1)(y+z+3) = wy+wz+y+z+3 <_{\mathtt{A}} a$. Finally, by (app-decr), $\Gamma \vdash \mathsf{nm}\,(\mathsf{if}\, u\,(\mathsf{nm}\,(\mathsf{if}\, v\, y\, z))\,(\mathsf{nm}\,(\mathsf{if}\, w\, y\, z))) : \mathsf{C}_b$ where $b = (u+1)((v+1)(y+z+3)+(w+1)(y+z+3)+3)$ since $b = uvy+uvz+uwy+uwz+2uy+2uz+vy+vz+wy+wz+9u+2y+2z+9 <_{\mathtt{A}} a$. Therefore, by (sub), $\Gamma \vdash \mathsf{nm}\,(\mathsf{if}\, u\,(\mathsf{nm}\,(\mathsf{if}\, v\, y\, z))\,(\mathsf{nm}\,(\mathsf{if}\, w\, y\,))) : \mathsf{C}_a$. ■

4.4 Proof of Theorem 2

We now prove Theorem 2 by checking that every term is computable, that is, $t \in T$ whenever $\Gamma \vdash t : T$.

Computability of constructors. We first prove that, for all $(\mathsf{c}, \mu, \vec{t})$ with $\Theta(\mathsf{c}) = T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}} \Rightarrow \mathsf{B}$ and $\overline{\Theta}(\mathsf{c}) = \overline{T_1} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}}} \Rightarrow \mathsf{B}_\sigma$ as in Definition 13 (we drop the c's in exponents), $|\vec{t}| = \mathrm{r}$ and $(\forall i) t_i \in \overline{T_i}\mu$, we have $\mathsf{c}\vec{t} \in \mathsf{B}_\sigma\mu$, by induction on $\vec{t}$ with $\leftarrow_{\mathrm{prod}}$ as well-founded relation. First, we have $\mathsf{c}\vec{t} \in \mathrm{SN}$ since $\vec{t} \in \mathrm{SN}$ and there is no rule of the form $\mathsf{c}\vec{l} \to r$. Second, for every accessible argument $i \in \{1, \ldots, \mathrm{q}\}$, we have $\overline{T_i}\mu \subseteq T_i$ since $\overline{T_i} = \mathrm{Annot}(T_i, \Sigma_i, \alpha_i)$ and $\mathrm{Pos}(\Sigma_i, T_i) \subseteq \mathrm{Pos}^+(T_i)$. Moreover, $o_{\mathcal{S}^i}(t_i) \leq \alpha_i\mu$ since $t_i \in \overline{T_i}\mu$. Therefore, $\mathsf{c}\vec{t} \in \mathsf{B}$. If $\sigma = \infty$, then we are done. Otherwise, we are left to prove that $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\vec{t}) \leq \sigma\mu$. If $\mathsf{c}\vec{t} \to u$ then $u = \mathsf{c}\vec{u}$ with $\vec{t} \to_{\mathrm{prod}} \vec{u}$ and, by induction

hypothesis, $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\vec{u}) \leq \sigma\mu$. Now, if $i \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, then $o_{\mathcal{S}^i}(t_i) \leq \alpha_i\mu < \sigma\mu$ since σ is strictly extensive on recursive arguments. Finally, $\Sigma(o_{\mathcal{S}}(\vec{t})) = \sigma\nu$ where $\nu = \{(\alpha_i, o_{\mathcal{S}^i}(t_i)) \mid i \in \{1, \ldots, \mathrm{q}\}, \alpha_i \in \mathtt{V}\}$ if the variables α_i are pairwise distinct, and $\nu = \{(\alpha, \sup\{o_{\mathcal{S}^i}(\vec{t}) \mid i \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}\})\}$ if the variables α_i are equal to some variable α. Hence, $\sigma\nu \leq \sigma\mu$ since σ is monotone and $\nu \leq \mu$. Therefore, by Theorem 1, $o_{\mathcal{S}^{\mathsf{B}}}(\mathsf{c}\vec{t}) \leq \sigma\mu$.

Computability of function symbols. We now prove that, for all $(\mathsf{f}, \mu, \vec{t})$ with $\Theta(\mathsf{f}) = T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}} \Rightarrow \mathsf{B}$ and $\overline{\Theta}(\mathsf{f}) = \overline{T_1} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}}} \Rightarrow \mathsf{B}_\sigma$ as in Definition 16 (we drop the f's in exponents), $|\vec{t}| = \mathrm{r}$ and $\vec{t} \in \overline{\vec{T}}\mu$, we have $\mathsf{f}\vec{t} \in \mathsf{B}_\sigma\mu$, by induction on $((\mathsf{f}, \mu), \vec{t})$ with $(<_{\mathfrak{h}}, \leftarrow_{\mathrm{prod}})_{\mathrm{lex}}$ as well-founded relation (1). Since $\mathsf{f}\vec{t}$ is neutral, it suffices to prove that, for all v such that $\mathsf{f}\vec{t} \to v$, we have $v \in \mathsf{B}_\sigma\mu$. There are two cases:

- $v = \mathsf{f}\vec{v}$ and $\vec{t} \to_{\mathrm{prod}} \vec{v}$. By subject reduction, $\vec{v} \in \overline{\vec{T}}\mu$. Therefore, by induction hypothesis, $v \in \mathsf{B}_\sigma\mu$.
- $\vec{t} = \vec{l}\theta\vec{v}$, $\mathsf{f}\vec{l} \to r \in \mathcal{R}$ and $v = r\theta\vec{v}$. By minimality, there is ν such that $\alpha_i\varphi\nu = o_{\mathcal{S}^{\Sigma_i}}(l_i\theta)$ and $o_{[\Sigma^x:\mathcal{S}^{\Sigma^x}]|U|}(x\theta) \leq \alpha^x\nu$ if $(x, U) \in \Gamma$. Hence, $\varphi\nu \leq \mu$.

  Computability of the matching substitution. We now prove that, for all $(x, U) \in \Gamma$, $x\theta \in U\nu$. There are two cases:

  - $k^x > \mathrm{q}$, $l_{k^x} = x$ and $U = T_{k^x}$. Then, $x\theta = l_{k^x}\theta = t_{k^x}$ and $U\nu = T_{k^x}\nu = T_{k^x}\mu = \overline{T_{k^x}}\mu$. Therefore, $x\theta \in U\nu$.
  - $k^x \leq \mathrm{q}$, $U = \mathrm{Annot}(|U|, \Sigma^x, \alpha^x)$ and $(x, |U|, \Sigma^x) \trianglelefteq_{\mathrm{a}} (l_{k^x}, T_{k^x}, T_{k^x})$. Hence, $(x\theta, |U|, \Sigma^x) \trianglelefteq_{\mathrm{a}} (l_{k^x}\theta, T_{k^x}, T_{k^x})$ and, by Lemma 11, $x\theta \in |U|$. Therefore, $x\theta \in U\nu$ since $U = \mathrm{Annot}(|U|, \Sigma^x, \alpha^x)$, $\mathrm{Pos}(\Sigma^x, |U|) \subseteq \mathrm{Pos}^+(|U|)$ and $o_{[\Sigma^x:\mathcal{S}^{\Sigma^x}]|U|}(x\theta) \leq \alpha^x\nu$.

  Correctness of the computability closure. We now prove that, for all (Γ, t, T, θ), if $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ and $x\theta \in U\nu$ whenever $(x, U) \in \Gamma$, then $t\theta \in T\nu$, by induction on $\vdash^{\mathsf{f}}_{\varphi}$ (2).

(app-decr) If $h \in \mathbb{V}$, this follows by assumption. So, assume that $h \in \mathbb{C} \cup \mathbb{F}$ and $\overline{\Theta}(h) = \vec{V} \Rightarrow V$. By induction hypothesis (2), $u_i \in V_i\psi\nu$. Since $(h, \psi) <_{\mathtt{A}} (\mathsf{f}, \varphi)$, we have $(h, \psi\nu) <_{\mathfrak{h}} (\mathsf{f}, \varphi\nu)$ by assumption on $<_{\mathtt{A}}$. Since $\varphi\nu \leq \mu$, we have $(h, \psi\nu) <_{\mathfrak{h}} (\mathsf{f}, \mu)$ by assumption on $<_{\mathfrak{h}}$. Therefore, by induction hypothesis (1), $h\vec{u} \in V\psi\nu$.

(lam) Wlog. we can assume that $x \notin \mathrm{dom}(\theta) \cup \mathrm{FV}(\theta)$. We have $(\lambda x^U v)\theta = \lambda x^U(v\theta) \in U\nu \Rightarrow V\nu$ because, for all $u \in U\nu$, $(v\theta)\{(x, u)\} \in V\nu$ since $(v\theta)\{(x, u)\} = v\theta'$ where $\theta' = \theta \cup \{(x, u)\}$ and $v\theta' \in V\nu$ by induction hypothesis (2).

(sub) We prove that $U\nu \subseteq V\nu$ whenever $U \leq V$ by induction on $\leq$ (3):

(size) If $b = \infty$, then $\mathsf{B}_a\nu \subseteq \mathsf{B}$ by definition. Otherwise, $a\nu \leq b\nu$ and $\mathsf{B}_a\nu = \mathcal{S}^{\mathsf{B}}_{a\nu} \subseteq \mathcal{S}^{\mathsf{B}}_{b\nu} = \mathsf{B}_b\nu$ since $\mathcal{S}^{\mathsf{B}}$ is monotone.

(prod) Let $t \in U\nu \to V\nu$ and $u' \in U'\nu$. By induction hypothesis (3), $U'\nu \subseteq U\nu$. Hence, $u \in U\nu$ and $tu \in V\nu$. By induction hypothesis (3), $V\nu \subseteq V\nu'$. Therefore, $tu' \in V'\nu$.

(refl) Immediate.

(trans) By induction hypothesis (3) and transitivity of $\subseteq$.

Hence, since $\Gamma \vdash^{\mathsf{f}}_{\varphi} r : U\varphi$ with $U = \overline{T_{|\vec{l}|+1}} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}}} \Rightarrow \mathsf{B}_\sigma$, and $x\theta \in U\nu$ whenever $(x, U) \in \Gamma$, we have $r\theta \in U\varphi\nu$. Hence, $v = r\theta\vec{v} \in \mathsf{B}_\sigma\varphi\nu$. Finally, since $\varphi\nu \leq \mu$ and $\mathrm{Pos}(\alpha_i, \sigma) \subseteq \mathrm{Pos}^+(\sigma)$, we have $v \in \mathsf{B}_\sigma\mu$.

Computability of every well-typed term. Now, it is easy to check that every well-typed term is computable. The proof is similar to (2). We just detail the case of a function symbol f with $\Theta(\mathsf{f}) = T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}} \Rightarrow \mathsf{B}$ and $\overline{\Theta}(\mathsf{f}) = \overline{T_1} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}}} \Rightarrow \mathsf{B}_\sigma$. Let $\vec{t}$ such that $|\vec{t}| = \mathrm{r}$ and $\vec{t} \in \vec{T}$. Let μ be the valuation mapping, for all $i \in \{1, \ldots, \mathrm{q}\}$, α_i to the smallest ordinal $\mathfrak{h}_i$ such that $\mathcal{S}^{T_i}_{\mathfrak{h}_i} = T_i$. Then, $t_i \in \overline{T_i}\mu$ and $\mathsf{f}\vec{t} \in \mathsf{B}_\sigma\mu \subseteq \mathsf{B}$. We conclude by noting that the identity substitution is computable. ■

5 Deciding $\vdash^{\mathsf{f}}_{\varphi}$

In this section, we study the decidability of the relation $\vdash^{\mathsf{f}}_{\varphi}$ used in Theorem 2 and defined in Figures 3 and 4.

The differences between $\vdash^{\mathsf{f}}_{\varphi}$ and the usual typing relation for simply-typed λ-calculus are the followings. First, the set of typable symbols is restricted to those smaller than f. Second, the application of t to u is restricted to the terms t whose head is not an abstraction. Moreover, when the head of t is a symbol equivalent to f, the number of arguments must be bigger than q^{f} and the size of the arguments must be decreasing.

If we remove the decreasingness condition, we get the relation $\vdash^{\mathsf{f}}$ defined by the same rules as those of $\vdash^{\mathsf{f}}_{\varphi}$ except (app-decr) replaced by:

$$\text{(app)}\quad \frac{\begin{array}{c}(h, \vec{V} \Rightarrow V) \in \Gamma \cup \overline{\Theta} \\ h <_{\mathbb{F}} \mathsf{f} \vee (h \simeq_{\mathbb{F}} \mathsf{f} \wedge |\vec{U}| \geq \mathrm{q}^h) \\ \psi : \{\vec{\alpha}^h\} \to \mathtt{A} \cup \{\infty\} \\ (\forall i)\Gamma \vdash^{\mathsf{f}} u_i : V_i\psi\end{array}}{\Gamma \vdash^{\mathsf{f}} h\vec{u} : V\psi}$$

that is, without checking the decreasingness condition $\psi <^{h,\mathsf{f}}_{\mathtt{A}} \varphi$ when $h \simeq_{\mathbb{F}} \mathsf{f}$. Hence, deciding $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ can be reduced to finding a derivation of $\Gamma \vdash^{\mathsf{f}} t : T$ where, at each application of (app), the decreasingness condition is satisfied.

The relation $\vdash^{\mathsf{f}}$ differs from Curry and Feys' typing relation with functional characters or type-schemes (a type with type variables) [CF58] in two points. First, we consider subtyping. Second, our type-schemes are not built from type variables but from size variables. We will however see that some techniques

developed for Curry and Feys' type system or, more generally, Milner's type system [Mil78] and its extensions, can be adapted to our framework.

In [Hin69], Hindley proved that a term is well-typed in Curry and Feys' system iff it has a most general type-scheme that can be computed by using unification [Her30, Rob65], a type-scheme T being more general than another type-scheme U, written $T \sqsubseteq U$, if U is an instance of T, that is, $T\theta = U$ for some substitution θ. This comes from the fact that, if $\Gamma \vdash t : T_1$ and $\Gamma \vdash t : T_2$, then there is T such that $\Gamma \vdash t : T$, $T \sqsubseteq T_1$ and $T \sqsubseteq T_2$. And, as shown by Huet [Hue76], every non-empty subset of types has a greatest lower bound wrt. $\sqsubseteq$. So, in particular $\{T \mid \Gamma \vdash t : T\}$ if t is typable in Γ.

This line of work was later extended in many directions by considering richer types, more complex constructions or by improving the algorithm computing the most general type-scheme. One of the most advanced generalization seems to be Sulzmann's HM(X) system [Sul01], where the type variables of a type-scheme are required to satisfy a formula of an abstract constraint system X. For his system, Sulzmann provides a generic constrained-type inference algorithm assuming a procedure for solving constraints in X. It would be interesting to study whether our framework can fit in this general setting. However, in this paper, we will simply follow Hindley's approach.

Hindley uses a procedure computing the most general unifier of two type-schemes. Unifying two type-schemes T and U simply consists in solving the equation $T = U$ in the algebra of type-schemes, that is, in finding a substitution θ such that $T\theta = U\theta$. In [Hue76], Huet proved that solving $T = U$ is equivalent to finding an $\sqsubseteq$-upper bound to both T and U. He also showed that every non-empty bounded set of types has a least upper bound wrt. $\sqsubseteq$. Hence, every solvable unification problem on types has a most general solution.

In our setting, we have to take into account subtyping. Hence, the following definitions:

Definition 17 (More general type) We say that an annotated type T is *more general than* another annotated type U, written $T \sqsubseteq U$, if there is a substitution θ such that $T\theta$ is a subtype of U, *i.e.* $T\theta \leq U$.

One can easily check that $\sqsubseteq$ is a quasi-ordering.

Definition 18 (Subtyping problem) A *subtyping problem* P is either $\bot$ or a finite set of subtyping constraints, a subtyping constraint being a pair of types (T, U) written $T \leq^? U$. It has a solution $\varphi : \mathtt{V} \to \mathtt{A} \cup \{\infty\}$ if $P \neq \bot$, $\mathrm{dom}(\varphi) \subseteq \mathrm{Var}(P)$ and, for all $T \leq^? U \in P$, $T\varphi \leq U\varphi$. Let $\mathrm{Sol}(P)$ be the set of all the solutions of P. A solution φ is *more general than* another solution ψ, written $\varphi \sqsubseteq \psi$, if there is θ such that $\varphi\theta \leq^\infty_{\mathtt{A}} \psi$, that is, for all α, $\alpha\varphi\theta \leq^\infty_{\mathtt{A}} \alpha\psi$. Finally, let $\equiv$ be the equivalence relation $\sqsubseteq \cap \sqsupseteq$ associated to $\sqsubseteq$.

Again, one can easily check that the ordering $\sqsubseteq$ on substitutions is a quasi-ordering.

Remark 1 In the case of a unification problem, *i.e.* when $\leq_{\mathtt{A}}^{\infty}$ is the equality, one may more naturally define the ordering on substitutions as the pointwise extension of $\sqsubseteq$: $\varphi \sqsubseteq_{\text{ext}} \psi$ if, for all T, $T\varphi \sqsubseteq T\psi$. Huet proved in [Hue76] that the two definitions are equivalent if the algebra is non-monadic, that is, if it has a term constructor of arity > 1, which is the case for types where $\Rightarrow$ is of arity 2 (the result also holds if all term constructors are of arity 0). Indeed, in a monadic algebra, if you take $\varphi = \{(\alpha, \gamma), (\beta, \gamma)\}$ with $\alpha \neq \beta$, and $\psi = \text{id}$, then $\varphi \sqsubseteq_{\text{ext}} \psi$ but $\varphi \not\sqsubseteq \psi$ (there is no θ such that $\varphi\theta = \psi$). Hence, the above definition is more appropriate than $\sqsubseteq_{\text{ext}}$.

Here, like for unification, we can check that $\sqsubseteq_{\text{ext}} \subseteq \sqsubseteq$. Indeed, assume that $\varphi \sqsubseteq_{\text{ext}} \psi$. Let $V = \text{dom}(\varphi) \cup \text{dom}(\psi) \cup \text{Var}(\varphi)$ where $\text{Var}(\varphi) = \bigcup\{\text{Var}(\alpha\varphi) \mid \alpha \in \text{dom}(\varphi)\}$. Then, let T be a type such that $\text{Var}(T) = V$ and, for all $\alpha \in V$, there is $\mathsf{B} \in \mathbb{S}$ and $p \in \text{Pos}^+(T)$ such that the subtype of T at position p is B_α. To build such a type, we can proceed by induction as follows. Assume that we have to build a type T_n with n variables $\alpha_1, \ldots, \alpha_n$. Then, let $T_1 = \mathsf{B}_{\alpha_1}$ and, for all $k \geq 1$, let $T_{k+1} = (\mathsf{B}_{\alpha_{k+1}} \Rightarrow \mathsf{B}) \Rightarrow T_k$, where B is any sort. Since $\varphi \sqsubseteq_{\text{ext}} \psi$, there is θ such that $T\varphi\theta \leq T\psi$. For all $\alpha \in V$, $\text{Pos}(\alpha, T) \subseteq \text{Pos}^+(T)$. Hence, $\alpha\varphi\theta \leq_{\mathtt{A}}^{\infty} \alpha\psi$. Now, by taking $\theta|_V = \{(\alpha, \alpha\theta) | \alpha \in V\}$, we have $\varphi(\theta|_V) \leq_{\mathtt{A}}^{\infty} \psi$. Therefore, $\varphi \sqsubseteq \psi$.

However, since $\leq_{\mathtt{A}}^{\infty}$ is not symmetric, the converse only hold for the types T such that, for all $\alpha \in \text{dom}(\varphi) \cup \text{dom}(\psi)$, $\text{Pos}(\alpha, T) \subseteq \text{Pos}^+(T)$. ■

To define our algorithm for computing the most general type of a term (Figure 6), we make the following assumptions:

- every solvable subtyping problem P has a most general solution $\text{mgs}(P)$;
- there is an algorithm for deciding whether a subtyping problem is solvable and, if so, computing its most general solution.

We will see in Section 7 that these assumptions are satisfied when types are annotated in the successor algebra. On the other hand, they are not generally satisfied in the Presburger algebra or the max-plus algebra.

In the case of an application $h\vec{u}$, the algorithm proceeds as follows:

1. It checks that h is declared and get back its declared type T.
2. It checks that h can take at least $|\vec{u}|$ arguments, that is, T is of the form $\vec{V} \Rightarrow V$ with $|\vec{V}| = |\vec{u}|$.
3. If h is a function symbol equivalent to f, then it checks that $|\vec{V}| \geq \mathrm{q}^h$.
4. It tries to infer the types of every u_i.
5. If it succeeds and gets U_i for the type of u_i, then it renames the variables of every U_i, using permutations $\rho_1, \ldots, \rho_n$, so that, for all i, $U_i\rho_i$ has no variable in common with the declared type of h and, for all $i \neq j$, $U_i\rho_i$ and $U_j\rho_j$ have no variable in common.

Figure 6: Type inference algorithm

$$\text{(inf-lam)}\quad \frac{\Gamma, x : U \vdash^{\mathsf{f}} v \uparrow V}{\Gamma \vdash^{\mathsf{f}} \lambda x^U v \uparrow U \Rightarrow V}$$

$$\text{(inf-app)}\quad \frac{\begin{array}{c}(h, \vec{V} \Rightarrow V) \in \Gamma \cup \overline{\Theta} \\ h <_{\mathbb{F}} \mathsf{f} \vee (h \simeq_{\mathbb{F}} \mathsf{f} \wedge |\vec{V}| \geq \mathrm{q}^h) \\ (\forall i)\Gamma \vdash^{\mathsf{f}} u_i \uparrow U_i \\ \rho_1, \ldots, \rho_n \text{ permutations on } \mathtt{V}\ (n = |\vec{V}|) \\ (\forall i)\mathrm{Var}(U_i\rho_i) \cap \mathrm{Var}(\vec{V} \Rightarrow V) = \emptyset \\ (\forall i)(\forall j) i \neq j \Rightarrow \mathrm{Var}(U_i\rho_i) \cap \mathrm{Var}(U_j\rho_j) = \emptyset \\ \eta = \mathrm{mgs}(\{U_1\rho_1 \leq^? V_1, \ldots, U_n\rho_n \leq^? V_n\})\end{array}}{\Gamma \vdash^{\mathsf{f}} h\vec{u} \uparrow V\eta}$$

6. Finally, it tries to compute the most general solution η of the problem $\{U_1\rho_1 \leq^? V_1, \ldots, U_n\rho_n \leq^? V_n\}$ and returns $V\eta$ in case of success.

We first check that our algorithm computes a valid type. To this end, first note that:

Lemma 12 If $\Gamma \vdash^{\mathsf{f}} t : T$ then, for every size substitution θ, $\Gamma\theta \vdash^{\mathsf{f}} t\theta : T\theta$.

Proof. Straightforward induction using the fact that $\leq_{\mathtt{A}}$ and thus $\leq_{\mathtt{A}}^{\infty}$ and $\leq$ are stable by substitution. ■

In the following, we assume that Γ and t contain no size variables. This is not a restriction wrt. the termination conditions since:

- t is then a subterm of a rule right-hand side and rules are made of terms in $\mathbb{T}$ and thus contain no annotated types;
- On the other hand, Γ contains the variables α^x. The condition $\mathrm{Var}(\Gamma) = \emptyset$ can however be enforced wlog. by replacing every size variable α^x by a new constant $\mathtt{c}^x$ so that $\mathtt{c}^x = \mathtt{c}^y$ whenever $\alpha^x = \alpha^y$.

Theorem 3 (Correctness wrt. $\vdash^{\mathsf{f}}$) If $\Gamma \vdash^{\mathsf{f}} t \uparrow T$ and $\mathrm{Var}(\Gamma) = \mathrm{Var}(t) = \emptyset$, then $\Gamma \vdash^{\mathsf{f}} t : T$.

Proof. By induction on $\Gamma \vdash^{\mathsf{f}} t \uparrow T$. We only detail the case (inf-app). By induction hypothesis, $\Gamma \vdash^{\mathsf{f}} u_i : U_i$. By Lemma 12, $\Gamma \vdash^{\mathsf{f}} u_i : U_i\rho_i\eta$ since $\mathrm{Var}(\Gamma) = \mathrm{Var}(u_i) = \emptyset$. Since $U_i\rho_i\eta \leq V_i\eta$, by (sub), $\Gamma \vdash^{\mathsf{f}} u_i : V_i\eta$. Therefore,

by (app), $\Gamma \vdash^{\mathsf{f}} h\vec{u} : V\eta$. ■

We now prove that our algorithm computes the most general type of t in Γ and thus is complete:

Theorem 4 (Completeness wrt. $\vdash^{\mathsf{f}}$) If $\Gamma \vdash^{\mathsf{f}} t : T$, $\mathrm{Var}(\Gamma) = \mathrm{Var}(t) = \emptyset$ and every function symbol occurring in t satisfies the monotony condition of Theorem 2, then there is U such that $\Gamma \vdash^{\mathsf{f}} t \uparrow U$ and $U \sqsubseteq T$.

Proof. We proceed by induction on $\Gamma \vdash^{\mathsf{f}} t : T$. We only detail the case (app) when $h \in \mathbb{C} \cup \mathbb{F}$. By induction hypothesis, $\Gamma \vdash^{\mathsf{f}} u_i \uparrow U_i$ and there is θ_i such that $U_i\theta_i \leq V_i\psi$. Wlog. we can assume that $\mathrm{dom}(\theta_i) \subseteq \mathrm{Var}(U_i)$. Let now $\rho_1, \ldots, \rho_n$ satisfying the conditions of (inf-app), and $\xi = \{(\alpha, \alpha\psi) \mid \alpha \in \mathrm{Var}(\vec{V} \Rightarrow V)\} \cup \{(\alpha, \alpha\rho_i^{-1}\theta_i) \mid \alpha \in \mathrm{Var}(U_i\rho_i), 1 \leq i \leq n\}$. Then, for all i, $U_i\rho_i\xi = U_i\theta_i \leq V_i\psi = V_i\xi$. Therefore, $P = \{U_1\rho_1 \leq^? V_1, \ldots, U_n\rho_n \leq^? V_n\}$ is solvable, $\eta = \mathrm{mgs}(P)$ exists and there is θ such that $\eta\theta \leq^{\infty}_{\mathtt{A}} \xi$. Hence, by (inf-app), $\Gamma \vdash^{\mathsf{f}} h\vec{u} \uparrow V\eta$. After the monotony condition, variables occur only positively in V. Therefore, $V\eta\theta \leq V\xi = V\psi$. Hence, $V\eta \sqsubseteq V\psi$. ■

Next, we prove that, if the size algebra is monotone, then it is enough to check the decreasingness condition on some appropriate instance of the derivation of the most general type. To this end, we need to make the notion of derivation more precise:

Definition 19 (Derivation) Derivations of $\Gamma \vdash^{\mathsf{f}} t : T$ are defined as follows:

- If $(h, \vec{V} \Rightarrow V) \in \Gamma \cup \overline{\Theta}$ and, for all i, π_i is a derivation of $\Gamma \vdash^{\mathsf{f}} u_i : V_i\psi$, written $\pi_i \rhd \Gamma \vdash^{\mathsf{f}} u_i : V_i\psi$, then $\mathrm{a}(\Gamma, h\vec{u}, \psi; \vec{\pi})$ is the derivation of $\Gamma \vdash^{\mathsf{f}} h\vec{u} : V\psi$ whose last rule is (app).
- If $\pi \rhd \Gamma, x : U \vdash^{\mathsf{f}} v : V$, then $\mathrm{l}(\pi)$ is the derivation of $\Gamma \vdash^{\mathsf{f}} \lambda x^U v : U \Rightarrow V$ whose last rule is (lam).
- If $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : U$ and $U \leq V$, then $\mathrm{s}(\pi, V)$ is the derivation of $\Gamma \vdash^{\mathsf{f}} t : V$ whose last rule is (sub).

Similarly, derivations of $\Gamma \vdash^{\mathsf{f}} t \uparrow T$ are defined as follows:

- If $(h, \vec{V} \Rightarrow V) \in \Gamma \cup \overline{\Theta}$, $\vec{\rho}$ are permutations satisfying the conditions of (inf-app) and, for all i, $\pi_i \rhd \Gamma \vdash^{\mathsf{f}} u_i \uparrow U_i$, then $\mathrm{i}(\Gamma, h\vec{u}, \vec{\rho}; \vec{\pi})$ is the derivation of $\Gamma \vdash^{\mathsf{f}} h\vec{u} \uparrow V\eta$ whose last rule is (inf-app).
- If $\pi \rhd \Gamma, x : U \vdash^{\mathsf{f}} v \uparrow V$, then $\mathrm{l}(\pi)$ is the derivation of $\Gamma \vdash^{\mathsf{f}} \lambda x^U v \uparrow U \Rightarrow V$ whose last rule is (inf-lam).

Given a derivation $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$ and a substitution θ, let $\pi\theta$ be the derivation of $\Gamma\theta \vdash^{\mathsf{f}} t\theta : T\theta$ obtained by applying θ everywhere in π.

After the proof of Theorem 4, a derivation of $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$ is transformed into a derivation $\pi\uparrow \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow U$ for some type U such that $\mathrm{a}(\Gamma, h\vec{u}, \psi; \vec{\pi})\uparrow = \mathrm{i}(\Gamma, h\vec{u}, \vec{\rho}; \vec{\pi}\uparrow)$ where $\vec{\rho}$ are some permutations satisfying the conditions of rule (inf-app). Moreover, there is θ such that $U\theta \leq^{\infty}_{\mathtt{A}} T$.

After the proof of Theorem 3, a derivation $\pi \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow T$ is transformed into a derivation $|\pi| \rhd \Gamma \vdash^{\mathsf{f}} t : T$ such that $|\mathrm{i}(\Gamma, h\vec{u}, \vec{\rho}; \vec{\pi})| = \mathrm{a}(\Gamma, h\vec{u}, \eta; \vec{\upsilon})$ where $\upsilon_i = \mathrm{s}(|\pi_i|\rho_i\eta, V_i\eta)$.

We can now prove that the most general type is complete wrt. the decreasingness condition:

Lemma 13 In a monotone algebra, if $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$, $\pi\xi' \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T\xi'$ and $\xi \leq^{\infty}_{\mathtt{A}} \xi'$, then $\pi\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T\xi$.

Proof. By induction on $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$. We only detail the case (app) when $h \simeq_{\mathbb{F}} \mathsf{f}$. We have $(h, \vec{V} \Rightarrow V) \in \Theta$, $\pi_i \rhd \Gamma \vdash^{\mathsf{f}} u_i : V_i\psi$, $\pi_i\xi' \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi\xi'$ and $\vec{\alpha}^h\psi\xi' <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$. By induction hypothesis, $\pi_i\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi\xi$. Since $\xi \leq^{\infty}_{\mathtt{A}} \xi'$ and the size algebra is monotone, we have $\psi\xi \leq^{\infty}_{\mathtt{A}} \psi\xi'$. Hence, $\vec{\alpha}^h\psi\xi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$ by assumption on $<^{h,\mathsf{f}}_{\mathtt{A}}$ (see Definition 16). Therefore, $\pi\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} h\vec{u} : V\psi\xi$. ■

Theorem 5 (Completeness wrt. $\vdash^{\mathsf{f}}_{\varphi}$) In a monotone algebra, if $\pi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$, $\mathrm{Var}(\Gamma) = \mathrm{Var}(t) = \emptyset$, $\pi\uparrow \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow U$ and $U\theta \leq T$, then $\mathrm{s}(|\pi\uparrow|\theta, T) \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$.

Proof. We proceed by induction on $\pi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$. We only detail the case (app-decr) when $t = h\vec{u}$, $(h, \vec{V} \Rightarrow V) \in \overline{\Theta}$, $T = V\psi$, $\vec{\alpha}^h\psi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$ and $U = V\eta$ where η is given by the rule (inf-app). We have $\pi = \mathrm{a}(\Gamma, h\vec{u}, \psi; \vec{\pi})$, $\pi\uparrow = \mathrm{i}(\Gamma, h\vec{u}, \vec{\rho}; \vec{\pi}\uparrow)$, $|\pi\uparrow| = \mathrm{a}(\Gamma, h\vec{u}, \eta; \vec{\upsilon})$ where $\upsilon_i = \mathrm{s}(|\pi_i\uparrow|\rho_i\eta, V_i\eta)$, $|\pi\uparrow|\theta = \mathrm{a}(\Gamma, h\vec{u}, \eta\theta; \vec{\upsilon}\theta)$ and, for all i, $\pi_i \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi$, $\pi_i\uparrow \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow U_i$ and $U_i\theta_i \leq V_i\psi$ for some θ_i. By induction hypothesis, $\mathrm{s}(|\pi_i\uparrow|\theta_i, V_i\psi) \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : V_i\psi$. In particular, $|\pi_i\uparrow|\theta_i \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : U_i\theta_i$, that is, $|\pi_i\uparrow|\rho_i\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : U_i\rho_i\xi$, where ξ is defined in the proof of Theorem 4. Since $\eta\theta \leq^{\infty}_{\mathtt{A}} \xi$, by Lemma 13, we get $|\pi_i\uparrow|\rho_i\eta\theta \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : U_i\rho_i\eta\theta$. Hence, $\upsilon_i\theta \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} u_i : V_i\eta\theta$. Moreover, since $\vec{\alpha}^h\eta\theta \leq^{\infty}_{\mathtt{A}} \vec{\alpha}^h\xi = \vec{\alpha}^h\psi$ and $\vec{\alpha}^h\psi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$, we get $\vec{\alpha}^h\eta\theta <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$ by assumption on $<^{h,\mathsf{f}}_{\mathtt{A}}$. Therefore, $\mathrm{s}(|\pi\uparrow|\theta, T) \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$. ■

We therefore get, for monotone algebras, the following complete procedure for deciding $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$:

1. Let γ be a size substitution mapping every α^x to a new constant c^x so that $\mathsf{c}^x = \mathsf{c}^y$ whenever $\alpha^x = \alpha^y$.
2. Check whether there is U such that $\Gamma\gamma \vdash^{\mathsf{f}} t \uparrow U$. If it fails, then t is not typable in Γ.
3. If it succeeds, then try to solve the problem $\{U \leq^{?} T\gamma\}$. If it fails, then $\Gamma \vdash^{\mathsf{f}} t : T$ does not hold.

4. If it succeeds, then let θ be the smallest solution of $\{U \leq^? T\gamma\}$. Then, try to check whether $|\pi|\theta \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : U\theta$, where π is the (unique) derivation of $\Gamma \vdash^{\mathsf{f}} t \uparrow U$. If it succeeds, then $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$. Otherwise, $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ does not hold.

6 Reducing subtyping problems to size problems

In this section, we show how a subtyping problem can be reduced to solving constraints in $\mathtt{A} \cup \{\infty\}$. As subtyping is not syntax-directed, we first prove that it is equivalent to a syntax-directed relation. To this end, we prove that the subtyping rules (refl) and (trans) are redundant/can be eliminated, following a proof technique used by Curien and Ghelli in [CG92]:

Theorem 6 $T \leq U$ iff $T \leq_{\mathrm{a}} U$, where $\leq_{\mathrm{a}}$ is inductively defined by the rules (size) and (prod) only.

Proof. Let $\leq_1$ be the relation inductively defined by the same rules as $\leq$ except for (size) replaced by:

$$\text{(size')}\ \frac{\mathsf{B}_b \leq_1 T \quad a \leq^{\infty}_{\mathtt{A}} b}{\mathsf{B}_a \leq_1 T}$$

We can easily check that $\leq$ and $\leq_1$ are in fact the same relations:

- $\leq \subseteq \leq_1$. (size) is admissible in $\leq_1$. Indeed, $\mathsf{B}_b \leq \mathsf{B}_b$ by (refl) and $\mathsf{B}_a \leq \mathsf{B}_b$ by (size').
- $\leq_1 \subseteq \leq$. We proceed by induction. In the case of (size'), we have $\mathsf{B}_b \leq T$ by induction hypothesis. By (size), $\mathsf{B}_a \leq \mathsf{B}_b$. Therefore, by (trans), $\mathsf{B}_a \leq T$.

Let $\leq_2$ be the relation inductively defined by the rules (refl), (size') and (prod). Clearly, $\leq_2 \subseteq \leq_1$. We now prove that any deduction tree for $U \leq_1 V$ can be transformed into a deduction tree for $U \leq_2 V$. To this end, we define a transformation Ξ on deduction trees for $\leq_1$ as follows.

Let π be a deduction tree of $T \leq_1 V$ ending by an application of (trans) from a deduction tree π_1 of $T \leq_1 U$ and a deduction tree π_2 of $U \leq_1 V$:

$$\frac{\dfrac{\pi_1}{T \leq_1 U} \qquad \dfrac{\pi_2}{U \leq_1 V}}{T \leq_1 V}\ \text{(trans)}$$

- If π_1 ends with (refl), then $T = U$ and Ξ replaces π by π_2:

$$\frac{\dfrac{}{T \leq_1 T}\ \text{(refl)} \qquad \dfrac{\pi_2}{T \leq_1 V}}{T \leq_1 V}\ \text{(trans)} \qquad \rightarrow_{\Xi} \qquad \frac{\pi_2}{T \leq_1 V}$$

- Symmetrically, if π_2 ends with (refl), then $U = V$ and T replaces π by π_1:

$$\dfrac{\dfrac{\pi_1}{T \leq_1 U} \qquad \dfrac{}{U \leq_1 U}\,(\text{refl})}{T \leq_1 U}\,(\text{trans}) \qquad \to_\Xi \qquad \dfrac{\pi_1}{T \leq_1 U}$$

- If π_1 ends with (size'), then there are a, b and π_0 such that π_0 is a deduction tree of $\mathsf{B}_a \leq U$, $a \leq^\infty_{\mathtt{A}} b$ and $T = \mathsf{B}_b$. Then, Ξ acts as follows:

$$\dfrac{\dfrac{\dfrac{\pi_0}{\mathsf{B}_b \leq_1 U} \qquad a \leq^\infty_{\mathtt{A}} b}{\mathsf{B}_a \leq_1 U}\,(\text{size'}) \qquad \dfrac{\pi_2}{U \leq_1 V}}{\mathsf{B}_a \leq_1 V}\,(\text{trans})$$

$$\to_\Xi \quad \dfrac{\dfrac{\dfrac{\pi_0}{\mathsf{B}_b \leq_1 U} \qquad \dfrac{\pi_2}{U \leq_1 V}}{\mathsf{B}_b \leq_1 V}\,(\text{trans}) \qquad a \leq_{\mathrm{a}} b}{\mathsf{B}_a \leq_1 V}\,(\text{size'})$$

- If both π_1 and π_2 ends with (prod), then Ξ acts as follows:

$$\dfrac{\dfrac{\dfrac{\pi_{11}}{A' \leq_1 A} \qquad \dfrac{\pi_{12}}{B \leq_1 B'}}{A \Rightarrow B \leq_1 A' \Rightarrow B'}\,(\text{prod}) \qquad \dfrac{\dfrac{\pi_{21}}{A'' \leq_1 A'} \qquad \dfrac{\pi_{22}}{B' \leq_1 B''}}{A' \Rightarrow B' \leq_1 A'' \Rightarrow B''}\,(\text{prod})}{A \Rightarrow B \leq_1 A'' \Rightarrow B''}\,(\text{trans})$$

$$\to_\Xi \quad \dfrac{\dfrac{\dfrac{\pi_{21}}{A'' \leq_1 A'} \qquad \dfrac{\pi_{11}}{A' \leq_1 A}}{A'' \leq_1 A}\,(\text{trans}) \qquad \dfrac{\dfrac{\pi_{12}}{B \leq_1 B'} \qquad \dfrac{\pi_{22}}{B' \leq_1 B''}}{B \leq_1 B''}\,(\text{trans})}{A \Rightarrow B \leq_1 A'' \Rightarrow B''}\,(\text{prod})$$

The transformation Ξ can be more synthetically represented by a rewrite system on first-order closed terms representing deduction trees for $\leq_1$. Indeed, if π represents a deduction tree for $\mathsf{B}_b \leq_1 T$ and $a \leq^\infty_{\mathtt{A}} b$, then let $\mathsf{s}(\pi)$ represent the deduction tree of $\mathsf{B}_a \leq_1 T$ made of π followed by an application of (size'). Similarly, we can use r for (refl), t for (trans), and p for (prod). Hence, Ξ consists in applying the following rewrite rules on the top of a term representing a deduction tree:

$$\begin{aligned}
\mathsf{t}\ \mathsf{r}\ \pi_2 &\to \pi_2 \\
\mathsf{t}\ \pi_1\ \mathsf{r} &\to \pi_1 \\
\mathsf{t}\ (\mathsf{s}\ \pi_0)\ \pi_2 &\to \mathsf{s}\ (\mathsf{t}\ \pi_0\ \pi_2) \\
\mathsf{t}\ (\mathsf{p}\ \pi_{11}\ \pi_{12})\ (\mathsf{p}\ \pi_{21}\ \pi_{22}) &\to \mathsf{p}\ (\mathsf{t}\ \pi_{21}\ \pi_{11})\ (\mathsf{t}\ \pi_{12}\ \pi_{22})
\end{aligned}$$

This rewrite system clearly terminates. Moreover, one can easily check that it is locally confluent (every critical pair is joinable) [KB70]. Therefore, it is confluent [New42] and every term t has a unique normal form. Now, we can easily check that every normal closed term t contains no $\mathtt{t}$, by induction on t. Indeed, assume that $t = \mathtt{t}\ \pi_1\ \pi_2$ is a normal closed term. Since t is closed, π_1 and π_2 must be headed by a symbol. By induction hypothesis, this cannot be $\mathtt{t}$. Since t is normal, π_1 cannot be headed by $\mathtt{r}$ or $\mathtt{s}$. So, π_1 must be headed by $\mathtt{p}$. Then, π_2 cannot be headed by $\mathtt{r}$ nor $\mathtt{p}$. Therefore, π_2 must be headed by $\mathtt{s}$. But these case is impossible since (size') applies to sorts only.

So, $\leq_1$ and $\leq_2$ are equal and we are left to prove that $\leq_2$ is equal to $\leq_{\mathrm{a}}$.

First note that $\leq_{\mathrm{a}} \subseteq \leq_2$ since, as seen at the beginning, (size) can be replaced by (refl) followed by (size') respectively.

Now, $\leq_2 \subseteq \leq_{\mathrm{a}}$ since a deduction tree for $\leq_2$ not ending with an application of (prod) is necessarily of the form $\mathtt{s}^n\mathtt{r}$, and thus can be replaced by a single application of (size). ■

As a consequence, we can prove that a subtyping problem can be reduced to an equivalent size problem as follows:

Definition 20 (Size problem) A *size problem* is either $\bot$ or a finite set of size constraints, a size constraint being a pair of extended size expressions (a, b), written $a \leq^? b$. It has a solution $\varphi : \mathtt{V} \to \mathtt{A} \cup \{\infty\}$ if $P \neq \bot$, $\mathrm{dom}(\varphi) \subseteq \mathrm{Var}(P)$ and, for all $a \leq^? b \in P$, $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$. Let $\mathrm{Sol}(P)$ ($\mathrm{Sol}_{\mathtt{A}}(P)$ resp.) be the set of all the (*algebraic* resp.) solutions $\varphi : \mathtt{V} \to \mathtt{A} \cup \{\infty\}$ ($\varphi : \mathtt{V} \to \mathtt{A}$ resp.) of P.

We define the size problem associated to a subtyping problem as follows:

- $|\emptyset| = \emptyset$,
- $|P \cup Q| = |P| \cup |Q|$ if $|P| \neq \bot$ and $|Q| \neq \bot$,
- $|\{\mathsf{B}_a \leq^? \mathsf{B}_b\}| = \{a \leq^? b\}$,
- $|\{U \Rightarrow V \leq^? U' \Rightarrow V'\}| = |\{U' \leq^? U, V \leq^? V'\}|$,
- $|P| = \bot$ otherwise.

Lemma 14 $\mathrm{Sol}(P) = \mathrm{Sol}(|P|)$.

Proof. We proceed by induction on P. We only detail the case where $P = \{T \leq^? T'\}$:

- Let $\varphi \in \mathrm{Sol}(P)$. Then, $T\varphi \leq_{\mathrm{a}} T'\varphi$. If $T = \mathsf{B}_a$, then $T' = \mathsf{B}_b$ and $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$. Otherwise, $T = U \Rightarrow V$, $T' = U' \Rightarrow V'$, $U'\varphi \leq_{\mathrm{a}} U\varphi$ and $V\varphi \leq_{\mathrm{a}} V'\varphi$. By induction hypothesis, $\varphi \in \mathrm{Sol}(|U' \leq^? U|) \cap \mathrm{Sol}(|V \leq^? V'|)$. So, in both cases, $\varphi \in \mathrm{Sol}(|P|)$.
- Let $\varphi \in \mathrm{Sol}(|P|)$. If $T = \mathsf{B}_a$, then $T' = \mathsf{B}_b$ and $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$. Otherwise, $T = U \Rightarrow V$, $T' = U' \Rightarrow V'$, $\varphi \in \mathrm{Sol}(|U' \leq^? U|) \cap \mathrm{Sol}(|V \leq^? V'|)$. By induction hypothesis, $U'\varphi \leq_{\mathrm{a}} U\varphi$ and $V\varphi \leq_{\mathrm{a}} V'\varphi$. So, in both cases, $\varphi \in \mathrm{Sol}(P)$. ■

To go further, we need to make more assumptions on the size algebra.

Figure 7: Ordering in the successor algebra

$$\frac{}{a \leq_{\mathtt{A}} a} \quad \frac{a <_{\mathtt{A}} b}{a \leq_{\mathtt{A}} b} \quad \frac{}{a <_{\mathtt{A}} \mathtt{s}\, a} \quad \frac{a <_{\mathtt{A}} b \quad b <_{\mathtt{A}} c}{a <_{\mathtt{A}} c}$$

7 Solving size problems in the successor algebra

In this section, we prove that, in the successor algebra, the solvability of a size problem is decidable in polynomial time, and solvable size problems have a most general solution that can be computed in polynomial time too. Although this algebra may seem overly simple, it is already enough to subsume the termination criterion that is for instance used in the Coq proof assistant, as we have seen in Section 4.3.

Definition 21 (Successor algebra) The *successor* size algebra is given by the first-order term algebra $\mathtt{A}$ built from an arbitrary (empty, finite or infinite) set $\mathtt{F}_0$ of nullary symbols, the unary function symbol $\mathtt{s}$ interpreted in $\mathfrak{h}$ as the successor function and, for $\leq_{\mathtt{A}}$, the reflexive closure of the smallest strict ordering $<_{\mathtt{A}}$ such that, for all a, $a <_{\mathtt{A}} \mathtt{s}\, a$, which can also be defined by the rules of Figure 7. Given $k \in \mathbb{N}$, let $\mathtt{s}^k$ denote the k-th iterate of $\mathtt{s}$.

Note that, in this definition, constants are incomparable. In particular, we do not assume that $\mathtt{A}$ has a smallest element, that is, a constant $\mathtt{0}$, interpreted by the ordinal 0 and such that that, for all a, $\mathtt{0} \leq_{\mathtt{A}} a$.

Lemma 15

- If $a <_{\mathtt{A}} b$, then there is b' such that $b = \mathtt{s}b'$ and $a \leq_{\mathtt{A}} b'$.
- If $\mathtt{s}a <_{\mathtt{A}} \mathtt{s}b$, then $a <_{\mathtt{A}} b$.
- Conversely, if $a <_{\mathtt{A}} b$, then $\mathtt{s}a <_{\mathtt{A}} \mathtt{s}b$, that is, $\mathtt{A}$ is a monotone algebra.
- $<_{\mathtt{A}}$ is a strict ordering.
- $\leq_{\mathtt{A}}$ is an ordering.

Proof. First note that $\leq_{\mathtt{A}}$ is the reflexive closure of $<_{\mathtt{A}}$ and $<_{\mathtt{A}}$ is transitive. Hence, $a \leq_{\mathtt{A}} b$ whenever $a <_{\mathtt{A}} b$, and $a <_{\mathtt{A}} b$ whenever $a \leq_{\mathtt{A}} c$ and $c <_{\mathtt{A}} b$, or $a <_{\mathtt{A}} c$ and $c \leq_{\mathtt{A}} b$, and $a \leq_{\mathtt{A}} b$ whenever $a \leq_{\mathtt{A}} c$ and $c \leq_{\mathtt{A}} b$.

- By induction on the derivation height of $a <_{\mathtt{A}} b$. If $b = \mathtt{s}a$, then this is immediate. Otherwise, there is c such that $a <_{\mathtt{A}} c$ and $c <_{\mathtt{A}} b$. By induction hypothesis, there is b' such that $b = \mathtt{s}b'$ and $c \leq_{\mathtt{A}} b'$. Therefore, $a \leq_{\mathtt{A}} b'$.
- By induction on the derivation height of $\mathtt{s}a <_{\mathtt{A}} \mathtt{s}b$. If $b = \mathtt{s}a$, then this is immediate. Otherwise, there is c such that $\mathtt{s}a <_{\mathtt{A}} c$ and $c <_{\mathtt{A}} \mathtt{s}b$. Hence, there is c' such that $c = \mathtt{s}c'$ and $\mathtt{s}a \leq_{\mathtt{A}} c'$. By induction hypothesis, $c' <_{\mathtt{A}} b$. Therefore, $a <_{\mathtt{A}} \mathtt{s}a \leq_{\mathtt{A}} c' <_{\mathtt{A}} b$.

- By induction on the derivation height of $a <_{\mathtt{A}} b$. If $b = \mathtt{s}a$, then this is immediate. Otherwise, there is c such that $a <_{\mathtt{A}} c$ and $c <_{\mathtt{A}} b$. By induction hypothesis, $\mathtt{s}a <_{\mathtt{A}} \mathtt{s}c$ and $\mathtt{s}c <_{\mathtt{A}} \mathtt{s}b$. Therefore, by transitivity, $\mathtt{a}a <_{\mathtt{A}} \mathtt{s}b$.
- We prove that $a \not<_{\mathtt{A}} a$ by induction on a. Assume that $a <_{\mathtt{A}} a$. After the first property, $a = \mathtt{s}b$ for some b. After the second property, $b <_{\mathtt{A}} b$. By induction hypothesis, $b \not<_{\mathtt{A}} b$. Therefore, $a \not<_{\mathtt{A}} a$ and $<_{\mathtt{A}}$ is a strict ordering.
- Assume that $a \leq_{\mathtt{A}} b \leq_{\mathtt{A}} a$ and $a \neq b$. Then, $a <_{\mathtt{A}} b <_{\mathtt{A}} a$. By transitivity, $a <_{\mathtt{A}} a$, which is not possible by the previous property. Therefore, $a = b$ and $\leq_{\mathtt{A}}$ is an ordering. ■

Note that these properties do not depend on the number of function symbols of $\mathtt{A}$. Hence, they will be preserved if we add new symbols without changing the definition of $<_{\mathtt{A}}$.

However, for the moment, since there is only one non-nullary symbol and this symbol is unary, every term of $\mathtt{A}$ is of the form $\mathtt{s}^k a$ with a being either a variable α or a constant $\mathtt{c} \in \mathtt{F}_0$.

7.1 Satisfiability

If there were no nullary symbol, a problem in $\mathtt{A} \cup \{\infty\}$ would be equivalent to a problem in the idempotent semi-ring $(\mathbb{Z} \cup \{\pm\infty\}, \max, +)$ for which there are well known resolution and optimization techniques [ABG14].

To deal with constants, we put a size problem into a canonical form. Solvability of this canonical form can be straightforwardly reduced to the satisfiability of a conjunction of integer inequalities between variables and integers, the decidability of which is in turn equivalent to proving that there is no cycle of strictly positive weight in a directed labeled graph built from the set of the inequalities [Pra77]:

Definition 22 (Integer problem) Let an *integer problem* be (here) a set of constraints of the form $a + k \leq^? b$ with $k \in \mathbb{Z}$ and $a, b \in \{0\} \uplus \mathtt{V}$. A solution of an integer problem P is a function $\psi : \mathrm{Var}(P) \to \mathbb{N}$ such that, for every constraint $a + k \leq^? b \in P$, one has $\psi(a) + k \leq \psi(b)$, where $\psi(0) = 0$. Let $\mathrm{Sol}(P)$ be the set of all the solutions of P.

An integer problem P can be represented by a directed graph $G(P)$ on $\{0\} \uplus \mathrm{Var}(P)$ such that there is a labeled edge $a \xrightarrow{k} b$ in $G(P)$ iff $a + k \leq^? b$ is a constraint in P. Conversely, a finite directed graph G on $\{0\} \uplus \mathtt{V}$ with its edges labeled by integers corresponds to the integer problem $P(G)$ such that $a + k \leq^? b \in P(G)$ iff $a \xrightarrow{k} b \in G$.

The *weight* of a path $a_1 \xrightarrow{k_1} \ldots \xrightarrow{k_n} a_{n+1}$ is $\Sigma_{i=1}^{n} k_i$. A *cycle* (*i.e.* when $a_{n+1} = a_1$) is *increasing* if its weight is > 0. A problem P is increasing if $G(P)$ has an increasing cycle.

Following Pratt [Pra77], an integer problem P is satisfiable iff P is not increasing. That a problem is increasing can be decided in polynomial time

"e.g., by forming the max/+ transitive closure of the graph and searching for a self-edge with a positive label".

Now, our "solved" form for size problems will be obtained by normalizing the problem wrt. a terminating set of rules of different kinds:

1. Since $\mathtt{s}a \leq \mathtt{s}b$ iff $a \leq b$, a constraint of the form $\mathtt{s}a \leq^? \mathtt{s}b$ can always be replaced by the simpler constraint $a \leq^? b$ without changing the set of solutions.

2. Some constraints are true whatever the instantiation of their variables is (*e.g.* $a \leq^? \infty$) and can thus be removed without changing the set of solutions. We will however keep some constraints of this form for preserving the set of variables of the problem.

3. Conversely, some constraints are always false (*e.g.* $\infty \leq^? \mathtt{c}$). In this case, the problem can be replaced by $\bot$ without changing the set of solutions (which is empty).

4. If there is a constraint of the form $\infty \leq^? \alpha$ then, for any solution φ, $\alpha\varphi = \infty$. This means that α can be replaced by ∞ in all the other constraints without changing the set of solutions. But this may also happen for more complex reasons. Take for instance the problem $P = \{\mathtt{s}\alpha \leq^? \beta, \beta \leq \alpha\}$. There is no ∞ symbol occurring in it but one can easily see that both α and β must be set to ∞ for P to be satisfiable. For detecting these situations, we can again use Pratt's graph:

Definition 23 (Linear problem) A constraint is *linear* if it is of the form $\mathtt{s}^k\alpha \leq^? \beta$ or $\alpha \leq^? \mathtt{s}^k\beta$ with $\alpha \neq \beta$. A problem is *linear* if it only contains linear constraints. Given a linear problem P and an injection $\mathrm{x} : \mathtt{V} \to \mathtt{V}$ (for technical reasons explained after Definition 25), let its associated integer problem be $I(P) = \{k + \mathrm{x}_\alpha \leq^? l + \mathrm{x}_\beta \mid \mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in P\}$, where $\mathrm{x}_\alpha = \mathrm{x}(\alpha)$. A linear problem P is non-increasing if $I(P)$ is non-increasing.

We will see that, given a linear problem P, if $G(I(P))$ is an increasing cycle, then every variable of P must be set to ∞.

5. Finally, if there is a constraint of the form $\alpha \leq^? \mathtt{s}^l\mathtt{c}$ then, for any solution φ, $\alpha\varphi$ is of the form $\mathtt{s}^k\mathtt{c}$ with $k \leq l$. This means that α can be replaced by $\mathtt{s}^k\mathtt{c}$ in all the other constraints without changing the set of solutions. However, we cannot know in advance what should be the value of k. To get around this issue, we consider a richer size algebra in which $\mathtt{s}^X\mathtt{c}$, where X is a variable, is a valid size expression; replace α by $\mathtt{s}^X\mathtt{c}$ in all the other constraints; and add a constraint of a new kind, namely $X \leq l$.

Definition 24 (Successor-iterator algebra) Let $\mathtt{B}$ be the following multi-sorted algebra:

- sorts: $\mathtt{O}$ interpreted by $\mathfrak{h}$, and $\mathtt{N}$ interpreted by ω;

- variables: every variable is given a sort;
- symbols: $\mathtt{0} : \mathtt{N}$ interpreted by zero, $\mathtt{s} : \mathtt{N} \to \mathtt{N}$ and $\mathtt{s} : \mathtt{O} \to \mathtt{O}$ (we use overloading) both interpreted by the successor function, $\mathtt{c} : \mathtt{O}$ for every $\mathtt{c} \in \mathtt{F}_0$, $\mathtt{s} : \mathtt{N} \times \mathtt{O} \to \mathtt{O}$, with $\mathtt{s}(a, b)$ written $\mathtt{s}^a b$, interpreted as the iteration of the successor function, that is, $(\mathtt{s}^a b)\mu = b\mu + a\mu$.
- quasi-ordering: let $\leq_{\mathtt{B}} = <_{\mathtt{A}} \cup \simeq_{\mathtt{B}}$ where $\simeq_{\mathtt{B}}$ is the smallest congruence defined by the following semantically valid equations:

$$\begin{array}{rcl} \mathtt{s}^{\mathtt{0}} y & \simeq_{\mathtt{B}} & y \\ \mathtt{s}^{\mathtt{s}^x} y & \simeq_{\mathtt{B}} & \mathtt{s}(\mathtt{s}^x y) \\ \mathtt{s}^x(\mathtt{s}y) & \simeq_{\mathtt{B}} & \mathtt{s}(\mathtt{s}^x y) \end{array}$$

Let $\mathrm{Var}_s(a)$ be the variables of sort s occurring in a.

In a multi-sorted algebra, substitutions map a variable of sort s to a term of sort s, and constraints are pairs of terms of the same sort.

Closed terms of sort $\mathtt{N}$ are isomorphic to natural numbers. Hence, in the following, we identity $\mathtt{s}^k\mathtt{0}$ where $k \in \mathbb{N}$ with k itself, use the letters k and l (e and f resp.) to denote closed (arbitrary resp.) expressions of sort $\mathtt{N}$, and denote $\mathtt{s}^k\alpha$ by $k + \alpha$.

One can easily check that, when oriented from left to right, the equations defining $\simeq_{\mathtt{B}}$ form a confluent and terminating rewrite system. Hence, every term has a normal form and two equivalent terms have the same normal form. So, in the following, we will always assume that terms are in normal form, and use $\leq_{\mathtt{A}}$ instead of $\leq_{\mathtt{B}}$. Moreover, any normalized term is of the form $\mathtt{s}^k\mathtt{s}^{\alpha_1} \ldots \mathtt{s}^{\alpha_n} a$ with $a \in \mathtt{V} \cup \mathtt{F}_0$. However, since we start from terms in $\mathtt{A}$ and only substitute terms of the form $\mathtt{s}^\alpha\mathtt{c}$, we will only have terms of the form $\mathtt{s}^k\alpha$ or $\mathtt{s}^e\mathtt{c}$ with $e = k$ or $e = k + \alpha$.

As explained above, to decide whether a size problem P in $\mathtt{A} \cup \{\infty\}$ is solvable, we are going to transform it into an equivalent size problem Q in $\mathtt{B} \cup \{\infty\}$, assuming that all the variables of P are of sort $\mathtt{O}$. But for the two problems to have the same solutions, we need to restrict the notion of solution for problems in $\mathtt{B} \cup \{\infty\}$ as follows:

Definition 25 A size problem P in $\mathtt{B} \cup \{\infty\}$ has a solution $\varphi : \mathtt{V} \to \mathtt{A} \cup \{\infty\}$ if $P \neq \bot$, $\mathrm{dom}(\varphi) \subseteq \mathrm{Var}(P)$, φ maps every variable of sort $\mathtt{N}$ to a *closed* term of sort $\mathtt{N}$ and, for every constraint $a \leq^? b \in P$, $a\varphi \leq^\infty_{\mathtt{A}} b\varphi$. A solution φ is *algebraic* if $\varphi : \mathtt{V} \to \mathtt{A}$. Let $\mathrm{Sol}(P)$ ($\mathrm{Sol}_{\mathtt{A}}(P)$ resp.) be the set of all the (algebraic resp.) solutions of P. Given a substitution φ and a set V of variables, let $\varphi|_V = \{(\alpha, \alpha\varphi) \mid \alpha \in V\}$.

The symbol ∞ will be considered as a symbol of sort $\mathtt{O}$. Hence, there will be no constraint of the form $\infty \leq^? \alpha$ with α a variable of sort $\mathtt{N}$. This means that an expression e will always be interpreted by a natural number and that $e\varphi \leq^\infty_{\mathtt{A}} f\varphi$ iff $e\varphi \leq_{\mathbb{N}} f\varphi$.

For technical reasons, we assume given an injection $\mathrm{x} : \mathtt{V} \to \mathtt{V}$ mapping every variable of sort $\mathtt{O}$ to a variable of sort $\mathtt{N}$.

A problem P can then be uniquely decomposed in 5 disjoint subsets as follows:

- P_0 the set of constraints $\alpha \leq^? \infty \in P$ such that $\alpha \notin \mathrm{Var}(P - \{\alpha \leq^? \infty\})$;
- P_1 the set of constraints $\infty \leq^? \alpha \in P$ such that $\alpha \notin \mathrm{Var}(P - \{\infty \leq^? \alpha\})$;
- P_2 the set of constraints $\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \in P$ or $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \leq^? \alpha \in P$ such that $\alpha \notin \mathrm{Var}(P - \{\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}, \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \leq^? \alpha\})$, and $\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \in P_2$ iff $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \leq^? \alpha \in P_2$;
- P_3 the set of constraints $e \leq^? f \in P$ with $\mathrm{Var}(e \leq^? f) \subseteq \{\mathrm{x}_\alpha \mid \alpha \in \mathrm{Var}(P_2)\}$;
- P_4 the remaining constraints.

Note that P_0, P_1, P_2 and $P_3 \cup P_4$ have disjoint sets of variables of sort $\mathtt{O}$. In the following, given $\varphi \in \mathrm{Sol}(P)$, let $\varphi_i = \varphi|_{\mathrm{Var}(P_i)}$ and $\varphi_{3,4} = \varphi_3 \cup \varphi_4$.

Applying the rules of Figure 8 (modulo $\simeq_{\mathtt{B}}$) until none is applicable will transform P into an equivalent problem Q such that either $Q = \bot$ and P is unsatisfiable, or $Q \neq \bot$ and P is satisfiable. Note that Figure 8 actually describes an infinite set of rules since a and b stand for arbitrary size expressions, e and f for arbitrary size expressions of sort $\mathtt{N}$, α for an arbitrary size variable, $\mathtt{c}$ and $\mathtt{d}$ for arbitrary constants, and $P \uplus Q$ for an arbitrary set with two disjoint parts, P and Q.

Lemma 16 (Termination) The rewrite relation generated by the rules of Figure 8 is terminating.

Proof. Given a problem R, let $\sharp R$ be the number of constraints in R, $|R|$ be the number of symbols in R, and $\ell(R)$ be the pair $(\sharp R_4^\flat, |R|)$, where $R_4^\flat$ is the set of constraints in R_4 that are not of the form $a \leq^? \infty$. Let $\succ$ be the well-founded ordering on problems such that $R \succ S$ iff $\ell(R)(>_{\mathbb{N}}, >_{\mathbb{N}})_{\mathrm{lex}}\ell(S)$. We prove that $R \succ S$ whenever $R \to S$.

No rule makes $\sharp R_4^\flat$ strictly increase. The rules 6-13 makes $\sharp R_4^\flat$ strictly decrease. For instance, in rule 10, the constraint $\infty \leq^? \alpha$ is moved from $R_4^\flat$ to S_0 since $\alpha \in \mathrm{Var}(P)$ and $\alpha \notin \mathrm{Var}(P\{(\alpha, \infty)\})$, and $S_4^\flat \subseteq P_4^\flat\{(\alpha, \infty)\}$. For the rules 1-5, when $\sharp R_4^\flat$ does not strictly decrease, then $|R|$ strictly decreases. For instance, $\sharp P_4^\flat$ is invariant by rule 4 ($a \leq^? \infty \notin R_4^\flat$ and $\{\alpha \leq^? \infty \mid \alpha \in \mathrm{Var}(a) - \mathrm{Var}(P)\} \subseteq S_0$). If $\mathrm{Var}(a) - \mathrm{Var}(P) = \emptyset$, then $S = P$ and $|R| > |S|$. Otherwise, $a = \mathtt{s}^{1+k}\alpha$ or $a = \mathtt{s}^{k+\alpha}\mathtt{c}$. In both cases, $S = P \cup \{\alpha \leq^? \infty\}$ and $|R| > |S|$. ■

Lemma 17 (Preservation of variables) If $R \to S$ and $S \neq \bot$, then $\mathrm{Var}(R) \subseteq \mathrm{Var}(S)$.

Proof. Straightforward. ■

Figure 8: Rules for deciding the satisfiability problem in the successor algebra

$$
\begin{array}{lrcl}
(1) & P \uplus \{\mathtt{s}\, a \leq^? \mathtt{s}\, b\} & \to & P \cup \{a \leq^? b\} \\
(2) & P \uplus \{\mathtt{s}^e \mathtt{c} \leq^? \mathtt{s}^f \mathtt{c}\} & \to & P \cup \{e \leq^? f\} \\
(3) & P \uplus \{\infty \leq^? \mathtt{s}^{1+k} \alpha\} & \to & P \cup \{\infty \leq^? \alpha\} \\
(4) & P \uplus \{a \leq^? \infty\} & \to & P \cup \{\alpha \leq^? \infty \mid \alpha \in \mathrm{Var}(a) - \mathrm{Var}(P)\} \\
 & & & \text{if } a \notin \mathtt{V} \text{ or } a \in \mathrm{Var}(P) \\
(5) & P \uplus \{a \leq^? \mathtt{s}^e a\} & \to & P \cup \{\alpha \leq^? \infty \mid \alpha \in \mathrm{Var}(a) - \mathrm{Var}(P)\} \\
(6) & P \uplus \{\infty \leq^? \mathtt{s}^e \mathtt{c}\} & \to & \bot \\
(7) & P \uplus \{\mathtt{s}^{1+k} \alpha \leq^? \mathtt{c}\} & \to & \bot \\
(8) & P \uplus \{\mathtt{s}^e \mathtt{c} \leq^? \mathtt{s}^f \mathtt{d}\} & \to & \bot \text{ if } \mathtt{c} \neq \mathtt{d} \\
(9) & P \uplus Q & \to & \bot \text{ if the variables of } Q \text{ are of sort } \mathtt{N} \text{ and } \mathrm{Sol}(Q) = \emptyset \\
(10) & P \uplus \{\infty \leq^? \alpha\} & \to & P\{(\alpha, \infty)\} \cup \{\infty \leq^? \alpha\} \text{ if } \alpha \in \mathrm{Var}(P) \\
(11) & P \uplus \{\mathtt{s}^{1+k} \alpha \leq^? \alpha\} & \to & P\{(\alpha, \infty)\} \cup \{\infty \leq^? \alpha\} \text{ if } \alpha \text{ is of sort } \mathtt{O} \\
(12) & P \uplus Q & \to & P\{(\alpha, \infty) \mid \alpha \in \mathrm{Var}(Q)\} \cup \{\infty \leq^? \alpha \mid \alpha \in \mathrm{Var}(Q)\} \\
 & & & \text{if the variables of } Q \text{ are of sort } \mathtt{O}, \\
 & & & Q \text{ is linear and } G(I(Q)) \text{ is an increasing cycle} \\
(13) & P \uplus \{\mathtt{s}^k \alpha \leq^? \mathtt{s}^e \mathtt{c}\} & \to & P\{(\alpha, \mathtt{s}^{\mathrm{x}_\alpha} \mathtt{c})\} \cup \{\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha} \mathtt{c}, \mathtt{s}^{\mathrm{x}_\alpha} \mathtt{c} \leq^? \alpha, k + \mathrm{x}_\alpha \leq^? e\} \\
 & & & \text{if } (k, e) \neq (0, \mathrm{x}_\alpha)
\end{array}
$$

Lemma 18 (Correctness) The rewrite relation generated by the rules of Figure 8 is correct: if $R \to S$ and $\varphi \in \mathrm{Sol}(S)$, then $\varphi|_{\mathrm{Var}(R)} \in \mathrm{Sol}(R)$, that is, $\{\varphi|_{\mathrm{Var}(R)} \mid \varphi \in \mathrm{Sol}(S)\} \subseteq \mathrm{Sol}(R)$.

Proof.

1. We have $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$. There are two cases:
   - $b\varphi = \infty$. If $b = \infty$, then $\mathtt{s}a \leq^{?} \mathtt{s}b$ is not a well-formed constraint. So, there are k and β such that $b = \mathtt{s}^k\beta$ and $\beta\varphi = \infty$. Therefore, $(\mathtt{s}b)\varphi = \infty$ and $(\mathtt{s}a)\varphi \leq^{\infty}_{\mathtt{A}} (\mathtt{s}b)\varphi$.
   - $a\varphi \leq_{\mathtt{A}} b\varphi$. Then, $(\mathtt{s}a)\varphi = \mathtt{s}(a\varphi) \leq_{\mathtt{A}} \mathtt{s}(b\varphi) = (\mathtt{s}b)\varphi$ and thus $(\mathtt{s}a)\varphi \leq^{\infty}_{\mathtt{A}} (\mathtt{s}b)\varphi$.
2. We have $e\varphi \leq_{\mathbb{N}} f\varphi$. Therefore, $(\mathtt{s}^e\mathtt{c})\varphi = \mathtt{s}^{e\varphi}\mathtt{c} \leq_{\mathtt{A}} \mathtt{s}^{f\varphi}\mathtt{c}$ and thus $(\mathtt{s}a)\varphi \leq^{\infty}_{\mathtt{A}} (\mathtt{s}b)\varphi$.

3-12. Immediate.

13. We have $\alpha\varphi \leq^{\infty}_{\mathtt{A}} \mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c}$, $\mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c} \leq^{\infty}_{\mathtt{A}} \alpha\varphi$ and $k + \mathrm{x}_\alpha\varphi \leq_{\mathbb{N}} e\varphi$. Since $\mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c} \neq \infty$, $\alpha\varphi \leq_{\mathtt{A}} \mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c}$ and $\mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c} \leq_{\mathtt{A}} \alpha\varphi$. Therefore, $(\mathtt{s}^k\alpha)\varphi \simeq_{\mathtt{B}} \mathtt{s}^{k+\mathrm{x}_\alpha\varphi}\mathtt{c} \leq_{\mathtt{A}} \mathtt{s}^{e\varphi}\mathtt{c} = (\mathtt{s}^e\mathtt{c})\varphi$. Now, given $a \leq^{?} b \in P$, we have $a\psi\varphi \leq^{\infty}_{\mathtt{A}} b\psi\varphi$ where $\psi = \{(\alpha, \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c})\}$. But $\psi\varphi \simeq_{\mathtt{B}} \varphi$. Therefore, $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$.

Lemma 19 (Completeness) The rewrite relation generated by the rules of Figure 8 is complete: if $R \to S$ and $\psi \in \mathrm{Sol}(R)$, then there is $\varphi \in \mathrm{Sol}(S)$ such that $\varphi|_{\mathrm{Var}(R)} = \psi$, that is, $\mathrm{Sol}(R) \subseteq \{\varphi|_{\mathrm{Var}(R)} \mid \varphi \in \mathrm{Sol}(S)\}$.

Proof.

1. We have $(\mathtt{s}a)\varphi \leq^{\infty}_{\mathtt{A}} (\mathtt{s}b)\varphi$. If $(\mathtt{s}b)\varphi = \infty$, then $b\varphi = \infty$ and $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$. Otherwise, $(\mathtt{s}a)\varphi \leq_{\mathtt{A}} (\mathtt{s}b)\varphi$, $a\varphi \leq_{\mathtt{A}} b\varphi$ and thus $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$.
2. We have $(\mathtt{s}^e\mathtt{c})\varphi = \mathtt{s}^{e\varphi}\mathtt{c} \leq^{\infty}_{\mathtt{A}} (\mathtt{s}^f\mathtt{c})\varphi = \mathtt{s}^{f\varphi}\mathtt{c}$. Hence, $\mathtt{s}^{e\varphi}\mathtt{c} \leq_{\mathtt{A}} \mathtt{s}^{f\varphi}\mathtt{c}$ and $e\varphi \leq_{\mathbb{N}} f\varphi$.

3-11. Immediate.

12. We first prove that, if $\alpha_1 \xrightarrow{k_1} \ldots \xrightarrow{k_n} \alpha_{n+1}$ is a path in $G(Q)$, $\varphi \in \mathrm{Sol}(Q)$ and $\Sigma_{i=1}^n k_i \geq 0$ ($\Sigma_{i=1}^n k_i < 0$ resp.), then $\mathtt{s}^{\Sigma_{i=1}^n k_i}\alpha_1\varphi \leq_{\mathtt{A}} \alpha_{n+1}\varphi$ ($\alpha_1\varphi \leq_{\mathtt{A}} \mathtt{s}^{-\Sigma_{i=1}^n k_i}\alpha_{n+1}\varphi$ resp.), by induction on n. If $n = 1$, this is immediate. We now prove it for $n+1$ assuming it for n. Let $k = \Sigma_{i=1}^n k_i$.
   - Case $k \geq 0$. Then, $\mathtt{s}^k\alpha_1\varphi \leq_{\mathtt{A}} \alpha_{n+1}\varphi$.
     - Case $k_{n+1} \geq 0$. Then, $\mathtt{s}^{k_{n+1}}\alpha_{n+1}\varphi \leq_{\mathtt{A}} \alpha_{n+2}\varphi$.
       - Case $k + k_{n+1} \geq 0$. By monotony and transitivity, $\mathtt{s}^{k+k_{n+1}}\alpha_1\varphi \leq_{\mathtt{A}} \alpha_{n+2}\varphi$.
       - Case $k + k_{n+1} < 0$. Impossible.

* Case $k_{n+1} < 0$. Then, $\alpha_{n+1}\varphi \leq_{\mathtt{A}} \mathtt{s}^{-k_{n+1}}\alpha_{n+2}\varphi$ and, by transitivity, $\mathtt{s}^{k}\alpha_1\varphi \leq_{\mathtt{A}} \mathtt{s}^{-k_{n+1}}\alpha_{n+2}\varphi$.
  - Case $k + k_{n+1} \geq 0$. Since $k_{n+1} \leq k$, $\mathtt{s}^{k+k_{n+1}}\alpha_1\varphi \leq_{\mathtt{A}} \alpha_{n+2}\varphi$.
  - Case $k + k_{n+1} < 0$. Since $k < -k_{n+1}$, $\alpha_1\varphi \leq_{\mathtt{A}} \mathtt{s}^{-k-k_{n+1}}\alpha_{n+2}\varphi$.

– Case $k < 0$. Symmetric to previous case.

Now, if $\alpha_{n+1} = \alpha_1$ and $k > 0$, then $\mathtt{s}^{k}\alpha_1\varphi \leq_{\mathtt{A}} \alpha_1\varphi$. Therefore, $\alpha\varphi = \infty$ for every $\alpha \in \mathrm{Var}(Q)$.

13. We have $\mathtt{s}^{k}\alpha\varphi \leq^{\infty}_{\mathtt{A}} \mathtt{s}^{e\varphi}\mathtt{c}$. Hence, there is l such that $\alpha\varphi = \mathtt{s}^{l}\mathtt{c}$ and $k + l \leq_{\mathbb{N}} e\varphi$. Therefore, $\varphi' = \varphi|_{\mathrm{Var}(R)} \cup \{(\mathrm{x}_\alpha, l)\} \in \mathrm{Sol}(S)$. ■

Combining correctness and completeness, we get that, if $R \to S$ then $\mathrm{Sol}(R) = \{\varphi|_{\mathrm{Var}(R)} \mid \varphi \in \mathrm{Sol}(S)\}$.

Lemma 20 Let P be a problem in $\mathtt{A} \cup \{\infty\}$ and Q a normal form of P wrt. the rules of Figure 8. If $Q \neq \bot$, then Q_4 is the disjoint union of:

- a linear and non-increasing set of constraints,
- a set of constraints of the form $\mathtt{s}^{e}\mathtt{c} \leq^{?} \mathtt{s}^{k}\alpha$.

Proof. Since P has no iterator symbols and iterators can only be introduced by rule 13, every sub-expression of Q that is headed by an iterator is of the form $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$. Hence, the size expressions occurring in Q are either ∞ or of the form $\mathtt{s}^{k}\alpha$, $\mathtt{s}^{k}\mathtt{c}$ or $\mathtt{s}^{k+\mathrm{x}_\alpha}\mathtt{c}$. Moreover, Q_4 cannot contain a constraint of the form:

- $a \leq^{?} \infty$ because of rule 4;
- $\infty \leq^{?} b$ because of the rules 3, 4, 6 and 10;
- $\mathtt{s}^{k}\alpha \leq^{?} \mathtt{s}^{e}\mathtt{c}$ because of the rules 7 and 13;
- $\mathtt{s}^{e}\mathtt{c} \leq^{?} \mathtt{s}^{f}\mathtt{d}$ because of the rules 2, 5, 8 and 9;
- $\mathtt{s}^{k}\alpha \leq^{?} \mathtt{s}^{l}\alpha$ because of the rules 5 and 11.

Therefore, a constraint of Q_4 can only be either of the form $\mathtt{s}^{k}\alpha \leq^{?} \mathtt{s}^{l}\beta$ with $kl = 0$ (because of rule 1), that is, a linear constraint, or of the form $\mathtt{s}^{k}\mathtt{c} \leq^{?} \mathtt{s}^{l}\beta$ or $\mathtt{s}^{k+\mathrm{x}_\alpha}\mathtt{c} \leq^{?} \mathtt{s}^{l}\beta$ with $kl = 0$ in both cases (because of rule 1). Moreover, linear constraints must be non-increasing because of rule 12. ■

Note that linear constraints and constraints of the form $\mathtt{s}^{e}\mathtt{c} \leq^{?} \mathtt{s}^{k}\alpha$ are always satisfiable by setting all their variables to ∞. Therefore, we can conclude:

Theorem 7 (Satisfiability) The satisfiability of a size problem in the successor algebra is decidable in polynomial time wrt. the number of symbols.

Proof. To decide the satisfiability of a problem P, we can proceed as follows.

First, we apply the rules of Figure 8 as long as possible. Whether a rule can be applied is decidable. Indeed, the satisfiability of a set of constraints of the form $e \leq^? f$ (for firing rule 9) is decidable in polynomial time [Pra77] since, if $\mathrm{Var}_{\mathtt{0}}(Q) = \emptyset$, then $\mathrm{Sol}(Q) = \mathrm{Sol}(I(Q))$. Whether a linear problem has an increasing cycle (for rule 12) is decidable by the same algorithm. Indeed, either there is an increasing cycle and no bounded solution exists, or there is no increasing cycle and the problem is satisfiable. Hence, as the rewrite system terminates, we must end on a normal form Q.

Now, if $Q = \bot$, then P is unsatisfiable, by completeness of the rewrite system. Otherwise, Q_3 is satisfiable by rule 10. So, let $\varphi \in \mathrm{Sol}(Q_3)$. Then, as one can easily check, $\varphi \cup \{(\alpha, \infty) \mid \alpha \in \mathrm{Var}(Q_1) \cup \mathrm{Var}(Q_4)\} \cup \{(\alpha, \mathtt{s}^{\mathrm{x}_\alpha \varphi}\mathtt{c}) \mid \alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \in Q_2\} \in \mathrm{Sol}(Q)$. Therefore, P is satisfiable by completeness of the rewrite system.

We now prove that the complexity of computing Q is polynomial in the size of P. Deciding whether a rule can be applied and computing the result of applying a rule can be done in polynomial time. Therefore, it suffices to show that, from P to Q, the size of intermediate problems and the number of rewrite steps are polynomially bounded by $|P|$.

Let R be an intermediate problem. Since only variables of the form x_α with $\alpha \in \mathrm{Var}(P)$ can be introduced (by rule 13), we have $\sharp\mathrm{Var}(R) \leq 2\sharp\mathrm{Var}(P)$. Since there are at most two variables per constraint, we have $\sharp\mathrm{Var}(P) \leq 2\sharp P$. Moreover, for all $i \in \{1, 2, 3\}$, we have $\sharp R_i \leq \sharp\mathrm{Var}(P)$. Hence, $\sharp R_i \leq 2\sharp P$. Now, $R_4 = R_4^\flat \uplus \{a \leq^? \infty \in R_4 | a \notin \mathtt{V}\} \uplus \{\alpha \leq^? \infty \in R_4\}$. After the termination proof, $\sharp R_4^\flat \leq \sharp P_4^\flat \leq \sharp P$. Next, one can easily check that $\sharp\{a \leq^? \infty \in R_4 | a \notin \mathtt{V}\}$ can only decrease (by rule 4). Hence, $\sharp\{a \leq^? \infty \in R_4 | a \notin \mathtt{V}\} \leq \sharp P$. Finally, $\sharp\{\alpha \leq^? \infty \in R_4\} \leq \sharp\mathrm{Var}(P) \leq 2\sharp P$. Therefore, $\sharp R_4 \leq 4\sharp P$ and $\sharp R \leq 10\sharp P$. Now, since every constraint contains at least 2 symbols, $\sharp P \leq |P|/2$. Hence, $\sharp R \leq 5|P|$.

We now prove that the maximum size of a constraint in R, $\|R\|_\infty$, is also linearly bounded by $|P|$. In rules 1-5, a constraint is removed or replaced by a smaller one. In rules 10-12, α is replaced by ∞, which does not change the size. On the other hand, in rule 13, α, which size is 1, is replaced by $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$, which size is 3, and the constraints $\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$ and $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \leq^? \alpha$, which size are 4, are added. Hence, if $R \to S$ and $S \neq \bot$, then $\|S\|_\infty \leq \max(4, \|R\|_\infty + 3) = \|R\|_\infty + 3$ since $\|R\|_\infty \geq 2$. However, such a substitution cannot be iterated since x_α is of sort $\mathtt{N}$. Hence, $\|R\|_\infty \leq \|P\|_\infty + 3 \leq |P| + 3$.

Therefore, $|R| \leq \sharp R \times \|R\|_\infty \leq M = 5|P|(|P|+3)$. Hence, after the termination proof (lexicographic ordering), there are at most $\sharp P_4^\flat \times M$ rewrite steps. In conclusion, since $\sharp P_4^\flat \leq \sharp P \leq |P|/2$, there are at most $5|P|^2(|P| + 3)/2$ rewrite steps. ■

Our procedure can be related to the one described in [BGP05] where, as many works on type inference, the authors consider constrained types. But they do not bring out the properties of the size algebra and, in particular that, in the successor algebra, satisfiable sets of constraints have a most general solution.

7.2 Computing the most general solution

We now turn to the problem of whether, in the successor algebra, a satisfiable problem has a most general solution and, if so, how to compute it.

We first prove some properties of the specialization quasi-ordering $\sqsubseteq$ and its associated equivalence relation $\equiv$ (see Definition 18).

First note that a substitution $\varphi : \mathtt{V} \to \mathtt{A} \cup \{\infty\}$ uniquely determines two functions $\varphi_{\mathtt{N}} : \mathtt{V} \to \mathbb{N}$ and $\varphi_{\mathtt{V}} : \mathtt{V} \to \mathtt{V} \cup \mathtt{F}_0 \cup \{\infty\}$ such that, for all α, $\alpha\varphi = \mathtt{s}^{\alpha\varphi_{\mathtt{N}}}\alpha\varphi_{\mathtt{V}}$, and $\alpha\varphi_{\mathtt{N}} = 0$ whenever $\alpha\varphi_{\mathtt{V}} = \infty$.

Lemma 21 $\varphi \sqsubseteq \psi$ iff there is $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{F}_0 \cup \{\infty\}$ such that $\varphi\rho \leq^{\infty}_{\mathtt{A}} \psi$.

Proof. Assume that there is θ such that $\varphi\theta \leq^{\infty}_{\mathtt{A}} \psi$. Let $\rho = \theta_{\mathtt{V}}|_{\mathrm{Var}(\varphi_{\mathtt{V}})}$, where $\mathrm{Var}(\varphi_{\mathtt{V}}) = \bigcup\{\mathrm{Var}(\alpha\varphi_{\mathtt{V}}) \mid \alpha \in \mathrm{dom}(\varphi_{\mathtt{V}})\}$. Then, one can easily check that $\varphi\rho \leq^{\infty}_{\mathtt{A}} \psi$. If $\alpha\varphi_{\mathtt{V}} \notin \mathtt{V}$, then $\alpha\varphi\rho = \alpha\varphi\theta \leq^{\infty}_{\mathtt{A}} \alpha\psi$. Otherwise, $\alpha\varphi\rho = \mathtt{s}^{\alpha\varphi_{\mathtt{N}}}\alpha\varphi_{\mathtt{V}}\theta_{\mathtt{V}} \leq^{\infty}_{\mathtt{A}} \mathtt{s}^{\alpha\varphi_{\mathtt{N}}+\alpha\varphi_{\mathtt{V}}\theta_{\mathtt{N}}}\alpha\varphi_{\mathtt{V}}\theta_{\mathtt{V}} = \alpha\varphi\theta \leq^{\infty}_{\mathtt{A}} \alpha\psi$. ■

Lemma 22 $\varphi_2 \equiv \varphi_1$ iff $\varphi_2 = \varphi_1\xi$ for some permutation $\xi : \mathtt{V} \to \mathtt{V}$.

Proof. In [Hue76], Huet proved this result when $\leq^{\infty}_{\mathtt{A}}$ is the equality. His proof can be easily adapted to our more general situation since, when $\alpha, \beta \in \mathtt{V}$, $\alpha \leq^{\infty}_{\mathtt{A}} \beta$ iff $\alpha = \beta$. By assumption, there are θ_1 and θ_2 such that $\varphi_1\theta_1 \leq^{\infty}_{\mathtt{A}} \varphi_2$ and $\varphi_2\theta_2 \leq^{\infty}_{\mathtt{A}} \varphi_1$. Hence, $\varphi_1\theta_1\theta_2 \leq^{\infty}_{\mathtt{A}} \varphi_1$ and $\varphi_2\theta_2\theta_1 \leq^{\infty}_{\mathtt{A}} \varphi_2$. Let $V = \mathrm{dom}(\varphi_1) \cup \mathrm{dom}(\varphi_2)$ and $V_i = \bigcup\{\mathrm{Var}(\alpha\varphi_i) \mid \alpha \in V\}$. Given $\alpha \in V_1$, either $\alpha\varphi_1 = \mathtt{s}^{\beta}\mathtt{c}$ and thus $\beta\theta_1\theta_2 \leq^{\infty}_{\mathtt{A}} \beta$, or $\alpha\varphi_1 = \mathtt{s}^k\beta$ and thus $\beta\theta_1\theta_2 \leq^{\infty}_{\mathtt{A}} \beta$. Hence, for all $\alpha \in V_1$, $\beta\theta_1\theta_2 = \beta$. Similarly, for all $\beta \in V_2$, $\beta\theta_2\theta_1 = \beta$. Therefore, θ_1 is an injection from V_1 to V_2, and θ_2 an injection from V_2 to V_1. Hence, V_1 and V_2 are equipotent, and $V_1 - V_2$ and $V_2 - V_1$ as well. Let ν be any bijection from $V_2 - V_1$ to $V_1 - V_2$, and $\xi = \{(\alpha, \alpha\theta_1) \mid \alpha \in V_1\} \cup \{(\alpha, \alpha\nu) \mid \alpha \in V_2 - V_1\}$. The function ξ is a bijection. We now prove that, for all α, $\alpha\varphi_1\xi = \alpha\varphi_2$. There are three cases:

- $\alpha\varphi_1 = \infty$. Since $\alpha\varphi_1\theta_1 \leq^{\infty}_{\mathtt{A}} \alpha\varphi_2$, we have $\alpha\varphi_1\xi = \alpha\varphi_2 = \infty$.
- $\alpha\varphi_1 = \mathtt{s}^k\beta$. Then, $\beta \in V_1$ and $\alpha\varphi_1\xi = \mathtt{s}^k\beta\theta_1$. Since $\varphi_1\theta_1 \leq^{\infty}_{\mathtt{A}} \varphi_2$, we have $\alpha\varphi_2 = \mathtt{s}^{k+l}\beta\theta_1$ for some l. But since $\varphi_2\theta_2 \leq^{\infty}_{\mathtt{A}} \varphi_1$, $l = 0$ and $\alpha\varphi_1\xi = \alpha\varphi_2$.
- $\alpha\varphi_1 = \mathtt{s}^k\mathtt{c}$. Since $\alpha\varphi_1\theta_1 \leq^{\infty}_{\mathtt{A}} \alpha\varphi_2$, either $\alpha\varphi_2 = \infty$ or $\alpha\varphi_2 = \mathtt{s}^{k+l}\mathtt{c}$ for some l. Since $\alpha\varphi_2\theta_2 \leq^{\infty}_{\mathtt{A}} \alpha\varphi_1$, $l = 0$ and $\alpha\varphi_1\xi = \alpha\varphi_2$. ■

Even after normalization, a problem P may contain two constraints $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^k\alpha$ and $\mathtt{s}^f\mathtt{d} \leq^? \mathtt{s}^l\alpha$ with $\mathtt{c} \neq \mathtt{d}$, in which case, for any $\varphi \in \mathrm{Sol}(P)$, $\alpha\varphi = \infty$. More generally, α must be set to ∞ if $\mathtt{c} \leq_P \alpha$, $\mathtt{d} \leq_P \alpha$ and $\mathtt{c} \neq \mathtt{d}$, where $\leq_P$ is defined as follows:

Definition 26 (Affine problem) Given a set of constraints P, let $\leq_P$ be the smallest quasi-ordering on $\mathtt{V} \cup \mathtt{F}_0$ such that $\alpha \leq_P \beta$ if there is a constraint $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in P$, and $\mathtt{c} \leq_P \beta$ if there is a constraint $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in P$. A problem P is *affine* if:

Figure 9: Additional rule for finding all the variables that must be set to ∞

$$(14) \quad P \quad \to \quad P\{(\alpha, \infty)\} \cup \{\infty \leq^? \alpha\} \quad \text{if } \mathtt{c} \leq_P \alpha, \mathtt{d} \leq_P \alpha, \mathtt{c} \neq \mathtt{d}$$

- it only contains constraints of the form $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta$ or $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta$ with $\alpha \neq \beta$;
- there is no tuple $(\alpha, \mathtt{c}, \mathtt{d})$ such that $\mathtt{c} \leq_P \alpha$, $\mathtt{d} \leq_P \alpha$ and $\mathtt{c} \neq \mathtt{d}$.

An affine problem is non-increasing if its linear constraints are non-increasing. Given an affine problem P, let its associated integer problem be $I(P) = \{k + \alpha \leq^? l + \beta \mid \mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in P\} \cup \{e \leq^? l + \beta \mid \mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in P\}$.

To detect all the variables that must be set to ∞, we extend the rules of Figure 8 with the rule of Figure 9. The results of the previous section are easily extended when adding this new rule:

Lemma 23 Let $\to$ be the relation generated by the rules of Figure 8 and 9.

- $\to$ terminates.
- If $R \to S$ and $S \neq \bot$, then $\mathrm{Var}(R) \subseteq \mathrm{Var}(S)$.
- $\mathrm{Sol}(R) = \{\varphi|_{\mathrm{Var}(P)} \mid \varphi \in \mathrm{Sol}(S)\}$.
- Let P be a problem in $\mathtt{A} \cup \{\infty\}$ and Q a normal form of P wrt. $\to$. If $Q \neq \bot$, then Q_4 is a non-increasing affine problem.

Proof. Rule 15 makes strictly decrease $\mathrm{card}(R_4^\flat)$ and preserves variables. One can easily check the third item since, if $\mathtt{c} \leq_P \alpha$ and $\varphi \in \mathrm{Sol}(P)$, then either $\alpha\varphi = \infty$ or $\alpha\varphi = \mathtt{s}^k\mathtt{c}$ for some $k \in \mathbb{N}$. ■

Assume now that we have a problem P. Let Q be a normal form of P wrt. the rules of Figure 8 and 9. As just seen, we have $\mathrm{Sol}(P) = \{\varphi|_{\mathrm{Var}(Q)} \mid \varphi \in \mathrm{Sol}(Q)\}$. We now prove that the most general solution of P, if it exists, is the restriction to $\mathrm{Var}(Q)$ of the most general solution of Q:

Lemma 24 Let Q be any normal form of P wrt. the rules of Figures 8 and 9. If P has a most general solution ψ, then Q has a most general solution φ and $\varphi|_{\mathrm{Var}(Q)} = \psi$. Conversely, if Q has a most general solution φ, then $\varphi|_{\mathrm{Var}(Q)}$ is the most general solution of P.

Proof. Assume that P has a most general solution ψ. Then, there is $\varphi \in \mathrm{Sol}(Q)$ such that $\psi = \varphi|_{\mathrm{Var}(P)}$. Assume that there is $\varphi' \in \mathrm{Sol}(Q)$ such that $\varphi \not\sqsubseteq \varphi'$. Since $\varphi'|_{\mathrm{Var}(P)} \in \mathrm{Sol}(P)$, $\psi \sqsubseteq \varphi'|_{\mathrm{Var}(P)}$, that is, there is θ such that, for all α, $\alpha\psi \leq_{\mathtt{A}}^\infty \alpha\varphi'|_{\mathrm{Var}(P)}$. Since $\varphi \not\sqsubseteq \varphi'$, there is α such that $\alpha\varphi\theta \not\leq_{\mathtt{A}}^\infty \alpha\varphi'$. Hence, $\alpha \notin \mathrm{Var}(P)$ and there are $\beta \in \mathrm{Var}(P)$, l and $\mathtt{c}$ such that $\alpha = \mathrm{x}_\beta$ and

$\beta \leq^? \mathtt{s}^\alpha\mathtt{c} \in Q$. Therefore, $\alpha\varphi$ is a closed term of sort $\mathtt{N}$ and $\alpha\varphi\theta = \alpha\varphi \not\leq_{\mathbb{N}} \alpha\varphi'$. Since $\varphi, \varphi' \in \mathrm{Sol}(Q)$, $\beta\varphi = \mathtt{s}^{\alpha\varphi}\mathtt{c}$ and $\beta\varphi' = \mathtt{s}^{\alpha\varphi'}\mathtt{c}$. Since $\beta \in \mathrm{Var}(P)$ and $\psi = \varphi|_{\mathrm{Var}(P)} \sqsubseteq \varphi'|_{\mathrm{Var}(P)}$, $\beta\varphi \leq^\infty_{\mathtt{A}} \beta\varphi'$. Therefore, $\alpha\varphi \leq^\infty_{\mathtt{A}} \alpha\varphi'$. Contradiction.

Assume now that Q has a most general solution φ, and let $\psi \in \mathrm{Sol}(P)$. Then, there is $\varphi' \in \mathrm{Sol}(Q)$ such that $\psi = \varphi'|_{\mathrm{Var}(P)}$ and $\varphi \sqsubseteq \varphi'$, *i.e.* there is θ such that $\varphi\theta \leq^\infty_{\mathtt{A}} \varphi'$. Hence, $\mathrm{dom}(\theta) \subseteq \mathrm{dom}(\varphi) \cup \mathrm{dom}(\varphi') \subseteq \mathrm{Var}(Q)$. For all $\alpha \in \mathrm{Var}(P)$, $\mathrm{Var}(\alpha\varphi) \subseteq \mathrm{Var}(P)$ since φ maps variables of sort $\mathtt{O}$ to terms of sort $\mathtt{O}$ and the variables of $\mathrm{Var}(Q) - \mathrm{Var}(P)$ are of sort $\mathtt{N}$. For all $\alpha \in \mathrm{Var}(Q) - \mathrm{Var}(P)$, $\alpha\varphi$ is closed. Thus, $\varphi|_{\mathrm{Var}(P)}\theta \leq^\infty_{\mathtt{A}} \varphi'|_{\mathrm{Var}(P)} = \psi$.[8] ■

Hence, we are left to find whether Q has a most general solution or not. One can easily check that $\mathrm{Sol}(Q)$ is fully determined by the solutions of Q_0 and $Q_3 \cup Q_4$. Moreover, the most general solution of Q is fully determined by the most general solution of $Q_3 \cup Q_4$:

Lemma 25 $\mathrm{Sol}(Q) = \{\varphi_0 \cup \varphi_1 \cup \overline{\varphi_{3,4}} \cup \varphi_{3,4} \mid \varphi_0 \in \mathrm{Sol}(Q_0), \varphi_{3,4} \in \mathrm{Sol}(Q_3 \cup Q_4)\}$, where $\varphi_1 = \{(\alpha, \infty) \mid \alpha \in \mathrm{Var}(Q_1)\}$ is the unique solution of Q_1 and $\overline{\varphi_{3,4}} = \{(\alpha, \mathtt{s}^{\mathtt{x}_\alpha\varphi_{3,4}}\mathtt{c}) \mid \alpha \leq^? \mathtt{s}^{\mathtt{x}_\alpha}\mathtt{c} \in Q_2\} \in \mathrm{Sol}(Q_2)$.

Moreover, if Q has a most general solution φ, then $\varphi_0 = \mathrm{id}$ and $\varphi_{3,4}$ is the most general solution of $Q_3 \cup Q_4$. Conversely, if $Q_3 \cup Q_4$ has a most general solution ψ, then $\varphi_1 \cup \overline{\psi} \cup \psi$ is the most general solution of Q.

Proof. Assume that $\varphi = \mathrm{mgs}(Q)$ and let $\psi \in \mathrm{Sol}(Q_3 \cup Q_4)$. Then, $\varphi_1 \cup \overline{\psi} \cup \psi \in \mathrm{Sol}(Q)$. Thus, there is θ such that $\varphi\theta \leq^\infty_{\mathtt{A}} \varphi_1 \cup \overline{\psi} \cup \psi$. Let $V = \bigcup\{\mathrm{Var}(\alpha\varphi) \mid \alpha \in \mathrm{Var}(Q_4)\}$. Then, $\varphi\theta|_V \leq^\infty_{\mathtt{A}} \varphi_1 \cup \overline{\psi} \cup \psi$. Indeed, if $\alpha \in \mathrm{Var}(Q_1)$, then $\alpha\varphi_1 = \infty$. If $\alpha \in \mathrm{Var}(Q_2)$, then $\alpha\varphi = \mathtt{s}^{\mathtt{x}_\alpha\varphi}\mathtt{c}$ for some $\mathtt{c}$. And if $\alpha \in \mathrm{Var}(Q_3)$, then $\alpha\varphi = k$ for some k. Therefore, $\varphi_{3,4}\theta|_V \leq^\infty_{\mathtt{A}} \psi$.

Assume now that $\psi = \mathrm{mgs}(Q_3 \cup Q_4)$ and let $\varphi \in \mathrm{Sol}(Q)$. Then, $\varphi_{3,4} \in \mathrm{Sol}(Q_3 \cup Q_4)$. Thus, there is θ such that $\psi\theta \leq^\infty_{\mathtt{A}} \varphi_{3,4}$. Hence, $\overline{\psi}\theta \leq^\infty_{\mathtt{A}} \overline{\varphi_{3,4}}$. Therefore, $(\varphi_1 \cup \overline{\psi} \cup \psi)\theta = \varphi_1 \cup \overline{\psi}\theta \cup \psi\theta \leq^\infty_{\mathtt{A}} \varphi$. ■

We now prove that any non-increasing affine problem has a most general *algebraic* solution. To this end, we first show that the set of algebraic solutions of an affine problem Q is fully determined by the set of solutions of $I(Q)$. Then, we prove that any satisfiable integer problem has a smallest solution.

Lemma 26 If Q is affine, then:

- there is a strictly monotone map $\psi \mapsto \widehat{\psi}$ from $(\mathrm{Sol}(I(Q)), \leq)$ to $(\mathrm{Sol}_{\mathtt{A}}(Q), \sqsubseteq)$;
- $\varphi \mapsto \varphi_{\mathtt{N}}$ is a monotone map from $(\mathrm{Sol}_{\mathtt{A}}(Q), \sqsubseteq)$ to $(\mathrm{Sol}(I(Q)), \leq)$;
- $\mathrm{Sol}_{\mathtt{A}}(Q) = \{\widehat{\psi}\rho \mid \psi \in \mathrm{Sol}(I(Q)), \rho : \mathtt{V} \to \mathtt{V}\}$;
- if $\mathrm{mgs}(Q) = \varphi$, then $\mathrm{mgs}(I(Q)) = \varphi_{\mathtt{N}}$;

[8] Note that $\varphi \sqsubseteq \psi \Rightarrow \varphi|_V \sqsubseteq \psi|_V$ does not hold in general. Take for instance the permutation of x and y for φ, its inverse for θ, the identity for ψ, and $V = \{x\}$. Then, $\varphi|_V \not\sqsubseteq \psi|_V$ since $y\varphi|_V\theta = y\theta = x$ and $y\psi|_V = y$.

- if $\mathrm{mgs}(I(Q)) = \psi$, then $\mathrm{mgs}(Q) = \widehat{\psi}$.

  Proof.

- Let $\sim$ be the smallest equivalence relation on $\mathtt{V} \cup \mathtt{F}_0$ containing $<_Q$ and such that $\alpha \sim \mathtt{0}$ if α is of sort $\mathtt{N}$, $[\alpha]$ be the equivalence class of α modulo $\sim$, and $\eta : \mathtt{V} \cup \mathtt{F}_0/\sim \to \mathtt{V} \cup \mathtt{F}_0$ be any injection such that $\eta(X) = \mathtt{c}$ iff $\mathtt{c} \in X$. Such a function exists since Q is affine and thus every equivalence class contains at most one constant.

  Now, given $\psi \in \mathrm{Sol}(I(Q))$, let $\alpha\widehat{\psi} = \mathtt{s}^{\alpha\psi}\alpha^*$ where $\alpha^* = \eta([\alpha])$.

  We check that $\widehat{\psi} \in \mathrm{Sol}_{\mathtt{A}}(Q)$. If $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in Q$, then $\mathtt{s}^{k+\alpha\psi}\alpha^* \leq \mathtt{s}^{l+\beta\psi}\beta^*$ since $k + \alpha\psi \leq l + \beta\psi$ and $\alpha^* = \beta^*$. If $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in Q$, then $\mathtt{s}^{e\psi}\mathtt{c} \leq \mathtt{s}^{l+\beta\psi}\beta^*$ since $e\psi \leq l + \beta\psi$ and $\mathtt{c} = \beta^*$.

  Then, one can easily check that $\psi \mapsto \widehat{\psi}$ is injective ($\psi_1 = \psi_2$ whenever $\widehat{\psi_1} = \widehat{\psi_2}$) and monotone wrt. $\leq^\infty_{\mathtt{A}}$ ($\widehat{\varphi} \leq^\infty_{\mathtt{A}} \widehat{\psi}$ whenever $\varphi \leq \psi$) and thus wrt. $\sqsubseteq$.

- Let $\varphi \in \mathrm{Sol}_{\mathtt{A}}(Q)$. We check that $\varphi_{\mathtt{N}} \in \mathrm{Sol}(I(Q))$ and $\varphi_{\mathtt{V}}$ is invariant by $\sim$. If $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in Q$, then $\mathtt{s}^{k+\alpha\varphi_{\mathtt{N}}}\alpha\varphi_{\mathtt{V}} \leq \mathtt{s}^{l+\beta\varphi_{\mathtt{N}}}\beta\varphi_{\mathtt{V}}$. So, $\alpha\varphi_{\mathtt{V}} = \beta\varphi_{\mathtt{V}}$ and $k + \alpha\varphi_{\mathtt{N}} \leq l + \beta\varphi_{\mathtt{N}}$. Assume now that $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in Q$. Then, $\mathtt{s}^{e\varphi_{\mathtt{N}}}\mathtt{c} \leq \mathtt{s}^{l+\beta\varphi_{\mathtt{N}}}\beta\varphi_{\mathtt{V}}$. So, $\mathtt{c} = \beta\varphi_{\mathtt{V}}$ and $e\varphi_{\mathtt{N}} \leq l + \beta\varphi_{\mathtt{N}}$.

  We now check that $\varphi \mapsto \varphi_{\mathtt{N}}$ is monotone. Let $\psi \in \mathrm{Sol}_{\mathtt{A}}(Q)$ such that $\varphi \sqsubseteq \psi$. Hence, there is $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{F}_0$ such that $\varphi\rho \leq^\infty_{\mathtt{A}} \psi$. Therefore, $\varphi_{\mathtt{N}} \leq \psi_{\mathtt{N}}$.

- Let $\psi \in \mathrm{Sol}(I(Q))$ and $\rho : \mathtt{V} \to \mathtt{V}$. Then, $\widehat{\psi} \in \mathrm{Sol}_{\mathtt{A}}(Q)$ and $\widehat{\psi}\rho \in \mathrm{Sol}_{\mathtt{A}}(Q)$ since $\mathrm{Sol}_{\mathtt{A}}(Q)$ is closed by algebraic substitution.

  Given $\varphi \in \mathrm{Sol}(Q)$, let $\alpha^*\rho = \alpha\varphi_{\mathtt{V}}$ for all $\alpha^* \in \mathtt{V}$. The function ρ is well defined since $\varphi_{\mathtt{V}}$ is invariant by $\sim$. Now, one can easily check that $\varphi = \widehat{\varphi_{\mathtt{N}}}\rho$, which in particular implies that $\widehat{\varphi_{\mathtt{N}}} \sqsubseteq \varphi$.

- Assume that Q has a most general solution φ, and let $\psi \in \mathrm{Sol}(I(Q))$. Then, $\widehat{\psi} \in \mathrm{Sol}(Q)$ and $\varphi \sqsubseteq \widehat{\psi}$. Therefore, $\varphi_{\mathtt{N}} \leq \widehat{\psi}_{\mathtt{N}} = \psi$.

- Assume that $I(Q)$ has a most general solution ψ, and let $\varphi \in \mathrm{Sol}_{\mathtt{A}}(Q)$. Then, $\varphi_{\mathtt{N}} \in \mathrm{Sol}(I(Q))$ and $\psi \leq \varphi_{\mathtt{N}}$. Therefore, $\widehat{\psi} \leq^\infty_{\mathtt{A}} \widehat{\varphi_{\mathtt{N}}} \sqsubseteq \varphi$. ■

If Q is a non-increasing affine problem, then $I(Q)$ is non-increasing. Therefore, $I(Q)$ is satisfiable and Q has an algebraic solution. So, if Q has a most general solution, then it must be algebraic. Moreover, the most general solution of Q is given by the most general solution of $I(Q)$. We now prove that:

Lemma 27 Any satisfiable integer problem has a smallest solution (wrt. the pointwise extension of $\leq_{\mathbb{N}}$).

Proof. Assume that P has no smallest solution. Then, it has two incomparable solutions φ_1 and φ_2. But, as we are going to see, $\varphi = \min(\varphi_1, \varphi_2)$ is a solution strictly smaller than both φ_1 and φ_2. Contradiction. We proceed by case on the form of the constraints of P:

- $\alpha + i \leq^? \beta$. Then, $\alpha\varphi_1 + i \leq \beta\varphi_1$ and $\alpha\varphi_2 + i \leq \beta\varphi_2$.
  - If $\alpha\varphi_1 \leq \alpha\varphi_2$ and $\beta\varphi_1 \leq \beta\varphi_2$, then $\alpha\varphi + i = \alpha\varphi_1 + i \leq \beta\varphi_1 = \beta\varphi$.
  - If $\alpha\varphi_1 \leq \alpha\varphi_2$ and $\beta\varphi_2 < \beta\varphi_1$, then $\alpha\varphi + i = \alpha\varphi_1 + i \leq \alpha\varphi_2 + i \leq \beta\varphi_2 = \beta\varphi$.
  - If $\alpha\varphi_2 < \alpha\varphi_1$ and $\beta\varphi_1 \leq \beta\varphi_2$, then $\alpha\varphi + i = \alpha\varphi_2 + i < \alpha\varphi_1 + i \leq \beta\varphi_1 = \beta\varphi$.
  - If $\alpha\varphi_2 < \alpha\varphi_1$ and $\beta\varphi_2 < \beta\varphi_1$, then $\alpha\varphi + i = \alpha\varphi_2 + i \leq \beta\varphi_2 = \beta\varphi$.
- $\alpha \leq^? i$. Then, $\alpha\varphi_1 \leq i$ and $\alpha\varphi_2 \leq i$. Therefore, $\alpha\varphi \leq i$.
- $i \leq^? \alpha$. Then, $i \leq \alpha\varphi_1$ and $i \leq \alpha\varphi_2$. Therefore, $i \leq \alpha\varphi$. ■

Lemma 28 The smallest solution of a satisfiable integer problem can be computed in polynomial time.

Proof. Let P be a satisfiable integer problem whose variables are $\alpha_1, \ldots, \alpha_n$. We first prove that P is equivalent to a problem in the dioid $(\overline{\mathbb{Z}}_{\max}^{n\times n}, \oplus, \otimes)$ where $\overline{\mathbb{Z}}_{\max} = \mathbb{Z} \cup \{\pm\infty\}$, $\oplus = \max$ and $\otimes = +$ both applied component wise [BCOQ92].

Wlog. we can assume that P contains no constraint of the form $0 + k \leq 0$ (k is a constant; since P is satisfiable, this constraint is always satisfied). Hence, P contains only constraints of the following two forms:

- $\alpha_i + k \leq \alpha_j$ or $0 + k \leq \alpha_i$,
- $\alpha_i + k \leq 0$.

Then, one can check that $\psi \in \mathrm{Sol}(P)$ iff there is $x \in \overline{\mathbb{Z}}_{\max}^{n\times n}$ such that $ax \oplus b \leq x \leq c$ and $\psi(\alpha_i) = x_{i1}$, where $a_{ij} = \max(\{k \in \overline{\mathbb{Z}}_{\max} \mid \alpha_j + k \leq^? \alpha_i \in P\} \cup \{-\infty\})$, $b_{ij} = \max(\{k \in \overline{\mathbb{Z}}_{\max} \mid 0 + k \leq^? \alpha_i \in P\} \cup \{-\infty\})$ and $c_{ij} = \min(\{k \in \overline{\mathbb{Z}}_{\max} | \alpha_i + (-k) \leq^? 0 \in P\} \cup \{+\infty\})$.

By Theorem 4.75 in [BCOQ92], $ax \oplus b \leq x$ has a smallest solution that is a^*b where $a^* = \bigoplus_{k=0}^{+\infty} a^k$. Since P is satisfiable, P is not increasing. Hence, in this case, $a^* = \bigoplus_{k=0}^{n} a^k$ [CG79] (Theorem 3.20 in [BCOQ92]). Therefore, $\psi \in \mathrm{Sol}(P)$ iff $a^*b \leq c$, and the smallest solution of P is the function ψ such that $\psi(\alpha_i) = \bigoplus_{k=1}^{n} a^*_{ik} b_{k1}$, which can clearly be computed in polynomial time. ■

We can therefore conclude:

Theorem 8 In the successor algebra, any satisfiable size problem has a most general solution that can be computed in polynomial time.

Proof. To compute the most general solution of P, we can proceed as follows. Let Q be a normal form of P wrt. to the rules of Figure 8 and 9, and ψ be the smallest solution of $I(Q_3 \cup Q_4)$. Then, return $\varphi_1 \cup \widehat{\psi}$, where $\varphi_1 = \{(\alpha, \infty) \mid \alpha \in \mathrm{Var}(Q_1)\}$.

After the previous lemmas, this algorithm is clearly correct. We now check that it is complete. By completeness, $Q \neq \bot$. Since $Q_3 \cup Q_4$ is affine, $\mathrm{mgs}(Q_3 \cup Q_4) = \widehat{\psi}$. Therefore, $\mathrm{mgs}(Q) = \varphi_1 \cup \overline{\widehat{\psi}} \cup \widehat{\psi}$ and $\mathrm{mgs}(P) = \varphi_1 \cup \widehat{\psi}$.

Whether rule 14 can be fired can be decided in polynomial time. Hence, computing Q can be done in polynomial time. Now, computing ψ can be done in polynomial time. Hence, mgs(P) can be computed in polynomial time. ■

8 Minimality property in the successor algebra

In this section, we study the minimality property of Theorem 2 in the successor algebra. To this end, we need to know how the size of $l_i\theta$ depends on the sizes of the subterms $x\theta$ where x is a variable of l_i.

To make things simpler, we will assume that the size of a term headed by a constructor c only depends on the size of the recursive arguments of c, that is, following Definition 13, we assume that, for every constructor c:

- $\alpha^{\mathsf{c}}_1 = \ldots = \alpha^{\mathsf{c}}_{\mathrm{p}^{\mathsf{c}}}$;
- $\alpha^{\mathsf{c}}_{\mathrm{p}^{\mathsf{c}}+1} = \ldots = \alpha^{\mathsf{c}}_{\mathrm{q}^{\mathsf{c}}} = \infty$;
- $\sigma^{\mathsf{c}} = \infty$ if $\mathrm{p}^{\mathsf{c}} = 0$, and $\sigma^{\mathsf{c}} = \mathsf{s}\,\alpha^{\mathsf{c}}_1$ otherwise.

So, after Definition 14, $\Sigma^{\mathsf{c}}(\mathfrak{a}_1, \ldots, \mathfrak{a}_{\mathrm{q}^{\mathsf{c}}}) = \sup\{\mathfrak{a}_1 + 1, \ldots, \mathfrak{a}_{\mathrm{p}^{\mathsf{c}}} + 1\}$.

We now introduce a number of definitions to express the size of a constructor-headed term wrt. the sizes of its non-constructor-headed subterms.

Definition 27 A *constructor term* of sort B is either a variable or a term of the form $\mathsf{c}\vec{t}$ with $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and, for all $i \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$, either $T_i = \mathsf{B}$ and t_i a constructor term of sort B, or $T_i \neq \mathsf{B}$ and t_i is a variable.

The *height* $\mathrm{h}(l)$ of a constructor term l is defined as follows:

- $\mathrm{h}(x) = 0$,
- $\mathrm{h}(\mathsf{c}\,\vec{l}) = \sup\{\mathrm{h}(l_1) + 1, \ldots, \mathrm{h}(l_{\mathrm{p}^{\mathsf{c}}}) + 1\}$.

The (maximal) *depth* of a variable x in a constructor term l, $\mathrm{d}_x(l)$, is the element of the semi-ring $(\mathbb{N} \cup \{-\infty\}, \max, +)$ defined as follows:

- $\mathrm{d}_x(x) = 0$,
- $\mathrm{d}_x(y) = -\infty$ if $x \neq y$,
- $\mathrm{d}_x(\mathsf{c}\,\vec{l}) = \max(\{-\infty\} \cup \{\mathrm{d}_x(l_1) + 1, \ldots, \mathrm{d}_x(l_{\mathrm{p}^{\mathsf{c}}}) + 1\})$

Note that $\mathrm{d}_x(l) \leq \mathrm{h}(l)$. Moreover, $\mathrm{d}_x(l) = -\infty$ iff $x \notin \mathrm{FV}(l)$.

Lemma 29 If l is a constructor term of sort B and $l\theta \in \mathsf{B}$, then:
$o_{\mathcal{S}^{\mathsf{B}}}(l\theta) = \max(\{\mathrm{h}(l)\} \cup \{o_{[\mathsf{B}:\mathcal{S}^{\mathsf{B}}]T}(x\theta) + \mathrm{d}_x(l) \mid (x, U, \mathsf{B}) \trianglelefteq_{\mathrm{a}} (l, \mathsf{B}, \mathsf{B})\})$.

Proof. We proceed by induction on the size of l. If l is a variable, this is immediate. Assume now that $l = \mathsf{c}\,\vec{l}$ with $\mathsf{c} : \vec{T} \Rightarrow \mathsf{B}$. By Corollary 2, $o_{\mathcal{S}^{\mathsf{B}}}(l\theta) = \Sigma^{\mathsf{c}}(o_{[T_1]}(l_1\theta), \ldots, o_{[T_{\mathrm{p}^{\mathsf{c}}}]}(l_{\mathrm{p}^{\mathsf{c}}}\theta), \ldots)$ where $[T]$ denotes the stratification $[\mathsf{B} : \mathcal{S}^{\mathsf{B}}]T$. By Definition 14, $o_{\mathcal{S}^{\mathsf{B}}}(l\theta) = \sup\{o_{[T_1]}(l_1\theta) + 1, \ldots, o_{[T_{\mathrm{p}^{\mathsf{c}}}]}(l_{\mathrm{p}^{\mathsf{c}}}\theta) + 1\}$. Let $i \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$. If $T_i = \mathsf{B}$ then l_i is a constructor term of sort B. By induction hypothesis, $o_{[T_i]}(l_i\theta) = o_{\mathcal{S}^{\mathsf{B}}}(l_i\theta) = \max(\{\mathrm{h}(l_i)\} \cup \{o_{[U]}(x\theta) + \mathrm{d}_x(l_i) \mid (x, U, \mathsf{B}) \trianglelefteq_{\mathrm{a}} (l_i, \mathsf{B}, \mathsf{B})\})$. Otherwise, $T_i \neq \mathsf{B}$ and $l_i = x$. Thus, $o_{\mathcal{S}^{\mathsf{B}}}(l\theta) = \sup(\{\mathrm{h}(l_i) + 1 \mid i \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}\} \cup \{o_{[U]}(x\theta) + \mathrm{d}_x(l_i) + 1 \mid (x, U, \mathsf{B}) \trianglelefteq_{\mathrm{a}} (l_i, \mathsf{B}, \mathsf{B})\})$ $= \max(\{\mathrm{h}(l)\} \cup \{o_{[U]}(x\theta) + \mathrm{d}_x(l) \mid (x, U, \mathsf{B}) \trianglelefteq_{\mathrm{a}} (l, \mathsf{B}, \mathsf{B})\})$. ■

In the following, we assume to be under the conditions of Theorem 2 for some rule $\mathsf{f}\vec{l} \to r \in \mathcal{R}$, typing environment Γ and size substitution φ. In particular:

$$\overline{\Theta}(\mathsf{f}) = \mathrm{Annot}(T_1, \Sigma_1^{\mathsf{f}}, \alpha_1^{\mathsf{f}}) \Rightarrow \ldots \Rightarrow \mathrm{Annot}(T_{\mathrm{q}^{\mathsf{f}}}, \Sigma_{\mathrm{q}^{\mathsf{f}}}^{\mathsf{f}}, \alpha_{\mathrm{q}^{\mathsf{f}}}^{\mathsf{f}}) \Rightarrow T_{\mathrm{q}^{\mathsf{f}}+1} \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^{\mathsf{f}}} \Rightarrow \mathsf{B}_{\sigma^{\mathsf{f}}}$$

with $\vec{\alpha}^{\mathsf{f}}$ distinct variables, $\sigma^{\mathsf{f}} \in \mathtt{A} \cup \{\infty\}$ and $\mathrm{Var}(\sigma^{\mathsf{f}}) \subseteq \{\vec{\alpha}^{\mathsf{f}}\}$;

Since we are in the successor algebra, for all j, there are $n_j \in \mathbb{N}$ and $\gamma_j \in \mathtt{V}$ such that $\alpha_j^{\mathsf{f}}\varphi = \mathtt{s}^{n_j}\gamma_j$. Let $x_1, \ldots, x_n$ be an enumeration of $\mathrm{dom}(\Gamma)$. For all j, let $\mathfrak{l}_j$ be the biggest possible size for $x_j\theta$. Finally, for all j, let $\mathrm{Acc}(j) = \{x \in \mathbb{V} \mid \exists U, (x, U, \Sigma_j^{\mathsf{f}}) \trianglelefteq_{\mathrm{a}} (l_j, T_j, \Sigma_j^{\mathsf{f}})\}$ be the set of variables accessible in l_j whose size is measured wrt. Σ_j^{f}.

The minimality property is then equivalent to the following purely numerical problem on ordinals: for all $\mathfrak{a}_1, \ldots, \mathfrak{a}_n$ (for the sizes of $x_1\theta, \ldots, x_n\theta$ respectively) smaller than or equal to $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ respectively, there are $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$ (for $\alpha^{x_1}\nu, \ldots, \alpha^{x_n}\nu$ respectively) and $\mathfrak{c}_1, \ldots, \mathfrak{c}_{\mathrm{q}^{\mathsf{f}}}$ (for $\gamma_1\nu, \ldots, \gamma_{\mathrm{q}^{\mathsf{f}}}\nu$ respectively) such that:

1. $(\forall j)(\forall k)\ \mathfrak{b}_j = \mathfrak{b}_k$ if $\alpha^{x_j} = \alpha^{x_k}$,
2. $(\forall j)(\forall k)\ \mathfrak{c}_j = \mathfrak{c}_k$ if $\gamma_j = \gamma_k$,
3. $(\forall j)(\forall k)\ \mathfrak{b}_j = \mathfrak{c}_k$ if $\alpha^{x_j} = \gamma_k$,
4. $(\forall j)\ \mathfrak{c}_j + n_j = \max(\{\mathrm{h}(l_j)\} \cup \{\mathfrak{a}_m + \mathrm{d}_{x_m}(l_j) \mid x_m \in \mathrm{Acc}(j)\})$,
5. $(\forall j)\ \mathfrak{a}_j \leq \mathfrak{b}_j$.

The first three constraints are coherence conditions for ν to be well defined. The last two constraints correspond to the first and second conditions of the minimality property respectively.

For the 4th constraint to be satisfiable, one may think at first glance that it is sufficient to have $n_j \leq \mathrm{h}(l_j)$. While this is true with finite ordinals, this is not true with infinite ordinals. For instance, with $l_1 = \mathsf{b}\,x_1(\mathsf{c}\,x_2)$, $\mathfrak{a}_1 = \omega$, $\mathfrak{a}_2 = 0$ and $n_1 = 2$, there is no ordinal $\mathfrak{c}$ such that $\mathfrak{c} + 2 = \max\{\omega + 1, 2\}$. So, we need to have $n_j \leq \min(\{0\} \cup \{\mathrm{d}_x(l_j) \mid x \in \mathrm{Acc}(j)\})$.

Now, this condition is not sufficient for the other constraints to be satisfied. Take for instance $l_1 = \mathsf{c}\,x_1$, $l_2 = \mathsf{b}\,(\mathsf{c}\,x_1)\,(\mathsf{c}\,x_2)$, $\alpha^{x_1} = \alpha^{x_2} = \gamma_1 = \gamma_2$, $n_1 = 1$ and $n_2 = 2$. In this case, the minimality condition says that, for all $\mathfrak{a}_1, \mathfrak{a}_2$, there

is $\mathfrak{b}$ such that $\mathfrak{a}_1 \leq \mathfrak{b}$, $\mathfrak{a}_2 \leq \mathfrak{b}$, $\sup\{\mathfrak{a}_1+1\} = \mathfrak{b}+1$ and $\sup\{\mathfrak{a}_1+2, \mathfrak{a}_2+2\} = \mathfrak{b}+2$, which is not possible. The problem comes from the fact that l_1 and l_2 share a variable x_1 but the depth of x_1 in l_1 is distinct from the depth of x_1 in l_2. Hence, the following conditions:

Lemma 30 Assume that the type of constructors are annotated as described at the beginning of this section, and the type of function symbols as in Definition 16. For all i, let n_i be the unique integer and γ_i be the unique variable such that $\alpha^{\mathsf{f}}_i\varphi = \mathfrak{s}^{n_i}\gamma_i$. For all j, let $\mathrm{Acc}(j) = \{x \in \mathbb{V} \mid \exists T, (x, T, \Sigma^{\mathsf{f}}_j) \trianglelefteq_{\mathrm{a}} (l_j, T_j, \Sigma^{\mathsf{f}}_j)\}$ be the set of variables accessible in l_j whose size is measured wrt. Σ^{f}_j.

Under the assumptions of Theorem 2, the minimality property is satisfied if the following conditions are verified:

- $(\forall j)\ n_j \leq \min(\{0\} \cup \{\mathrm{d}_x(l_j) \mid x \in \mathrm{Acc}(j)\})$;
- $(\forall j)(\forall k)$ if $\gamma_j = \gamma_k$, then:
  - $n_j = n_k$,
  - $\mathrm{h}(l_j) = \mathrm{h}(l_k)$,
  - $\mathrm{Acc}(j) = \mathrm{Acc}(k)$,
  - $\forall x \in \mathrm{Acc}(j)$, $\mathrm{d}_x(l_j) = \mathrm{d}_x(l_k)$;
- $(\forall j)(\forall x)$ if $\gamma_j = \alpha^x$ then $x \in \mathrm{Acc}(j)$.

Proof. Let $\mathfrak{c}_i = \max(\{\mathrm{h}(l_i)\} \cup \{\mathfrak{a}_p + \mathrm{d}_{x_p}(l_i) \mid x_p \in \mathrm{Acc}(i)\}) - n_i$. It is well defined since $n_i \leq \min(\{0\} \cup \{\mathrm{d}_x(l_i) \mid x \in \mathrm{Acc}(i)\}) \leq \mathrm{h}(l_i)$. Now, let $\mathfrak{b}_i = \mathfrak{c}_m$ if $\alpha^{x_i} = \gamma_m$ for some m, and $\mathfrak{b}_i = \sup\{\mathfrak{a}_p \mid \alpha^{x_p} = \alpha^{x_i}\}$ otherwise. It is well-defined since, if $\gamma_j = \gamma_k$, then $\mathfrak{c}_j = \mathfrak{c}_k$. We now prove that the five numerical constraints equivalent to minimality are satisfied:

1. Assume that $\alpha^{x_j} = \alpha^{x_k}$. If $\alpha^{x_j} = \gamma_m$, then $\mathfrak{b}_j = \mathfrak{c}_m = \mathfrak{b}_k$. Otherwise, $\mathfrak{b}_j = \sup\{\mathfrak{a}_m \mid \alpha^{x_m} = \alpha^{x_j}\} = \mathfrak{b}_k$.
2. Assume that $\gamma_j = \gamma_k$. Then, $\mathfrak{c}_j = \mathfrak{c}_k$.
3. Assume that $\alpha^{x_j} = \gamma_k$. Then, $\mathfrak{b}_j = \mathfrak{c}_k$.
4. For all j, $\mathfrak{c}_j + n_j = \max(\{\mathrm{h}(l_j)\} \cup \{\mathfrak{a}_m + \mathrm{d}_{x_m}(l_j) \mid x_m \in \mathrm{Acc}(j)\})$ by definition.
5. For all j, $\mathfrak{a}_j \leq \mathfrak{b}_j$. Indeed, if $\alpha^{x_j} = \gamma_m$, then $\mathfrak{b}_j = \mathfrak{c}_m \geq \mathfrak{a}_j$ since $x_j \in \mathrm{Acc}(m)$. Otherwise, $\mathfrak{b}_j = \sup\{\mathfrak{a}_p \mid \alpha^{x_p} = \alpha^{x_j}\} \geq \mathfrak{a}_j$. ■

Now, one can easily check that all the examples of Section 4.3 that are annotated in the successor algebra satisfy the above conditions.

9 Conclusion

We have presented a general termination criterion for the combination of β-reduction and user-defined rewrite rules, based on the use of type-checking with size-annotated types approximating a semantic notion of size defined by the annotations given to constructor symbols. This extends to rewriting-based function definitions and more general notions of size, an approach initiated by Hughes, Pareto and Sabry for function definitions based on a fixpoint combinator and case analysis [HPS96].

First, we have shown that these termination conditions can be reduced to solving problems in the quasi-ordered algebra used for size annotations. Then, we have shown that the successor algebra (successor symbol with arbitrary constants) enjoys nice properties: decidability of the satisfiability of sets of inequalities (in polynomial time), and existence and computability of a most general solution for satisfiable problems (in polynomial time too). As a consequence, we have a complete algorithm for checking the termination conditions in the successor algebra.

We have implemented a simple heuristic that turns this termination criterion into a fully automated termination prover for higher-order rewriting called HOT [Bla12], which tries to detect size-preserving functions and, following [AA02], to find a lexicographic ordering on arguments. Combined with other (non-)termination techniques [JO91, Bla00, BJO02], HOT won the 2012 international competition of termination provers [Ter16] for higher-order rewriting against THOR [BR14] and WANDA [Kop15]. It could be improved by replacing the lexicographic ordering by the size-change principle [LJBA01, Hyv14], and using abstract interpretation techniques for annotating function symbols [TT00, CK01]. A more complete (and perhaps more efficient) implementation would be obtained by encoding constraints into a SAT problem and send it to state-of-art SAT solvers [FGM$^+$07, BAC08, CGSKT11].

A natural following is to study other size algebras like the max-successor algebra (*i.e.* the successor algebra extended with a `max` operator), the plus algebra (*i.e.* the successor algebra extended with addition) or their combination, the max-plus algebra. Indeed, the richer the size algebra is, the more precise the typing of function symbols is, and the more functions can be proved terminating.

Following [BR06], it is also possible to consider full Presburger arithmetic [Pre29] and handle conditional rewrite rules, by extending the system with explicit quantifiers and constraints on size variables, in the spirit of HM(X) [Sul01]. Simplification of constraints is then an important issue in practice [Pot01].

We have presented this criterion in Church' simply typed λ-terms but, following [Bla05b], it should be possible to extend it to richer type systems with polymorphic and dependent types. Similarly, we considered matching modulo α-congruence only but, following [Bla15], it should be possible to extend it to rewriting modulo some equational theory and to rewriting on β-normal forms with matching modulo $\beta\eta$ as used in Klop's combinatory reduction systems [KvOvR93] or Nipkow's higher-order rewrite systems [MN98].

Another interesting extension would be to consider size-annotated types in

the computability path ordering [BJR15], following Kamin and Lévy's extension of Dershowitz' recursive path ordering [Der79b, KL80], and Borralleras and Rubio's extension of Jouannaud and Okada's higher-order recursive path ordering [JR99, BR01].

References

[AA02] A. Abel and T. Altenkirch. A predicative analysis of structural recursion. *Journal of Functional Programming*, 12(1):1–41, 2002.

[Abe04] A. Abel. Termination checking with types. *Theoretical Informatics and Applications*, 38(4):277–319, 2004.

[Abe06] A. Abel. *A polymorphic lambda-calculus with sized higher-order types*. PhD thesis, Ludwig-Maximilians-Universität München, Germany, 2006.

[Abe08] A. Abel. Semi-continuous sized types and termination. *Logical Methods in Computer Science*, 4(2):1–33, 2008.

[Abe10] A. Abel. MiniAgda: integrating sized and dependent types. In *Proceedings of the Workshop on Partiality and Recursion in Interactive Theorem Provers*, Electronic Proceedings in Theoretical Computer Science 43, 2010.

[Abe12] A. Abel. Type-based termination, inflationary fixed-points, and mixed inductive-coinductive types. In *Proceedings of the 8th Workshop on Fixed-points in Computer Science*, Electronic Proceedings in Theoretical Computer Science 77, 2012.

[ABG14] M. Akian, R. Bapat, and S. Gaubert. Max-plus algebra. In L. Hogben, editor, *Handbook of Linear Algebra*, Discrete Mathematics and its Applications, chapter 35. CRC Press, 2nd edition, 2014.

[ACG97] R. Amadio and S. Coupet-Grimal. Analysis of a guard condition in type theory (preliminary report). Technical Report 3300, INRIA, France, 1997.

[ACG98] R. Amadio and S. Coupet-Grimal. Analysis of a guard condition in type theory (Extended abstract). In *Proceedings of the 1st International Conference on Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science 1378, 1998.

[Ack25] W. Ackermann. Begründung des "tertium non datur" mittels der Hilbertschen Theorie der Widerspruchsf *Mathematische Annalen*, 93:1–36, 1925.

[AG00] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.

[Agd16] Agda, a dependently typed functional programming language and proof assistant, 2016.

[AM10] M. Avanzini and G. Moser. Closing the gap between runtime complexity and polytime computability. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, Leibniz International Proceedings in Informatics 6, 2010.

[Art96] T. Arts. Termination by absence of infinite chains of dependency pairs. In *Proceedings of the 21st Colloquium on Trees in Algebra and Programming*, Lecture Notes in Computer Science 1059, 1996.

[ATS] The ATS programming language.

[BAC08] A. M. Ben-Amram and M. Codish. A SAT-based approach to size change termination with global ranking functions. In *Proceedings of the 14th International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 4963, 2008.

[BCOQ92] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: An Algebra for Discrete Event Systems.* Wiley, 1992.

[Ber05] U. Berger. Continuous semantics for strong normalization. In *Proceedings of the 1st Conference on Computability in Europe* , Lecture Notes in Computer Science 3526, 2005.

[Ber08] U. Berger. A domain model characterising strong normalisation. *Annals of Pure and Applied Logic*, 156(1):39–50, 2008.

[BFG97] F. Barbanera, M. Fernández, and H. Geuvers. Modularity of strong normalization in the algebraic-λ-cube. *Journal of Functional Programming*, 7(6):613–660, 1997.

[BFG+04] G. Barthe, M. J. Frade, E. Giménez, L. Pinto, and T. Uustalu. Type-based termination of recursive definitions. *Mathematical Structures in Computer Science*, 14(1):97–141, 2004.

[BGP05] G. Barthe, B. Grégoire, and F. Pastawski. Practical inference for type-based termination in a polymorphic setting. In *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 3461, 2005.

[BGP06] G. Barthe, B. Grégoire, and F. Pastawski. CIC$\widehat{\ }$: Type-Based Termination of Recursive Definitions in the Calculus of Inductive Constructions. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4246, 2006.

[BGR08] G. Barthe, B. Grégoire, and C. Riba. Type-based termination with sized products. In *Proceedings of the 22nd International Conference on Computer Science Logic*, Lecture Notes in Computer Science 5213, 2008.

[BJO02] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive-data-type systems. *Theoretical Computer Science*, 272:41–68, 2002.

[BJR15] F. Blanqui, J.-P. Jouannaud, and A. Rubio. The computability path ordering. *Logical Methods in Computer Science*, 11(4):1–45, 2015.

[Bla00] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Proceedings of the 11th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 1833, 2000.

[Bla04] F. Blanqui. A type-based termination criterion for dependently-typed higher-order rewrite systems. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 3091, 2004.

[Bla05a] F. Blanqui. Decidability of type-checking in the calculus of algebraic constructions with size annotations. In *Proceedings of the 19th International Conference on Computer Science Logic*, Lecture Notes in Computer Science 3634, 2005.

[Bla05b] F. Blanqui. Definitions by rewriting in the calculus of constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.

[Bla06a] F. Blanqui. Higher-order dependency pairs. In *8th International Workshop on Termination*, 2006.

[Bla06b] F. Blanqui. (HO)RPO revisited. Technical Report 5972, INRIA, France, 2006.

[Bla12] F. Blanqui. HOT, an automated termination prover for higher-order rewrite systems, 2012.

[Bla15] F. Blanqui. Termination of rewrite relations on λ-terms based on Girard's notion of reducibility. *Theoretical Computer Science*, 611:50–86, 2015.

[BM79] R. Boyer and J. Moore. *A computational logic*. Academic Press, 1979.

[BMP11] G. Bonfante, J.-Y. Marion, and R. Péchoux. Quasi-interpretations a way to control resources. *Theoretical Computer Science*, pages 2776–2796, 2011.

[Bou12] P. Boutillier. A relaxation of Coq's guard condition. In *Proceedings of the 23èmes Journées Francophones des Langages Applicatifs*, 2012.

[BQS80] R. Burstall, D. Mac Queen, and D. Sannella. HOPE: an experimental applicative language. In *Proceedings of the ACM Symposium on Lisp and Fonctional Programming*, 1980.

[BR01] C. Borralleras and A. Rubio. A monotonic higher-order semantic path ordering. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 2250, 2001.

[BR06] F. Blanqui and C. Riba. Combining typing and size constraints for checking the termination of higher-orde In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4246, 2006.

[BR09] F. Blanqui and C. Roux. On the relation between sized-types based termination and semantic labelling. In *Proceedings of the 23rd International Conference on Computer Science Logic*, Lecture Notes in Computer Science 5771, 2009.

[BR14] C. Borralleras and A. Rubio. THOR, a tool for proving the termination of higher-order rewrite systems, 2014.

[BTG89] V. Breazu-Tannen and J. Gallier. Polymorphic rewriting conserves algebraic strong normalization. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 372, 1989.

[CC79] P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.

[CF58] H. B. Curry and R. Feys. *Combinatory logic*. North-Holland, 1958.

[CG79] R. Cuninghame-Green. *Minimax algebra*. Number 166 in Lecture Notes in Economics and Mathematical Systems. SV, 1979.

[CG92] P.-L. Curien and G. Ghelli. Coherence of subsumption, minimum typing and type-checking in $F_{\leq}$. *Logical Methods in Computer Science*, 2(1):55–91, 1992.

[CGP10] P. Courtieu, G. Gbedo, and O. Pons. Improved matrix interpretation. In *Proceedings of the 36th International Conference on Current Trends in Theory and Practice of Computer Science*, Lecture Notes in Computer Science 5901, 2010.

[CGSKT11] M. Codish, J. Giesl, P. Schneider-Kamp, and R. Thiemann. SAT solving for termination proofs with recursive path orders and dependency pairs. *Journal of Automated Reasoning*, 49(1):53–93, 2011.

[Che03] J. Cheney. First-class phantom types. Technical Report TR2003-1901, Cornell University, 2003.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[CK01] W. N. Chin and S. C. Khoo. Calculating sized types. *Journal of Higher-Order and Symbolic Computation*, 14(2-3):261–300, 2001.

[CL87] A. Ben Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its in *Science of Computer Programming*, 9(2):137–159, 1987.

[CMTU05] E. Contejean, C. Marché, A. P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. *Journal of Automated Reasoning*, 34(4):325–363, 2005.

[Col75] G. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Proceedings of the 2nd GI conference on Automata Theory and Formal Languages*, number 33 in Lecture Notes in Computer Science, 1975.

[Coq92] T. Coquand. Pattern matching with dependent types. In *Proceedings of the International Workshop on Types for Proofs and Programs*, 1992.

[Cou96] P. Cousot. Abstract interpretation. *ACM Computing Surveys*, 28(2):324–328, 1996.

[Cou97] P. Cousot. Types as abstract interpretations (invited paper). In *Proceedings of the 24th ACM Symposium on Principles of Programming Languages*, 1997.

[CPM88] T. Coquand and C. Paulin-Mohring. Inductively defined types. In *Proceedings of the International Conference on Computer Logic*, Lecture Notes in Computer Science 417, 1988.

[CS07] T. Coquand and A. Spiwack. A proof of strong normalization using domain theory. *Logical Methods in Computer Science*, 3(4):1–16, 2007.

[CT96] E. A. Cichoń and H. Touzet. An ordinal calculus for proving termination in term rewriting. In *Proceedings of the 21st Colloquium on Trees in Algebra and Programming*, Lecture Notes in Computer Science 1059, 1996.

[dB70] N. G. de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In *Proceedings of the 1968 Symposium on Automatic Demonstration*, volume 125 of *Lecture Notes in Mathematics*, 1970.

[Der79a] N. Dershowitz. A note on simplification orderings. *Information Processing Letters*, 9(5):212–215, 1979.

[Der79b] N. Dershowitz. Orderings for term rewriting systems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, 1979.

[Der82] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[Der13] N. Dershowitz. Dependency Pairs are a Simple Semantic Path Ordering. In *13th International Workshop on Termination*, 2013.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B: formal models and methods*, chapter 6, pages 243–320. North-Holland, 1990.

[DM79] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[DO88] N. Dershowitz and M. Okada. Proof-theoretic techniques for term rewriting. In *Proceedings of the 3rd IEEE Symposium on Logic in Computer Science*, 1988.

[dV87] R. de Vrijer. Exactly estimating functionals and strong normalization. *Indagationes Mathematicae*, 90(4):479–493, 1987.

[EWZ08] J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2-3):195–220, 2008.

[FGM+07] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT solving for termination analysis with polynomial interpretations. In *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing*, Lecture Notes in Computer Science 4501, 2007.

[FK12] C. Fuhs and C. Kop. Polynomial interpretations for higher-order rewriting. In *Proceedings of the 23rd International Conference on Rewriting Techniques and Applications*, Leibniz International Proceedings in Informatics 15, 2012.

[FM90] Y.-C. Fuh and P. Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73(2):155–175, 1990.

[FNO+08] C. Fuhs, R. Navarro, C. Otto, J. Giesl, S. Lucas, and P. Schneider-Kamp. Search techniques for rational polynomial orders. In *Proceedings of the 9th International Conference on Artificial Intelligence and Symbolic Computation*, Lecture Notes in Computer Science 5144, 2008.

[FPT99] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, 1999.

[FR74] M. Fischer and M. Rabin. Super-exponential complexity of Presburger arithmetic. In *Proceedings of the SIAM-AMS Symposium in Applied Mathematics*, 1974.

[Gan80a] R. O. Gandy. An early proof of normalization by A. M. Turing. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 453–455. Academic Press, 1980.

[Gan80b] R. O. Gandy. Proofs of strong normalization. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 457–477. Academic Press, 1980.

[Gen35] G. Gentzen. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112(1):493–565, 1935. English translation in [Sza69].

[Gie97] J. Giesl. Termination of nested and mutually recursive algorithms. *Journal of Automated Reasoning*, 19(1):1–29, 1997.

[Gim94] E. Giménez. Codifying guarded definitions with recursion schemes. In *Proceedings of the International Workshop on Types for Proofs and Programs*, Lecture Notes in Computer Science 996, 1994.

[Gim96] E. Giménez. *Un calcul de constructions infinies et son application à la vérification de systèmes communiq* PhD thesis, ENS Lyon, France, 1996.

[Gim98] E. Giménez. Structural recursive definitions in type theory. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 1443, 1998.

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris 7, France, 1972.

[GLT88] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and types*. Cambridge University Press, 1988.

[Göd31] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931. English translation in [vH77].

[Göd58] K. Gödel. Über einer bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(3-4):280–287, 1958. Reprinted in [Göd90].

[Göd90] K. Gödel. *Collected works - vol. 2: publications 1938-1974*. Oxford University Press, 1990.

[GS10] B. Grégoire and J. L. Sacchini. On strong normalization of the calculus of constructions with type-based te In *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 6397, 2010.

[GTSKF06] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[Ham06] M. Hamana. An initial algebra approach to term rewriting systems with variable binders. *Journal of Higher-Order and Symbolic Computation*, 19(2-3):231–262, 2006.

[Ham07] M. Hamana. Higher-order semantic labelling for inductive datatype systems. In *Proceedings of the 9th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, 2007.

[Har04] G. H. Hardy. A theorem concerning the infinite cardinal numbers. *Quarterly Journal of Mathematics*, 35:87–94, 1904.

[Har15] F. Hartogs. Über das Problem der Wohlordnung. *Mathematische Annalen*, 76:438–443, 1915.

[Her30] J. Herbrand. *Recherches sur la théorie de la démonstration*. PhD thesis, Faculté des sciences de Paris, France, 1930.

[Hes09] G. Hessenberg. Kettentheorie und Wohlordnung. *Journal für die reine und angewandte Mathematik*, 135:81–133, 1909.

[HH82] G. Huet and J.-M. Hullot. Proofs by induction in equational theories with constructors. *Journal of Computer and System Sciences*, 25(2):239–266, 1982.

[Hin69] R. Hindley. The principal type-scheme of an object in combinatory logic. *Transactions of the American Mathematical Society*, 146:29–60, 1969.

[HJ98] H. Hong and D. Jakuš. Testing positiveness of polynomials. *Journal of Automated Reasoning*, 21(1):23–38, 1998.

[HJ99] K. Hrbacek and T. Jech. *Introduction to set theory*. M. Dekker, 3rd, revised and expanded edition, 1999.

[HL89] D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proceedings of the 3rd International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 355, 1989.

[HM05] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1-2):172–199, 2005.

[HM06] N. Hirokawa and A. Middeldorp. Predictive labeling. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 4098, 2006.

[Hof95] M. Hofmann. Approaches to recursive data types - a case study. Unpublished note cited in [Mat00] p. 61, 1995.

[How70] W. A. Howard. Assignment of ordinals to terms for primitive recursive functionals of finite type. In *Intuitionism and Proof Theory: Proceedings of the Summer Conference at Buffalo N. Y.*, volume 60 of *Studies in Logic and the Foundations of Mathematics*, 1970.

[How72] W. A. Howard. A system of abstract constructive ordinals. *Journal of Symbolic Logic*, 37(2):355–374, 1972.

[HPS96] J. Hughes, L. Pareto, and A. Sabry. Proving the correctness of reactive systems using sized types. In *Proceedings of the 23th ACM Symposium on Principles of Programming Languages*, 1996.

[Hue76] G. Huet. Résolution d'équations dans les langages d'ordre 1, 2, ..., ω, 1976. Thèse d'État, Université Paris 7, France.

[Hyv14] P. Hyvernat. The Size-Change Termination Principle for Constructor Based Languages. *Logical Methods in Computer Science*, 10(1):1–30, 2014.

[JB08] N. D. Jones and N. Bohr. Call-by-value termination in the untyped λ-calculus. *Logical Methods in Computer Science*, 4(1):1–39, 2008.

[JO91] J.-P. Jouannaud and M. Okada. A computation model for executable higher-order algebraic specification la In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, 1991.

[JR99] J.-P. Jouannaud and A. Rubio. The higher-order recursive path ordering. In *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*, 1999.

[JR07] J.-P. Jouannaud and A. Rubio. Polymorphic higher-order recursive path orderings. *Journal of the ACM*, 54(1):1–48, 2007.

[Kah95] S. Kahrs. Towards a domain theory for termination proofs. In *Proceedings of the 6th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 914, 1995.

[KB70] D. Knuth and P. Bendix. Simple word problems in universal algebra. In *Computational problems in abstract algebra, Proceedings of a Conference held at Oxford in 1967*, pages 263–297. Pergamon Press, 1970. Reproduced in [SW83].

[KISB09] K. Kusakari, Y. Isogai, M. Sakai, and F. Blanqui. Static dependency pair method based on strong computability for higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E92-D(10):2007–2015, 2009.

[KL80] S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. Unpublished note, 1980.

[Kop11] C. Kop. Higher order dependency pairs for algebraic functional systems. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications*, Leibniz International Proceedings in Informatics 10, 2011.

[Kop15] C. Kop. WANDA, a higher-order termination tool, 2015.

[KS07] K. Kusakari and M. Sakai. Enhancing dependency pair method using strong computability in simply-typed *Applicable Algebra in Engineering Communication and Computing*, 18(5):407–431, 2007.

[KT28] B. Knaster and A. Tarski. Un théorème sur les fonctions d'ensembles. *Annales de la Société Polonaise de Mathématiques*, 6:133–134, 1928.

[Kur22] C. Kuratowski. Une méthode d'élimination des nombres transfinis des raisonnements mathématiques. *Fundamenta Mathematicae*, 3(1):76–108, 1922.

[KvOvR93] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[KZ06] A. Koprowski and H. Zantema. Automation of recursive path ordering for infinite labelled rewrite systems. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning*, Lecture Notes in Computer Science 4130, 2006.

[LJBA01] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In *Proceedings of the 28th ACM Symposium on Principles of Programming Languages*, 2001.

[Luc05] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *Theoretical Informatics and Applications*, 39:547–586, 2005.

[Mat70] Y. V. Matiyasevich. Enumerable sets are diophantine. *Soviet Mathematics. Doklady*, 11:354–358, 1970.

[Mat93] Y. V. Matiyasevich. *Hilbert's tenth problem*. MIT Press, 1993.

[Mat00] R. Matthes. *Lambda calculus: a case for inductive definitions*, 2000.

[Men87] N. P. Mendler. *Inductive definition in type theory*. PhD thesis, Cornell University, USA, 1987.

[Men91] N. P. Mendler. Inductive types and type constraints in the second-order lambda calculus. *Annals of Pure and Applied Logic*, 51(1-2):159–172, 1991.

[Mil78] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(3):348–375, 1978.

[Mil91] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unificat In *Proceedings of the International Workshop on Extensions of Logic Programming*, Lecture Notes in Computer Science 475, 1991.

[Min14] MiniAgda, 2014.

[Mit84] J. Mitchell. Coercion and type inference (summary). In *Proceedings of the 11th ACM Symposium on Principles of Programming Languages*, 1984.

[ML75] P. Martin-Löf. An intuitionistic theory of types: predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Proceedings of the 1973 Logic Colloquium*, volume 80 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1975.

[MN70] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Proceedings of the 3rd Hawaii International Conference on System Sciences*, 1970.

[MN98] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(2):3–29, 1998.

[Mos14] G. Moser. KBOs, ordinals, subrecursive hierarchies and all that. *Journal of Logic and Computation*, pages 1–27, 2014. Published online on December 3.

[MOZ96] A. Middeldorp, H. Ohsaki, and H. Zantema. Transforming termination by self-labelling. In *Proceedings of the 13th International Conference on Automated Deduction*, Lecture Notes in Computer Science 1104, 1996.

[MS01] F. Monin and M. Simonot. An ordinal measure based procedure for termination of functions. *Theoretical Computer Science*, 254(1-2):63–94, 2001.

[New42] M. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*, 1991.

[Oka89] M. Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, 1989.

[Par00] L. Pareto. *Types for crash prevention*. PhD thesis, Chalmers University of Technology, Göteborg, Sweden, 2000.

[Pau86] L. Paulson. Proving termination of normalization functions for conditional expressions. *Journal of Automated Reasoning*, 2(1):63–74, 1986.

[Pea89] G. Peano. *Arithmetices principia, nova methodo exposita*. Fratres Bocca, 1889. Partial English translation in [vH77].

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5(3):223–255, 1977.

[Pot01] F. Pottier. Simplifying subtyping constraints: a theory. *Information and Computation*, 170(2):153–183, 2001.

[Pra77] V. Pratt. Two easy theories whose combination is hard. Unpublished note, 1977.

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Sprawozdanie z I Kongresu Matematykow Krajow Slowcanskich, Warszawa, Poland*, 1929.

[Rat06] M. Rathjen. The art of ordinal analysis. In *Proceedings of the International Congress of Mathematicians*, volume 2, pages 45–69, 2006.

[Rib07] C. Riba. On the stability by union of reducibility candidates. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science 4423, 2007.

[Rib08] C. Riba. Stability by union of reducibility candidates for orthogonal constructor rewriting. In *Proceedings of the 4th Conference on Computability in Europe* , Lecture Notes in Computer Science 5028, 2008.

[Rib09] C. Riba. On the values of reducibility candidates. In *Proceedings of the 9th International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 5608, 2009.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[RR63] H. Rubin and J. E. Rubin. *Equivalents of the axiom of choice*. North-Holland, 1963.

[Sac11] J. L. Sacchini. *On Type-Based Termination and Dependent Pattern Matching in the Calculus of Inductive* PhD thesis, ParisTech, France, 2011.

[Sac15] J. L. Sacchini. Cicminus, Coq with type-based termination, 2015.

[Sch14] S. Schmitz. Complexity Bounds for Ordinal-Based Termination (invited talk). In *Proceedings of the 8th International Workshop on Reachability Problems*, volume 8762 of *Lecture Notes in Computer Science*, pages 1–19, 2014.

[Sco72] D. S. Scott. Continuous lattices. In E. Lawvere, editor, *Toposes, algebraic geometry and logic*, number 274 in Lecture Notes in Mathematics, pages 97–136. Springer, 1972.

[SD03] C. Sprenger and M. Dam. On the structure of inductive reasoning: circular and tree-shaped proofs in the μ In *Proceedings of the 6th International Conference on Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science 2620, 2003.

[Sel93] M. P. A. Sellink. Verifying process algebra proofs in type theory. In *Proceedings of the 1st International Workshop on Semantics of Specification Languages*, 1993.

[SM08] C. Sternagel and A. Middeldorp. Root labeling. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 5117, 2008. This paper contains errors described in [ST10].

[ST10] C. Sternagel and R. Thiemann. Signature extensions preserve termination - an alternative proof via depen In *Proceedings of the 24th International Conference on Computer Science Logic*, Lecture Notes in Computer Science 6247, 2010.

[Sul00] M. Sulzmann. *A general framework for Hindley/Milner type systems with constraints*. PhD thesis, Yale University, USA, 2000.

[Sul01] M. Sulzmann. A general type inference framework for Hindley/Milner style systems. In *Proceedings of the 5th Fuji International Symposium on Functional and Logic Programming*, Lecture Notes in Computer Science 2024, 2001.

[SW83] J. H. Siekmann and G. Wrightson, editors. *Automation of reasoning. 2: classical papers on computational logic 1967-1970*. Symbolic computation. Springer, 1983.

[SWS01] M. Sakai, Y. Watanabe, and T. Sakabe. An extension of dependency pair method for proving termination of higher-order rewrite systems. *IEICE Transactions on Information and Systems*, E84-D(8):1025–1032, 2001.

[Sza69] M. E. Szabo, editor. *Collected papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland, 1969.

[Tai67] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967.

[Tar48] A. Tarski. A decision method for elementary algebra and geometry. Technical Report R-109, RAND Corporation, USA, 1948.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[TeR03] TeReSe. *Term rewriting systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[Ter16] Termination competition website, 2016.

[TG05] R. Thiemann and J. Giesl. The size-change principle and dependency pairs for termination of term rewriti
Applicable Algebra in Engineering Communication and Computing, 16(4):229–270, 2005.

[TT00] A. Telford and D. Turner. Ensuring termination in ESFP. In *Proceedings of the 15th British Colloquium for Theoretical Computer Science*, Journal of Universal Computer Science 6(4), 2000.

[Tur42] A. M. Turing. Some theorems about Church's system. Unpublished typescript reproduced in [Gan80a], 1942.

[vdP93] J. van de Pol. Termination proofs for higher-order rewrite systems. In *Proceedings of the 1st International Workshop on Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science 816, 1993.

[vdP95] J. van de Pol. Two different strong normalization proofs? Computability versus functionals of finite type. In *Proceedings of the 2nd International Workshop on Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science 1074, 1995.

[vdP96] J. van de Pol. *Termination of higher-order rewrite systems*. PhD thesis, Utrecht Universiteit, NL, 1996.

[vH77] J. v. Heijenoort, editor. *From Frege to Gödel, a source book in mathematical logic, 1879-1931*. Harvard University Press, 1977.

[vO94] V. van Oostrom. *Confluence for abstract and higher-order rewriting*. PhD thesis, Vrije Universiteit Amsterdam, NL, 1994.

[Wah07] D. Wahlstedt. *Dependent type theory with first-order parameterized data types and well-founded recursion.* PhD thesis, Chalmers University of Technology, Sweden, 2007.

[Wal88] C. Walther. Argument-bounded algorithms as a basis for automated termination proofs. In *Proceedings of the 9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310, 1988.

[Wei98] A. Weiermann. How is It That Infinitary Methods can be Applied to Finitary Mathematics? Gödel's T: A *Journal of Symbolic Logic*, 63(4):1348–1370, 1998.

[Wer94] B. Werner. *Une théorie des constructions inductives.* PhD thesis, Université Paris 7, France, 1994.

[WW12] G. Wilken and A. Weiermann. Derivation Lengths Classification of Gödel's T Extending Howard's Assignn *Logical Methods in Computer Science*, 8(1):1–44, 2012.

[XCC03] H. Xi, C. Chen, and G. Chen. Guarded recursive datatype constructors. In *Proceedings of the 30th ACM Symposium on Principles of Programming Languages*, 2003.

[Xi02] H. Xi. Dependent types for program termination verification. *Journal of Higher-Order and Symbolic Computation*, 15(1):91–131, 2002.

[Xi03] H. Xi. Applied Type System (extended abstract). In *Proceedings of the International Workshop on Types for Proofs and Programs*, Lecture Notes in Computer Science 3085, 2003.

[Zan95] H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.

[Zen97] C. Zenger. Indexed types. *Theoretical Computer Science*, 187(1-2):147–165, 1997.