Size-based termination of higher-order rewriting

Frédéric Blanqui

Under consideration for publication in J. Functional Programming

Size-based termination of higher-order rewriting

Frédéric Blanqui
INRIA

ENS / Université Paris-Saclay
LSV, 61 avenue du Président Wilson, 94235 Cachan Cedex, France

Abstract

We provide a general and modular criterion for the termination of simply-typed λ-calculus extended with function symbols defined by user-defined rewrite rules. Following a work of Hughes, Pareto and Sabry for functions defined with a fixpoint operator and pattern-matching, several criteria use typing rules for bounding the height of arguments in function calls. In this paper, we extend this approach to rewriting-based function definitions and more general user-defined notions of size.

1 Introduction

In this paper, we are interested in the termination of Church's simply-typed λ-calculus (Church, 1940) extended with function symbols defined by user-defined rewrite rules (Dershowitz & Jouannaud, 1990; TeReSe, 2003) like the ones of Figure 1. Our results could be used to check the termination of typed functional programs (*e.g.* in OCaml (OCaml, 2017) or Haskell (Haskell, 2017)), rewriting-based programs (*e.g.* in Maude (Maude, 2015)), or function definitions in proof assistants (*e.g.* Coq (Coq, 2017), Agda (Agda, 2017), Dedukti (Dedukti, 2018)). By termination, we mean the strong normalization property, that is, the absence of infinite rewrite sequences $t_0 \to t_1 \to \ldots$ The mere existence of a normal form is a weaker property called weak normalization. Termination is an important property in program verification.

The rewrite system of Figure 1 defines 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. A way to prove the termination of this system is to show that, in two successive functions calls, arguments are strictly decreasing wrt some well-founded order. A natural order, based on the inductive nature of N, is to compare the height of terms. More precisely, let the size of a terminating term t of sort N be the number of s symbols at the top of the normal form of t (this rewrite system is weakly orthogonal and thus confluent (van Oostrom, 1994)). While the termination of sub (*i.e.* the absence of infinite reductions starting from a term of the form $\mathsf{sub}\ t\ u$ 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 the observation that sub is not size-increasing, that is, the size of $(\mathsf{sub}\ t\ u)$ is less than or equal to the size of t.

The idea of sized types, introduced by Hughes, Pareto and Sabry in (Hughes *et al.*, 1996) for fixpoint-based function definitions, is to consider an abstract interpretation of this

Fig. 1. Rewrite system defining substraction and division on natural numbers

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

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 one 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 every recursive call. Hence, termination is reduced to checking typing and abstract size decreasingness.

In our example, this amounts to saying: 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 $\mathsf{div}\ (\mathsf{s}\ t)\ (\mathsf{s}\ u)$ terminates because its first argument is of size $\alpha+1$ while, in the recursive call $\mathsf{div}\ (\mathsf{sub}\ t\ u)\ (\mathsf{s}\ u)$, 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 (Sellink, 1993), Howard's constructive ordinals in Example 5), the size of a term is generally not a finite natural number but a transfinite ordinal number. However, 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 $\to$ 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 $t \to u$.

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 (Gödel, 1931), defining $\|\ \|$ and proving that $\|t\| >_{\mathbb{D}} \|u\|$ whenever $t \to u$, 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 $\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ but PA cannot prove the well-foundedness of ε_0 itself (Gentzen, 1935). Yet, there is a function $\|\ \|$ from the terms of Gödel's system T (Gödel, 1958) (which extends PA) to ω such that $\|t\| >_{\mathbb{D}} \|u\|$ whenever $t \to u$ (Weiermann, 1998).

An equivalent approach is finding a well-founded relation containing $\rightarrow$. For instance, Dershowitz's recursive path ordering (RPO) (Dershowitz, 1979b; Dershowitz, 1982) or its extension to the higher-order case by Jouannaud and Rubio (Jouannaud & Rubio, 1999; Jouannaud & Rubio, 2007; Blanqui *et al.*, 2015). But, in this paper, we will focus on the explicit use of size functions. For a connection between RPO and ordinals, see for instance (Dershowitz & Okada, 1988).

Early examples of this approach are given by Ackermann's proof of termination of second-order primitive recursive arithmetic functions using $\mathfrak{h} = \omega^{\omega^\omega}$ (Ackermann, 1925), Gentzen's proof of termination of cut elimination in Peano arithmetic using $\mathfrak{h} = \varepsilon_0$ (Gentzen, 1935; Howard, 1970; Wilken & Weiermann, 2012), Turing's proof of weak normalization of Church's simply-typed λ-calculus (Turing, 1942), and Howard's proof of termination of his system V (an extension of Gödel's system T with an inductive type for representing ordinals) using Bachmann's ordinal (Howard, 1972). 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 (Rathjen, 2006) for some recent survey. Instead, Monin and Simonot developed an algorithm for trying to find size assignments in $\mathfrak{h} = \omega^\omega$ (Monin & Simonot, 2001).

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 (Girard, 1972; Girard *et al.*, 1988), 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 (Arts, 1996; Arts & Giesl, 2000; Hirokawa & Middeldorp, 2005; Giesl *et al.*, 2006), and size-change principle for first-order functional programs (Lee *et al.*, 2001). These two notions are thoroughly compared in (Thiemann & Giesl, 2005). 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 (Sakai *et al.*, 2001; Wahlstedt, 2007; Jones & Bohr, 2008; Kusakari *et al.*, 2009; Kop, 2011), 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 (Hofbauer & Lautemann, 1989), 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 (Moser, 2014), unless perhaps one bounds the growth rate of the size of terms (measured here as the number of symbols) (Schmitz, 2014). 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 whose subterms are in normal form and whose size is smaller than n, seems to provide a better (Turing related) complexity model (Avanzini & Moser, 2010).

1.2 Model-based termination

In (Manna & Ness, 1970), Manna and Ness proposed to interpret every term whose free variables are $x_1,\dots,x_n$ 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 $x_1,\dots,x_n\in\mathbb{E}$, $f(x_1,\dots,x_n)<_{\mathbb{E}} g(x_1,\dots,x_n)$.

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,

$$f(x_1,\dots,x_p)>g(x_1,\dots,x_q) \text{ is equivalent to } f(x_1,\dots,x_p)-g(x_1,\dots,x_q)-1\geq 0$$

and monotony is equivalent to checking that, for all i, $f(\dots,x_i+1,\dots)-f(\dots,x_i,\dots)-1\geq 0$. 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 (Cherifa & Lescanne, 1987; Lucas, 2005; Contejean *et al.*, 2005; Fuhs *et al.*, 2007). Unfortunately, polynomial absolute positivity is undecidable on $\mathbb{N}$ since it is equivalent to the solvability of Diophantine equations (Proposition 6.2.11 in (TeReSe, 2003)), which is undecidable (Matiyasevich, 1970; Matiyasevich, 1993). 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.* $\mathsf{f}_{\mathbb{E}}(x_1,\dots,x_n)>x_i$ for all i) (Dershowitz, 1979a), or with the well-founded relation $<_\delta$ where, for some fixed $\delta>0$, $x<_\delta y$ if $x+\delta\leq y$ (Lucas, 2005; Fuhs *et al.*, 2008). In the case of $\mathbb{R}^+$, polynomial absolute positivity is decidable but of exponential complexity (Tarski, 1948; Collins, 1975). Useful heuristics have however been studied (Hong & Jakuš, 1998).

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) (Endrullis *et al.*, 2008; Courtieu *et al.*, 2010), where $\mathbb{B}$ is a well-founded domain.

Instead of polynomial functions, Cichoń considered the class of Hardy functions (Hardy, 1904) indexed by ordinals smaller than ε_0 (Cichoń & Touzet, 1996). 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 (Gandy, 1980b), 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 (de Vrijer, 1987).

Gandy's approach was later extended by van de Pol (van de Pol, 1993; van de Pol, 1996) and Kahrs (Kahrs, 1995) to arbitrary higher-order rewriting *à la* Nipkow (Nipkow, 1991; Mayr & Nipkow, 1998), that is, to rewriting on terms in β-normal η-long form with higher-order pattern-matching (Miller, 1991). But this approach has been implemented only recently (Fuhs & Kop, 2012).

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 (van de Pol, 1995; van de Pol, 1996).

Finally, a general categorical framework has been developed by Hamana (Hamana, 2006), that is complete wrt. the termination of binding term rewrite systems, a formalism based on Fiore, Plotkin and Turi's binding algebra (Fiore *et al.*, 1999) and close to a typed version of Klop's combinatory reduction systems (Klop *et al.*, 1993).

To the best of our knowledge, nobody seems to have studied the relations between Howard's approach based on ordinals (Howard, 1970; Wilken & Weiermann, 2012) and Gandy's approach based on interpretations (Gandy, 1980b; de Vrijer, 1987; van de Pol, 1996).

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 (Bonfante *et al.*, 2011) but, sometimes, may also simplify the search of a termination proof. Indeed, Zantema proved in (Zantema, 1995) that the termination of a first-order rewrite system $\mathscr{R}$ is equivalent to the termination of $\mathrm{lab}(\mathscr{R}) \cup >_{\mathbb{D}}$, where $\mathrm{lab}(\mathscr{R})$ are all the variants of $\mathscr{R}$ 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 (Middeldorp *et al.*, 1996; Koprowski & Zantema, 2006; Sternagel & Middeldorp, 2008). This result was later extended to the higher-order case by Hamana (Hamana, 2007).

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 (Burstall *et al.*, 1980).

The semantics of an inductive type B, $[\![\mathsf{B}]\!]$, is usually defined, following Hessenberg's theorem (Hessenberg, 1909), Knaster and Tarski's theorem (Knaster & Tarski, 1928) or Tarski's theorem (Tarski, 1955), as the smallest fixpoint of a monotone function $\mathbb{H}^{\mathsf{B}}$ on some complete lattice. Moreover, following Kuratowski (Kuratowski, 1922; Cousot & Cousot, 1979), 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 $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$, where $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$ 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's system T (Gödel, 1958) and Howard's system V (Howard, 1972) to functionals defined by

recursion on higher-order inductive types, *i.e.* types with constructors taking functions as arguments (Mendler, 1987; Mendler, 1991), in which case the size of a term can be bigger than ω.

In (Hughes *et al.*, 1996; Pareto, 2000), 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 $\mathsf{B}_0, \mathsf{B}_1, \ldots \mathsf{B}_\infty = \mathsf{B}$ for typing the terms of type B of size smaller than or equal to $0, 1, \ldots, \infty$ 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 $\mathtt{0}$ for zero, $\mathtt{s}$ for successor and $+$ for addition, and arbitrary size variables $\alpha, \beta, \ldots$, that is the language of Presburger arithmetic (Presburger, 1929). So, for instance, the usual list constructor cons gets the type $\mathsf{N} \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{L}_{\mathtt{s}\alpha}$, 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 (Milner, 1978).

Hughes, Pareto and Sabry do not actually prove the termination of their calculus but provide a domain-theoretic model (Scott, 1972). However, following Plotkin (Plotkin, 1977), 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 (Amadio & Coupet-Grimal, 1997; Amadio & Coupet-Grimal, 1998), who independently developed a system similar to the one of Hughes, Pareto and Sabry, inspired by Giménez's work on the use of typing annotations for termination and productivity (Giménez, 1996). Giménez himself later proposed a similar system in (Giménez, 1998) 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 (Berger, 2005; Coquand & Spiwack, 2007; Berger, 2008).

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 (Walther, 1988; Giesl, 1997). Following (Cousot, 1997), it could certainly be described as an actual 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 (Hindley, 1969; Milner, 1978) where the set of types of a term has a smallest element wrt. the instantiation ordering if it is not empty (Huet, 1976). In this case, there is a complete type-checking algorithm for (t, T) which consists of checking that T is an instance of the smallest type of t (Hindley, 1969). 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 $(\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow (\mathsf{B} \Rightarrow \mathsf{C})$) (Fuh & Mishra, 1990). 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 (Mitchell, 1984; Fuh & Mishra, 1990; Pottier, 2001), indexed types (Zenger, 1997), DML(C) (Xi, 2002), HM(X) (Sulzmann, 2000), or generalized algebraic data types (GADTs) (Xi *et al.*, 2003; Cheney, 2003), which are all a restricted form of dependent types (de Bruijn, 1970; Martin-Löf, 1975).

Hughes, Pareto and Sabry's approach was later extended to higher-order data types (Barthe *et al.*, 2004), polymorphic types (Abel, 2004; Barthe *et al.*, 2005; Abel, 2006; Abel, 2008), rewriting-based function definitions in the calculus of constructions (Blanqui, 2004; Blanqui, 2005a), conditional rewriting (Blanqui & Riba, 2006), product types (Barthe *et al.*, 2008), and fixpoint-based function definitions in the calculus of constructions (Barthe *et al.*, 2006; Grégoire & Sacchini, 2010; Sacchini, 2011).

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 to differentiate between their size. Hence, transfinite computations can be reduced to finite ones.

Finally, Roux and the author proved in (Blanqui & Roux, 2009) 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 (Hirokawa & Middeldorp, 2006). Interestingly, semantic labeling allows one to deal with function definitions using matching on defined symbols, like in a rule 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 (Xi, 2003; ATS, 2018), MiniAgda (Abel, 2010; MiniAgda, 2014), Agda (Agda, 2017), cicminus (Sacchini, 2011; cicminus, 2015) or HOT (HOT, 2012). 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 (Telford & Turner, 2000; Chin & Khoo, 2001). They are both based on abstract interpretation techniques (Cousot, 1996).

1.4 Contributions

1. 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 (Blanqui, 2004; Blanqui, 2005a), hence providing the first complete account of the extension of Hughes, Pareto and Sabry's approach to rewriting-based function definitions (Dershowitz & Jouannaud, 1990; TeReSe, 2003).
2. 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 (Presburger, 1929; Fischer & Rabin, 1974). In those considering higher-order data types, the successor symbol $\mathtt{s}$ is usually the only symbol allowed, except in (Barthe *et al.*, 2008) 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 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.
3. In all the previous works, the notion of size is also fixed: the size of t is the height of the set-theoretical tree representation of the normal form of t (an abstraction being represented as an infinite set of trees).
   The third contribution of the paper is to enable users to define their own notion of size by annotating the types of constructors. These annotations generate a stratification of the interpretation of inductive types. We prove that one can build such a stratification in the domain of Girard's computability predicates (Girard, 1972; Girard *et al.*, 1988) when annotations form monotone and extensive functions.
4. 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 computable in polynomial time too.

In contrast with (Blanqui, 2004; Blanqui, 2005a), 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 recall the definitions of types, terms and rewriting, and the interpretation of types as computability predicates. In Section 3, we introduce the notions of stratification,

size and constructor size functions, and prove properties on the size of computable terms. In Section 4, we present the termination criterion. The main ingredient of the termination criterion is a type system with subtyping, parametrized by a quasi-ordered first-order term algebra for abstract size expressions. It also requires that annotations of arguments are minimal in some sense. In Section 5, we provide a sufficient syntactic condition for the minimality property to be satisfied when the size is defined as the height. In Section 6, we provide various examples of the expressive power of our termination criterion. In Section 7, we provide a complete algorithm for checking subject-reduction and size-decreasingness under some general assumptions on the size algebra. In Section 8, we show how subtyping problems can be reduced to ordering problems in the size algebra. Finally, in Section 9, we prove that the simplest possible algebra, the successor algebra, satisfies the required conditions for the type-checking algorithm to be complete.

2 Types, terms and computability

In this section, we define the set of terms that we consider (Church's simply-typed λ-calculus with constants (Church, 1940)), the operational semantics (the combination of β-reduction and user-defined rewrite rules (Dershowitz & Jouannaud, 1990; TeReSe, 2003)), 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 $f(x)$.

We recall that, if X is a bounded set of ordinals, *i.e.* when there is b such that $x \leq b$ for all $x \in X$, then the least upper bound of X, written $\sup X$, exists. In particular, $\sup \emptyset = 0$.

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;

Fig. 2. Typing rules

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

- 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 $(\dots((tu_1)u_2)\dots u_{n-1})u_n$ 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 (Curry & Feys, 1958).

As usual, positions in a tree (type or term) are denoted by words on positive integers. Word concatenation is denoted by juxtaposition and the empty word by ε. Given a tree t and a position p in t, let $t|_p$ be the subtree of t at position p, and $\mathrm{Pos}(u,t)$ be the set of positions p in t such that $t|_p=u$.

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 is 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 recalled 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 $T_1,\dots,T_{\mathrm{r}^s}$ and a sort B such that $\Theta(s)=T_1\Rightarrow\dots\Rightarrow T_{\mathrm{r}^s}\Rightarrow\mathsf{B}$.

Given $\mathsf{B}\in\mathbb{S}$, let $\mathbb{C}^{\mathsf{B}}=\{(\mathsf{c},\vec{t},\vec{T})\mid\mathsf{c}\in\mathbb{C},\mathsf{c}:\vec{T}\Rightarrow\mathsf{B},|\vec{t}|=|\vec{T}|\}$ be the set of tuples $(\mathsf{c},\vec{t},\vec{T})$ such that c is maximally applied in $\mathsf{c}\vec{t}$ and $\vec{T}$ are the types declared for the arguments of c (but t_i does not need to be of type T_i).

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* $\to_\beta$ is the smallest monotone relation containing all the pairs $((\lambda x^U t)u, t\{(x,u)\})$.

A *rewrite rule* is a pair of terms (l,r), written $l \to r$, such that there are $\mathsf{f} \in \mathbb{F}$, $\vec{l}$, Δ and T such that $l = \mathsf{f}\vec{l}$, $\mathrm{FV}(r) \subseteq \mathrm{FV}(l)$, $\Delta \vdash l : T$ and, (SR) for all Γ and U, $\Gamma \vdash r : U$ whenever $\Gamma \vdash l : U$.

Given a set $\mathscr{R}$ of rewrite rules, let $\to_{\mathscr{R}}$ denote the smallest monotone and stable relation containing $\mathscr{R}$. The condition (SR) implies that $\to_{\mathscr{R}}$ preserves typing: if $\Gamma \vdash t : U$ and $t \to_{\mathscr{R}} u$, then $\Gamma \vdash u : U$ (subject-reduction property). Note that it is satisfied if, for instance, l contains no abstraction and no subterm of the form xt (Barbanera *et al.*, 1997).

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

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

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

Given B and t, let $\mathbb{C}^{\mathsf{B}}_{\to^*}(t) = \{(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}} \mid t \to^* \mathsf{c}\vec{t}\}$ be the set of triples $(\mathsf{c},\vec{t},\vec{T})$ such that t reduces to $\mathsf{c}\vec{t}$, c is maximally applied in $\mathsf{c}\vec{t}$, and $\vec{T}$ are the types of the arguments of c.

Given a relation R, let $\vec{x}\, R_{\mathrm{prod}}\, \vec{y}$ 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 $R_1,\dots,R_n$, let $\vec{x}\ (R_1,\dots,R_n)_{\mathrm{lex}}\ \vec{y}$ 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 $(R_1,\dots,R_n)_{\mathrm{lex}}$ are well-founded whenever $R, R_1,\dots,R_n$ so are.

2.5 Computability

Following Tait (Tait, 1967), Girard (Girard, 1972; Girard *et al.*, 1988), Mendler (Mendler, 1987), Okada (Okada, 1989), Breazu-Tannen and Gallier (Breazu-Tannen & Gallier, 1989), and Jouannaud and Okada (Jouannaud & Okada, 1991; Blanqui *et al.*, 2002), ... 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, to handle 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 (Blanqui, 2016; Riba, 2009) 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 $x\vec{v}$, $(\lambda xt)u\vec{v}$ or $\mathsf{f}\vec{t}$ with $|\vec{t}| \geq \sup\{|\vec{l}| \mid \exists r, \mathsf{f}\vec{l} \to r \in \mathscr{R}\}$[1]. A *computability predicate* is a set of terms $\mathscr{S}$ satisfying the following properties:

[1] The supremum exists since, by assumption, for all $\mathsf{f}\vec{l} \to r \in \mathscr{R}$, $\mathsf{f}\vec{l}$ is typable and thus $|\vec{l}| \leq \mathrm{r}^{\mathsf{f}}$.

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

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, that is, 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:

- the set $\mathbb{V}$ of variables is included in every computability predicate;
- given a computability predicate $\mathscr{S}$, $(\lambda x^U v)u \in \mathscr{S}$ iff $v\{(x,u)\} \in \mathscr{S}$ 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 system $\mathscr{R} = \{\mathsf{f} \to \mathsf{a}, \mathsf{f} \to \mathsf{b}\}$, if $\mathbb{P}(\mathscr{X})$ denotes the smallest computability predicate containing $\mathscr{X}$, 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 (Riba, 2007; Riba, 2008), Riba proves that his set of computability predicates is stable by union if $\mathscr{R}$ is an orthogonal constructor rewrite system.
- In (Werner, 1994) (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 non-empty 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 $\mathscr{S} \in \mathbb{Q}$ such that $t \in \mathscr{S}$. Since $\mathscr{S} \subseteq \mathrm{SN}$, we have $t \in \mathrm{SN}$.
- Let $t \in \bigcup \mathbb{Q}$ and u such that $t \to u$. Then, there is $\mathscr{S} \in \mathbb{Q}$ such that $t \in \mathscr{S}$. Since $\to(\mathscr{S}) \subseteq \mathscr{S}$, we have $u \in \mathscr{S}$ and thus $u \in \mathbb{Q}$.
- Let t be a neutral term such that $\to(t) \subseteq \bigcup \mathbb{Q}$. If $\to(t) = \emptyset$, then t belongs to every element of $\mathbb{Q}$. Therefore, $t \in \bigcup \mathbb{Q}$. 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 $\mathscr{S}_i \in \mathbb{Q}$ such that $t_i \in \mathscr{S}_i$. Since $\mathbb{Q}$ is well-ordered wrt. inclusion, there is $k \in \{1, \ldots, n\}$ such that $\mathscr{S}_k$ is the biggest element of $\{\mathscr{S}_1, \ldots, \mathscr{S}_n\}$ wrt. inclusion. Hence, $\to(t) \subseteq \mathscr{S}_k$ and $t \in \mathscr{S}_k$. Therefore, $t \in \bigcup \mathbb{Q}$. ■

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 $\tilde{\mathbb{I}} : \mathbb{T} \to \wp(\mathbb{L})$ as follows:

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

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

Note also that $\mathscr{U} \Rightarrow \mathscr{V}$ is a computability predicate whenever $\mathscr{U}$ and $\mathscr{V}$ so are. Hence, $\tilde{\mathbb{I}}(T)$ 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 $\mathsf{c} : (\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow \mathsf{B}$, $\mathsf{f} : \mathsf{B} \Rightarrow (\mathsf{B} \Rightarrow \mathsf{C})$, $\mathscr{R} = \{\mathsf{f}(\mathsf{c}\ x) \to x\}$ 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 $t(\mathsf{c}\ t) \to_\beta \mathsf{f}(\mathsf{c}\ t)(\mathsf{c}\ t) \to_{\mathscr{R}} t(\mathsf{c}\ t)$. 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 (Mendler, 1987):

Definition 3 (Positive and negative positions in a type) The subsets of *positive* ($s = +$) and *negative* ($s = -$) *positions* in a type T, $\mathrm{Pos}^s(T)$, are defined as follows:

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

where $-- = +$ and $-+ = -$.

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

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:

[2] Arguments can be permuted if needed.

- the arguments 1 to p^c are accessible and recursive,
- the arguments $\mathsf{p}^\mathsf{c}+1$ to q^c are accessible and not recursive:

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

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) (Peano, 1889), we can take $\mathsf{p}^\mathsf{0} = \mathsf{q}^\mathsf{0} = 0$ and $\mathsf{p}^\mathsf{s} = \mathsf{q}^\mathsf{s} = 1$ 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) (Howard, 1972), we can take $\mathsf{p}^\mathsf{zero} = \mathsf{q}^\mathsf{zero} = 0$, $\mathsf{p}^\mathsf{succ} = \mathsf{q}^\mathsf{succ} = 1$ since O occurs only positively in O, and $\mathsf{p}^\mathsf{lim} = \mathsf{q}^\mathsf{lim} = 1$ since O occurs only positively in $\mathsf{N} \Rightarrow \mathsf{O}$ if one takes $\mathsf{N} <_\mathbb{S} \mathsf{O}$. Now, for the sort L of lists of natural numbers with the constructors $\mathsf{nil} : \mathsf{L}$ and $\mathsf{cons} : \mathsf{L} \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}$, we can take $\mathsf{p}^\mathsf{cons} = 1$ and $\mathsf{q}^\mathsf{cons} = 2$ if one takes $\mathsf{N} <_\mathbb{S} \mathsf{L}$.

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 proof assistant (Coquand & Paulin-Mohring, 1988). Here, we do not forbid non-positive arguments and do not require arguments to be strictly positive. Hence, one can have a sort D with the constructors $\mathsf{app} : \mathsf{D} \Rightarrow \mathsf{D} \Rightarrow \mathsf{D}$ and $\mathsf{lam} : (\mathsf{D} \Rightarrow \mathsf{D}) \Rightarrow \mathsf{D}$, for which we must have $\mathsf{p}^\mathsf{lam} = \mathsf{q}^\mathsf{lam} = 0$ 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 by themselves 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 types are identified (*e.g.* $\mathsf{add\ empty\ empty}$ is typable if $\mathsf{T} = \mathsf{F}$).

Since $<_\mathbb{S}$ is well-founded, we 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:

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

where $\widetilde{[\mathsf{B} : \mathscr{X}, \mathbb{I}]}$ is introduced in Definition 2.

That such a least fixpoint exists follows from Knaster and Tarski's fixpoint theorem (Knaster & Tarski, 1928; Tarski, 1955) and the following fact:

[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 $\mathrm{Pos}(\mathsf{B}, \vec{U}) = \emptyset$.

Proposition 1 ((Blanqui, 2005b)) 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, $[\widetilde{\mathsf{B} : \mathscr{X}}, \mathbb{I}](T)$ is monotone wrt. $\mathscr{X}$.

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

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

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 (Hartogs, 1915), 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 $|\wp(\mathbb{L})| = 2^{\omega}$ (Hrbacek & Jech, 1999).

3.1 Stratifications

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

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

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

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

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

Because $\to$ is finitely branching, we immediately remark:

Lemma 2 For all continuous stratifications $\mathscr{S}$ and limit ordinal $0 < \mathfrak{a} \leq \mathfrak{h}$, we have $\mathscr{S}_{\mathfrak{a}} = \bigcup(\{\mathscr{S}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$.

Proof. By definition, $\mathscr{S}$ is monotone. So, for all $\mathfrak{a} \leq \mathfrak{h}$, $\{\mathscr{S}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\}$ is well-ordered wrt. inclusion. Since $\to$ is finitely branching, the conclusion follows from Lemma 1. ■

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

Lemma 3 Let $\mathscr{S}$ be a stratification and $t \in \mathscr{S}_{\mathfrak{h}}$.

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

Proof.

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

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

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

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

Lemma 4 If $\mathscr{S}$ is a stratification of B, $v \in \vec{U} \Rightarrow \mathsf{B}$ and $\mathrm{Pos}(\mathsf{B}, \vec{U}) = \emptyset$, then $o_{[\mathsf{B}:\mathscr{S}](\vec{U}\Rightarrow\mathsf{B})}(v) = \sup\{o_{\mathscr{S}}(v\vec{u}) \mid \vec{u} \in \vec{U}\}$.

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

A continuous stratification of a sort B can be obtained by the transfinite iteration of $\mathbb{H}^{\mathsf{B}}$ from the smallest computability predicate $\bot$ (Kuratowski, 1922; Cousot & Cousot, 1979):

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

The fact that $\mathscr{D}^{\mathsf{B}}$ is monotone follows from the facts that $\mathscr{D}^{\mathsf{B}}_0 \subseteq \mathscr{D}^{\mathsf{B}}_1$ and $\mathbb{H}^{\mathsf{B}}$ is monotone (Cousot & Cousot, 1979). Now, by definition of $\mathfrak{h}$, $\mathscr{D}^{\mathsf{B}}$ is not injective. Therefore, there are $\mathfrak{c} < \mathfrak{d} < \mathfrak{h}$ such that $\mathscr{D}^{\mathsf{B}}_{\mathfrak{c}} = \mathscr{D}^{\mathsf{B}}_{\mathfrak{d}}$. Since $\mathscr{D}^{\mathsf{B}}$ is monotone, $\mathscr{D}^{\mathsf{B}}_{\mathfrak{c}} = \mathscr{D}^{\mathsf{B}}_{\mathfrak{c}+1} = \mathscr{D}^{\mathsf{B}}_{\mathfrak{d}} = \mathscr{D}^{\mathsf{B}}_{\mathfrak{h}} = \mathsf{B}$ (Rubin & Rubin, 1963).

We call this stratification the *default* stratification. It is the one used in all the previous works on sized types, except in (Abel, 2012) where, after (Sprenger & Dam, 2003), Abel uses a stratification having better properties, namely $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}(\{\mathbb{H}^{\mathsf{B}}(\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}}) \mid \mathfrak{b} < \mathfrak{a}\})$.

The size wrt. the default stratification of a term t is the set-theoretical height of the tree representation of t when abstractions are interpreted as set-theoretical functions. 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 $\mathsf{inj}\ 0 \to$

zero and $\mathsf{inj}\ (\mathsf{s}\ x) \to \mathsf{succ}\ (\mathsf{inj}\ x)$. Let us prove that $o_{\mathscr{D}^{\mathsf{O}}}(\mathsf{lim\ inj}) = \omega + 1$. By definition, $o_{\mathscr{D}^{\mathsf{O}}}(\mathsf{lim\ inj})$ is the smallest ordinal $\mathfrak{a}$ such that $\mathsf{lim\ inj} \in \mathscr{D}^{\mathsf{O}}_{\mathfrak{a}}$. By definition of $\mathscr{D}$, $\mathfrak{a} = o_{\mathscr{S}}(\mathsf{inj}) + 1$ where $\mathscr{S} = [\mathsf{O} : \mathscr{D}^{\mathsf{O}}](\mathsf{N} \Rightarrow \mathsf{O})$. By Lemma 4, $o_{\mathscr{S}}(\mathsf{inj}) = \sup\{o_{\mathscr{D}^{\mathsf{O}}}(\mathsf{inj}\ t) \mid t \in \mathsf{N}\}$. 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, $o_{\mathscr{D}^{\mathsf{O}}}(\mathsf{inj}\ t) < \omega$. Finally, one can easily prove that, for all $n < \omega$, $o_{\mathscr{D}^{\mathsf{O}}}(\mathsf{inj}(\mathsf{s}^n\mathsf{0})) = n+1$. Therefore, $o_{\mathscr{S}}(\mathsf{inj}) = \omega$ and $o_{\mathscr{D}^{\mathsf{O}}}(\mathsf{lim\ inj}) = \omega + 1$. Moreover, $\mathscr{S}$ is not continuous since $\mathsf{inj} \in \mathscr{S}_\omega - \bigcup\{\mathscr{S}_n \mid n < \omega\}$. ■

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

3.2 Stratifications based on constructor size functions

We now introduce a general way of defining a stratification:

Definition 7 (Constructor 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 $\mathsf{B}^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$ 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 $\mathsf{B}^{\mathsf{c}}_k = \mathsf{B}$ if $k \in \{1, \ldots, \mathrm{p}^{\mathsf{c}}\}$).

In practice, there is usually no choice for $\mathsf{B}^{\mathsf{c}}_k$. For having a choice, the order of T_k must be greater than or equal to 2. For instance, if $T_k = (\mathsf{C} \Rightarrow \mathsf{D}) \Rightarrow \mathsf{E}$, then one can choose between C and E if both are different from D.

On the other hand, there are many possible choices for Σ^{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.

Interestingly, Σ^{c} may depend on all accessible arguments, including the non-recursive ones. 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 $\mathrm{p}^{\mathsf{pair}} = 0$, $\mathrm{q}^{\mathsf{pair}} = 2$, $\mathsf{B}^{\mathsf{pair}}_1 = \mathsf{B}^{\mathsf{pair}}_2 = \mathsf{N}$ and $\Sigma^{\mathsf{pair}}(\mathfrak{a}, \mathfrak{b}) = \mathfrak{a} + \mathfrak{b}$.

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

- $\mathscr{S}^{\mathsf{B}}_0$ 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),
— $\forall k \in \{\mathrm{p}^{\mathsf{c}}+1,\dots,\mathrm{q}^{\mathsf{c}}\}$, $t_k \in T_k$,
— $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq 0$.

- $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1}$ is the set of terms $t \in \mathrm{SN}$ such that, for all $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$:

— $\forall k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$, $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$
— $\forall k \in \{\mathrm{p}^{\mathsf{c}}+1,\dots,\mathrm{q}^{\mathsf{c}}\}$, $t_k \in T_k$
— $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}+1$.

- $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}(\{\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\})$ if $\mathfrak{a}$ is an infinite limit ordinal.

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

- in the case of $\mathscr{S}^{\mathsf{B}}_0$:

— $\mathrm{p}^{\mathsf{c}} = 0$ and thus, for all $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$, $o_{\mathscr{S}^{\mathsf{c}},k}(t_k) = o_{[\mathsf{B}^{\mathsf{c}}_k : \mathscr{S}^{\mathsf{B}^{\mathsf{c}}_k}]T_k}(t_k)$ is well-defined since $t_k \in T_k$ and $\mathsf{B}^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$.

- in the case of $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1}$:

— $\forall k \leq \mathrm{p}^{\mathsf{c}}$, $o_{\mathscr{S}^{\mathsf{c}},k}(t_k) = o_{[\mathsf{B}:\mathscr{S}^{\mathsf{B}}]T_k}(t_k)$ is well-defined and $\leq \mathfrak{a}$ since $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$;
— $\forall k \in \{\mathrm{p}^{\mathsf{c}}+1,\dots,\mathrm{q}^{\mathsf{c}}\}$, $o_{\mathscr{S}^{\mathsf{c}},k}(t_k)$ is well-defined since $t_k \in T_k$ and $\mathsf{B}^{\mathsf{c}}_k <_{\mathbb{S}} \mathsf{B}$.

The definition of $\mathscr{S}^{\mathsf{B}}$ 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 $\mathscr{S}^{\mathsf{B}}$ by taking $\mathscr{S}^{\mathsf{B}}_0 = \bot$. The current definition has the advantage that both variables and nullary constructors whose size function is 0 have size 0. Hence, if one takes $\Sigma_0 = \Sigma_{\mathsf{s}}(\mathfrak{a}) = 0$, then $o_{\mathscr{S}^{\mathsf{N}}}(\mathsf{s}^i x) = o_{\mathscr{S}^{\mathsf{N}}}(\mathsf{s}^i 0) = i$ while, in the default stratification, $o_{\mathscr{D}^{\mathsf{N}}}(\mathsf{s}^i x) = i$ and $o_{\mathscr{D}^{\mathsf{N}}}(\mathsf{s}^i 0) = i+1$ (nullary constructors do not belong to $\bot$).

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

Lemma 5 For every sort B and ordinal $\mathfrak{a} < \mathfrak{h}$, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B}$.

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

- Let $t \in \mathscr{S}^{\mathsf{B}}_0$. Then, $t \in \mathrm{SN}$. Let now $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$. 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 $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1}$. Then, $t \in \mathrm{SN}$. Let now $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$. If $k \leq \mathrm{p}^{\mathsf{c}}$, then $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$. By induction hypothesis, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B}$. Since B occurs only positively in T_k, Proposition 1 gives $[\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k \subseteq [\mathsf{B} : \mathsf{B}]T_k = T_k$. Therefore, $t_k \in T_k$. Now, if $k \in \{\mathrm{p}^{\mathsf{c}}+1,\dots,\mathrm{q}^{\mathsf{c}}\}$, then $t_k \in T_k$ too. Therefore, $t \in \mathsf{B}$ since $\mathsf{B} = \mathbb{H}^{\mathsf{B}}(\mathsf{B})$.
- Let $\mathfrak{a}$ be an infinite limit ordinal. Then, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}\{\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\}$. For every $\mathfrak{b} < \mathfrak{a}$, by induction hypothesis, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathsf{B}$. Therefore, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B}$. ■

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

Proof. We proceed by induction on $<_{\mathbb{S}}$ and $\mathfrak{a}$. If $\mathfrak{a}$ is an infinite limit ordinal, then $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$ is a computability predicate by definition of lub since, by induction hypothesis, for all $\mathfrak{b} < \mathfrak{a}$, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$ 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 $\mathscr{S}^{\mathsf{B}}_0 = \mathrm{SN}^{\mathsf{B}}(P_0)$ for some predicate P_0, and $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1} = \mathrm{SN}^{\mathsf{B}}(P_{\mathfrak{a}+1})$ for some predicate $P_{\mathfrak{a}+1}$. However, for all predicates 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, $\mathscr{S}^{\mathsf{B}}$ is monotone.

Proof. We prove that, for all $(\mathfrak{a},\mathfrak{b},\mathfrak{c})$, if $\mathfrak{b} \le \mathfrak{c} \le \mathfrak{a}$, then (1) $\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{c}}$, hence $\mathscr{S}^{\mathsf{B}}|_{\mathfrak{a}}$ is monotone, (2) $\mathscr{S}^{\mathsf{B}}_{\mathfrak{c}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$, and (3) $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1}$, by induction on $\mathfrak{a}$. There are 3 cases:

- $\mathfrak{a} = 0$. Then, $\mathfrak{b} = \mathfrak{c} = 0$ and (1) and (2) hold trivially. We now prove (3). Let $t \in \mathscr{S}^{\mathsf{B}}_0$. We prove that $t \in \mathscr{S}^{\mathsf{B}}_1$:
  — $t \in \mathrm{SN}$ since, by definition, $\mathscr{S}^{\mathsf{B}}_0 \subseteq \mathrm{SN}$.

  Let now $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$.
  — We have to prove that, for all $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$, $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_0]T_k$. Since $t \in \mathscr{S}^{\mathsf{B}}_0$, we have $\mathrm{p}^{\mathsf{c}} = 0$. Therefore, the property holds since there is no $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$.
  — We have to prove that, for all $k \in \{\mathrm{p}^{\mathsf{c}}+1,\dots,\mathrm{q}^{\mathsf{c}}\}$, $t_k \in T_k$. This holds since $t \in \mathscr{S}^{\mathsf{B}}_0$.
  — Finally, we have to prove that $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \le 1$. This holds since $t \in \mathscr{S}^{\mathsf{B}}_0$ and thus $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \le 0$.
- $\mathfrak{a} = \mathfrak{a}' + 1$.
  1. If $\mathfrak{c} \le \mathfrak{a}'$ then (1) holds by induction hypothesis (1) on $(\mathfrak{a}',\mathfrak{b},\mathfrak{c})$. Otherwise $\mathfrak{c} = \mathfrak{a}'+1$. If $\mathfrak{b} = \mathfrak{c}$, then (1) holds trivially. Otherwise $\mathfrak{b} \le \mathfrak{a}'$ and (1) holds by induction hypothesis (1) and (3) on $(\mathfrak{a}',\mathfrak{b},\mathfrak{a}')$, and transitivity of $\le$.
  2. If $\mathfrak{c} \le \mathfrak{a}'$ then (2) holds by induction hypothesis (2) and (3) on $(\mathfrak{a}',\mathfrak{b},\mathfrak{c})$, and transitivity of $\le$. Otherwise (2) holds trivially.
  3. Let $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1}$. We prove that $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+2}$:
     — $t \in \mathrm{SN}$ since, by definition, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1} \subseteq \mathrm{SN}$.

     Let now $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$.
     — Assume that $k \le \mathrm{p}^{\mathsf{c}}$. Since $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1}$, we have $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'}]T_k$. Therefore, $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1}]T_k$ since B occurs only positively in T_k and $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1}$ by induction hypothesis (3) on $(\mathfrak{a}',\mathfrak{a}',\mathfrak{a}')$.
     — Assume that $k > \mathrm{p}^{\mathsf{c}}$. Then, $t_k \in T_k$ since $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1}$.
     — Since $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}'+1}$, we have $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \le \mathfrak{a}'+1$. Therefore, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \le \mathfrak{a}'+2$.
- $\mathfrak{a}$ is an infinite limit ordinal. Then, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}\{\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\}$.
  1. If $\mathfrak{c} < \mathfrak{a}$, then (1) follows by induction hypothesis (1) on $(\mathfrak{c},\mathfrak{b},\mathfrak{c})$. Otherwise, $\mathfrak{c} = \mathfrak{a}$. If $\mathfrak{b} = \mathfrak{c}$ then (1) holds trivially. Otherwise, $\mathfrak{b} < \mathfrak{c}$ and (1) holds by definition of lub.
  2. (2) holds by definition of lub.

3. Let $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$. We have to prove that $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1}$.
   After (1), $\mathscr{S}^{\mathsf{B}}|_{\mathfrak{a}}$ is monotone. Therefore, by Lemma 1, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} = \bigcup\{\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\}$ and $t \in \mathscr{S}^{\mathsf{B}}$ for some $\mathfrak{b} < \mathfrak{a}$. Now, since $\mathfrak{a}$ is a limit ordinal, $\mathfrak{b}+1 < \mathfrak{a}$. Therefore, by induction hypothesis (2) on $(\mathfrak{b}+1, \mathfrak{b}, \mathfrak{b})$, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$ and $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$. We now prove that $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}+1}$:
   — $t \in \mathrm{SN}$ since $\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}}$ is a computability predicate.
   Let now $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$ and $k \in \{1, \ldots, \mathrm{q}^{\mathsf{c}}\}$.
   — Assume that $k \leq \mathrm{p}^{\mathsf{c}}$. Since $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$, we have $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}}]T_k$. Therefore, $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}]T_k$ since B occurs only positively in T_k and $\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{a}}$.
   — Assume that $k > \mathrm{p}^{\mathsf{c}}$. Then, $t_k \in T_k$ since $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$.
   — Since $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$, we have $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{b}+1$. Therefore, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}+1$. ■

Lemma 8 For every sort B, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{h}} = \mathsf{B}$.

Proof. By Lemma 5, $\mathscr{S}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathsf{B}$. Now, since $\mathsf{B} = \mathscr{D}^{\mathsf{B}}_{\mathfrak{h}}$,where $\mathscr{D}^{\mathsf{B}}$ is the default stratification, it suffices to prove that, for all $\mathfrak{a}$, $\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{h}}$, that is, for all $\mathfrak{a}$, there is $\mathfrak{b} < \mathfrak{h}$ such that $\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}}$. We proceed by induction on $<_{\mathbb{S}}$ and $\mathfrak{a}$.

- $\mathscr{D}^{\mathsf{B}}_{0} = \bot \subseteq \mathscr{S}^{\mathsf{B}}_{0}$.
- Let $\mathfrak{a}$ be an infinite limit ordinal smaller than $\mathfrak{h}$. By induction hypothesis, for all $\mathfrak{b} < \mathfrak{a}$, $\mathscr{D}^{\mathsf{B}}_{\mathfrak{b}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{h}}$. Therefore, $\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}} = \mathrm{lub}\{\mathscr{D}^{\mathsf{B}}_{\mathfrak{b}} \mid \mathfrak{b} < \mathfrak{a}\} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{h}}$.
- Let now $\mathfrak{a}+1 < \mathfrak{h}$. By induction hypothesis, $\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{h}}$.
  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 $\mathfrak{c} = \sup(X)$ where $X = \{o_{\mathscr{S}^{\mathsf{B}}}(t) \mid t \in \mathscr{D}^{\mathsf{B}}_{\mathfrak{a}}\}$. Since $|X| \leq |\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}}| \leq |\mathbb{L}| \leq \omega$, we have $\mathfrak{c} < \mathfrak{h}$ and $\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{c}}$.
  Let now $\mathfrak{d} = \sup(X \cup Y)$ where Y is the set of the ordinals $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t}))$ such that there are $t \in \mathscr{D}^{\mathsf{B}}_{\mathfrak{a}}$ and $(\mathsf{c}, \vec{t}, \vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$. Since $|Y| \leq \omega$ ($\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}} \subseteq \mathrm{SN}$ and $\to$ is finitely branching), we have $\sup(Y) < \mathfrak{h}$ and thus $\mathfrak{d} < \mathfrak{h}$. Since $\mathfrak{h}$ is a limit ordinal, $\mathfrak{d}+1 < \mathfrak{h}$. We now prove that $\mathscr{D}^{\mathsf{B}}_{\mathfrak{a}+1} \subseteq \mathscr{S}^{\mathsf{B}}_{\mathfrak{d}+1}$. Let $t \in \mathscr{D}^{\mathsf{B}}_{\mathfrak{a}+1}$. 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 $k > \mathrm{p}^{\mathsf{c}}$, then $t_k \in T_k$. Otherwise, $t_k \in [\mathsf{B} : \mathscr{D}^{\mathsf{B}}_{\mathfrak{a}}]T_k$. Since B occurs only positively in T_k, we have $[\mathsf{B} : \mathscr{D}^{\mathsf{B}}_{\mathfrak{a}}]T_k \subseteq [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k$. Since $\mathfrak{c} \leq \mathfrak{d}$ and $\mathscr{S}^{\mathsf{B}}$ is monotone by Lemma 7, we have $[\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{c}}]T_k \subseteq [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{d}}]T_k$. Finally, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{d}$. Therefore, $t \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{d}+1}$. ■

This ends the proof that $\mathscr{S}^{\mathsf{B}}$ is a stratification of B. We now see some of its properties:

Lemma 9

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

Proof.

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

Lemma 10 If $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $\mathsf{c}\vec{t} \in \mathsf{B}$, then:

- $o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t}) \geq \Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t}))$.
- $o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t}) > o_{\mathscr{S}^{\mathsf{c},k}}(t_k)$ for all $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$.

Proof. Let $\mathfrak{a} = o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t})$. Since $\mathscr{S}^{\mathsf{B}}$ is continuous, by Lemma 3, either $\mathfrak{a} = 0$ or $\mathfrak{a} = \mathfrak{b}+1$ for some $\mathfrak{b}$.

- If $\mathfrak{a} = 0$, then $\mathsf{c}\vec{t} \in \mathscr{S}^{\mathsf{B}}_0$ and $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}$ by definition of $\mathscr{S}^{\mathsf{B}}_0$. Otherwise, $\mathsf{c}\vec{t} \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$ and $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}$ by definition of $\mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$.
- If $\mathfrak{a} = 0$, then $\mathsf{c}\vec{t} \in \mathscr{S}^{\mathsf{B}}_0$ and $\mathrm{p}^{\mathsf{c}} = 0$. So, there is no $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$. Otherwise, $\mathsf{c}\vec{t} \in \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}+1}$ and $t_k \in [\mathsf{B} : \mathscr{S}^{\mathsf{B}}_{\mathfrak{b}}]T_k$. Thus, $o_{\mathscr{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{b} < \mathfrak{a}$. ■

Lemma 11 If $t \in \mathsf{B}$, then $o_{\mathscr{S}^{\mathsf{B}}}(t) = \delta\sup(R \cup S \cup T)$ where:

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

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

We first prove that $\mathfrak{b} \geq \delta\mathfrak{a}$. Let t' such that $t \to t'$. Then, $\mathfrak{b} \geq o_{\mathscr{S}^{\mathsf{B}}}(t')$ by Lemma 3. Assume now that $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $t = \mathsf{c}\vec{t}$. By Lemma 10, $\mathfrak{b} \geq \Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t}))$ and, for all $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$, $\mathfrak{b} > o_{\mathscr{S}^{\mathsf{c},k}}(t_k)$. Therefore, $\mathfrak{b} \geq \mathfrak{a}$.

Since $\mathscr{S}^{\mathsf{B}}$ 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 $\mathfrak{b} \geq \mathfrak{a}+1 = \delta\mathfrak{a}$. Otherwise, $\delta\mathfrak{a} = \mathfrak{a}$ and $\mathfrak{b} \geq \delta\mathfrak{a}$.

Now, to have $\mathfrak{b} \leq \delta\mathfrak{a}$, we prove that $t \in \mathscr{S}^{\mathsf{B}}_{\delta\mathfrak{a}}$ using Lemma 9:

- Case $\delta\mathfrak{a} = 0$. Then, $\mathfrak{a} = 0$. Let $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$.
  - — Case $t = \mathsf{c}\vec{t}$. Then, $S = \emptyset$, $\mathrm{p}^{\mathsf{c}} = 0$ and $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) = 0$. Therefore, $t \in \mathscr{S}^{\mathsf{B}}_0$.
  - — Case $t \to t' \to^* \mathsf{c}\vec{t}$. Then, $o_{\mathscr{S}^{\mathsf{B}}}(t') = 0$. So, $\mathrm{p}^{\mathsf{c}} = 0$, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) = 0$ and $t \in \mathscr{S}^{\mathsf{B}}_0$.
- Case $\delta\mathfrak{a} = \mathfrak{a}'+1$. Let $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}_{\to^*}(t)$.
  - — Case $t = \mathsf{c}\vec{t}$. First, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \sup(T) \leq \mathfrak{a} \leq \delta\mathfrak{a} = \mathfrak{a}'+1$. Second, if $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$, then $o_{\mathscr{S}^{\mathsf{c},k}}(t_k) < o_{\mathscr{S}^{\mathsf{c},k}}(t_k)+1 \leq \sup(S) \leq \mathfrak{a} \leq \delta\mathfrak{a}$. Therefore, $o_{\mathscr{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}'$ and $t \in \mathscr{S}_{\delta\mathfrak{a}}$.
  - — Case $t \to t' \to^* \mathsf{c}\vec{t}$. First, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq \mathfrak{a}'+1$ since, by Lemma 10, $\Sigma^{\mathsf{c}}(o_{\mathscr{S}^{\mathsf{c}}}(\vec{t})) \leq o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t})$ and, by Lemma 3, $o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t}) \leq o_{\mathscr{S}^{\mathsf{B}}}(t') \leq \sup(S) \leq \mathfrak{a} \leq \delta\mathfrak{a}$. Second, if $k \in \{1,\dots,\mathrm{p}^{\mathsf{c}}\}$, then $o_{\mathscr{S}^{\mathsf{c},k}}(t_k) \leq \mathfrak{a}'$ since, by Lemma 10, $o_{\mathscr{S}^{\mathsf{c},k}}(t_k) < o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t})$ and, by Lemma 3, $o_{\mathscr{S}^{\mathsf{B}}}(\mathsf{c}\vec{t}) \leq o_{\mathscr{S}^{\mathsf{B}}}(t') \leq \sup(R) \leq \mathfrak{a} \leq \delta\mathfrak{a} = \mathfrak{a}'+1$. So, $t \in \mathscr{S}_{\delta\mathfrak{a}}$. ■

Note 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} gives 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 never 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 $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{t}) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{t}))$.

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

Corollary 2 Assume that Σ^{c} is monotone wrt. every argument, strictly extensive wrt. recursive arguments and never returns an infinite limit ordinal. Then, for all $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}$ with $\mathsf{c}\vec{t} \in \mathsf{B}$, we have $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{t}) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{t}))$.

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, $o_{\mathscr{J}^{\mathsf{c}}}(\vec{u}) \leq_{\mathrm{prod}} o_{\mathscr{J}^{\mathsf{c}}}(\vec{t})$ and, by induction hypothesis, $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{u}) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{u}))$. So, $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{u}) \leq \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{t}))$ since Σ^{c} is monotone. Therefore, $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{t}) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{t}))$. ■

Finally, we are going to prove that, if $\to$ is locally confluent, hence confluent on strongly normalizing terms (Newman, 1942), 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[4] $\mathsf{C} <_{\mathbb{S}} \mathsf{B}$ 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 numbers and Howard constructive ordinals.

Lemma 12 Assume that $\to$ is locally confluent and, for every constructor c, Σ^{c} is monotone wrt. every argument, strictly extensive wrt. recursive arguments and never returns an infinite limit ordinal. Then, for every strictly positive sort B and term $t \in \mathsf{B}$, $o_{\mathscr{J}^{\mathsf{B}}}(t) = o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow})$, where $t{\downarrow}$ is the normal form of t.

Proof. First note that $o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow}) \leq o_{\mathscr{J}^{\mathsf{B}}}(t)$ since $t \to^* t{\downarrow}$. We now prove that, for all strictly positive B, for all $t \in \mathsf{B}$, $o_{\mathscr{J}^{\mathsf{B}}}(t) \leq o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow})$, hence that $o_{\mathscr{J}^{\mathsf{B}}}(t) = o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow})$, by induction on $(\mathsf{B}, o_{\mathscr{J}^{\mathsf{B}}}(t), t)$ with $(<_{\mathbb{S}}, <, \leftarrow)_{\mathrm{lex}}$ as well-founded relation. By Lemma 11, $o_{\mathscr{J}^{\mathsf{B}}}(t) = \delta\sup(R \cup S \cup T)$. Since Σ^{c} is strictly extensive, $o_{\mathscr{J}^{\mathsf{B}}}(t) = \delta\sup(R \cup T)$. Since $o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow})$ cannot be an infinite limit ordinal, it is sufficient to prove that $\sup(R \cup T) \leq o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow})$.

Assume that $t \to u$. Then, $o_{\mathscr{J}^{\mathsf{B}}}(u) \leq o_{\mathscr{J}^{\mathsf{B}}}(t)$. Hence, by induction hypothesis on the 2nd or 3rd component, $o_{\mathscr{J}^{\mathsf{B}}}(u) \leq o_{\mathscr{J}^{\mathsf{B}}}(u{\downarrow}) = o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow})$.

[4] This is a restriction wrt. the definition given in (Coquand & Paulin-Mohring, 1988) where T_k can be any type where B does not occur.

Assume now that $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $t = \mathsf{c}\vec{t}$. By Corollary 2, $o_{\mathscr{J}^{\mathsf{B}}}(t) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{t}))$ and $o_{\mathscr{J}^{\mathsf{B}}}(t{\downarrow}) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c}}}(\vec{t}{\downarrow}))$. Since Σ^{c} is monotone, it suffices to prove that, for all $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$, $o_{\mathscr{J}^{\mathsf{c},k}}(t_k) \leq o_{\mathscr{J}^{\mathsf{c},k}}(t_k{\downarrow})$. 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, $o_{\mathscr{J}^{\mathsf{c},k}}(t_k) = o_{\mathscr{J}^{\mathsf{C}}}(t_k) \leq o_{\mathscr{J}^{\mathsf{C}}}(t_k{\downarrow}) = o_{\mathscr{J}^{\mathsf{c},k}}(t_k{\downarrow})$.
- There is $\vec{U}$ such that $T_k = \vec{U} \Rightarrow \mathsf{B}$ and $\mathrm{Pos}(\mathsf{B},\vec{U}) = \emptyset$. Then, by Lemma 4, $o_{\mathscr{J}^{\mathsf{c},k}}(t_k) = \sup\{o_{\mathscr{J}^{\mathsf{B}}}(t_k\,\vec{u}) \mid \vec{u} \in \vec{U}\}$ and $o_{\mathscr{J}^{\mathsf{c},k}}(t_k{\downarrow}) = \sup\{o_{\mathscr{J}^{\mathsf{B}}}(t_k{\downarrow}\vec{u}) \mid \vec{u} \in \vec{U}\}$. Let $\vec{u} \in \vec{U}$. Since $o_{\mathscr{J}^{\mathsf{B}}}(t_k{\downarrow}\vec{u}) \leq o_{\mathscr{J}^{\mathsf{B}}}(t_k\,\vec{u}) < o_{\mathscr{J}^{\mathsf{B}}}(t)$, by induction hypothesis on the 2nd component, $o_{\mathscr{J}^{\mathsf{B}}}(t_k\,\vec{u}) = o_{\mathscr{J}^{\mathsf{B}}}(t_k{\downarrow}\vec{u})$. Therefore, $o_{\mathscr{J}^{\mathsf{c},k}}(t_k) = o_{\mathscr{J}^{\mathsf{c},k}}(t_k{\downarrow})$. ■

We end this section by introducing the reflexive and transitive closure of the notion of accessible argument (Definition 4) 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 $[\mathsf{B} : \mathscr{S}^{\mathsf{B}}]T$.

Definition 10 (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 $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$ such that $t = \mathsf{c}\vec{t}$, $T = \mathsf{B}$ and $(u,U,\mathsf{C}) \trianglelefteq_{\mathrm{a}} (t_k,T_k,\mathsf{B}^{\mathsf{c}}_k)$, where $\mathsf{B}^{\mathsf{c}}_k = \mathsf{B}$ if $k \leq \mathrm{p}^{\mathsf{c}}$, and $\mathsf{B}^{\mathsf{c}}_k$ is given by the size function of c if $k > \mathrm{p}^{\mathsf{c}}$ (see Definition 7).

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 $\mathsf{c} : (\mathsf{B} \Rightarrow \mathsf{C}) \Rightarrow \mathsf{B}$, because B occurs negatively in $\mathsf{B} \Rightarrow \mathsf{C}$ and thus $\mathrm{q}^{\mathsf{c}} = 0$.

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 13 If $(u,U,\mathsf{C}) \trianglelefteq_{\mathrm{a}} (t,T,\mathsf{B})$ and $t \in T$, then $u \in U$.

Proof. We proceed by induction on $\trianglelefteq_{\mathrm{a}}$. If $(u,U,\mathsf{C}) = (t,T,\mathsf{B})$, this is immediate. Otherwise, there are $(\mathsf{c},\vec{t},\vec{T}) \in \mathbb{C}^{\mathsf{B}}$ and $k \in \{1,\dots,\mathrm{q}^{\mathsf{c}}\}$ such that $t = \mathsf{c}\vec{t}$, $T = \mathsf{B}$ and $(u,U,\mathsf{C}) \trianglelefteq_{\mathrm{a}} (t_k,T_k,\mathsf{B}^{\mathsf{c}}_k)$. By definition of $\tilde{\mathbb{I}}(\mathsf{B})$, we have $t_k \in T_k$. So, by induction hypothesis, $u \in U$. ■

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 $\mathsf{f}l \to r$ 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 (Hughes *et al.*, 1996), consists of 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 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 (Definition 12). 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 (Presburger, 1929).

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 9, we prove that these conditions are in particular satisfied by the successor algebra.

Definition 11 (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-order $\leq_{\mathtt{A}}$ on $\mathtt{A}$ stable by substitution: $a\varphi \leq_{\mathtt{A}} b\varphi$ whenever $a \leq_{\mathtt{A}} b$ and $\varphi : \mathtt{V} \to \mathtt{A}$;
- a strict order $<_{\mathtt{A}} \subseteq \leq_{\mathtt{A}}$ also stable: $a\varphi <_{\mathtt{A}} b\varphi$ whenever $a <_{\mathtt{A}} b$ 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 $\mu : \mathtt{V} \to \mathfrak{h}$, $a\mu \leq b\mu$ ($a\mu < b\mu$ resp.) whenever $a \leq_{\mathtt{A}} b$ ($a <_{\mathtt{A}} b$ 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 wrt $\leq_{\mathtt{A}}$ in every argument, that is, $\mathtt{f}\,\vec{a} \leq_{\mathtt{A}} \mathtt{f}\,\vec{b}$ whenever $\vec{a}\,(\leq_{\mathtt{A}})_{\mathrm{prod}}\,\vec{b}$. Given a size substitution φ and a set V of variables, let $\varphi|_V = \{(\alpha,\alpha\varphi) \mid \alpha \in V\}$.

Let $a \leq_{\mathrm{ext}} b$ ($a <_{\mathrm{ext}} b$ resp.) iff, for all μ, $a\mu \leq b\mu$ ($a\mu < b\mu$ resp.). Note that $(\leq_{\mathrm{ext}}, <_{\mathrm{ext}})$ satisfies the above conditions and, for every pair of relations $(\leq_{\mathtt{A}}, <_{\mathtt{A}})$ satisfying the above conditions, we have $\leq_{\mathtt{A}} \subseteq \leq_{\mathrm{ext}}$ and $<_{\mathtt{A}} \subseteq <_{\mathrm{ext}}$. So, one can always take $\leq_{\mathrm{ext}}$ ($<_{\mathrm{ext}}$ resp.) for $\leq_{\mathtt{A}}$ ($<_{\mathtt{A}}$ resp.).

As remarked in (Giesl *et al.*, 2002), the strict part of a stable quasi-order $\leq_{\mathtt{A}}$, that is $\lneq_{\mathtt{A}} = \leq_{\mathtt{A}} - \geq_{\mathtt{A}}$, is not necessarily stable. On the other hand, its stable-strict part $<_{\mathtt{A}}$ is stable, where $a <_{\mathtt{A}} b$ iff, for all closed substitution φ, $a\varphi \lneq_{\mathtt{A}} b\varphi$.

The simplest size algebra is:

Definition 12 (Successor algebra) The *successor* size algebra is obtained by taking:

- $\mathtt{F} = \mathtt{C} \cup \{\mathtt{s}\}$ where $\mathtt{C}$ is an infinite set of constants and $\mathtt{s}$ a unary symbol interpreted by the successor function[5];
- $<_{\mathtt{A}}$ is the smallest strict ordering on $\mathtt{A}$ such that, for all a, $a <_{\mathtt{A}} \mathtt{s}\,a$;
- $\leq_{\mathtt{A}}$ is the reflexive closure of $<_{\mathtt{A}}$.

Although this algebra may seem overly simple, it is already sufficient to overtake the Coq termination checker (see Section 6 for various examples using it). We will study the properties of this algebra in Section 9.

Definition 13 (Size-annotated types) The set $\mathbb{T}_{\mathtt{A}}$ of 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 every annotation.

Given a sort B, a size expression a 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 definition of *positive* ($s = +$) and *negative* ($s = -$) *positions* in a type (Definition 3) is extended to annotated types as follows:

- $\mathrm{Pos}^s(\mathsf{B}_b) = \{1p \mid p \in \mathrm{Pos}^s(b)\}$;
- $\mathrm{Pos}^s(\alpha) = \{\varepsilon \mid s = +\}$;
- $\mathrm{Pos}^s(\mathtt{f}) = \{\varepsilon \mid s = +\}$ if $\mathtt{f}$ is of arity 0;
- $\mathrm{Pos}^s(\mathtt{f}\,b_1 \ldots b_n) = \{ip \mid i \in \mathrm{Mon}^+(\mathtt{f}), p \in \mathrm{Pos}^s(b_i)\} \cup \{ip \mid i \in \mathrm{Mon}^-(\mathtt{f}), p \in \mathrm{Pos}^{-s}(b_i)\}$ if $\mathtt{f}$ is of arity $n > 0$,

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

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 allows us to define a size function, hence a stratification for every sort, and thus an interpretation of every annotated type in computability predicates. To this end, we use notations similar to the ones of Definition 7:

[5] $\mathfrak{h}$ is closed by successor since it is a limit ordinal.

Definition 15 (Annotated types of constructors) We assume that every $\mathsf{c} \in \mathbb{C}$ with $\Theta(\mathsf{c}) = T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^\mathsf{c}} \Rightarrow \mathsf{B}$ is equipped with an annotated type $\overline{\Theta}(\mathsf{c}) = \overline{T_1} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}^\mathsf{c}}} \Rightarrow \mathsf{B}_{\sigma^\mathsf{c}}$ with:

- for all $i \in \{1,\ldots,\mathrm{q}^\mathsf{c}\}$, $\overline{T_i} = \mathrm{Annot}(T_i, \mathsf{B}_i^\mathsf{c}, \alpha_i^\mathsf{c})$;
- for all $i \in \{\mathrm{q}^\mathsf{c}+1,\ldots,\mathrm{r}^\mathsf{c}\}$, $\overline{T_i} = T_i$;
- $\alpha_1^\mathsf{c},\ldots,\alpha_{\mathrm{p}^\mathsf{c}}^\mathsf{c} \in \mathtt{V}$;
- $\alpha_{\mathrm{p}^\mathsf{c}+1}^\mathsf{c},\ldots,\alpha_{\mathrm{q}^\mathsf{c}}^\mathsf{c} \in \mathtt{V} \cup \{\infty\}$;
- the variables of $\{\alpha_1^\mathsf{c},\ldots,\alpha_{\mathrm{q}^\mathsf{c}}^\mathsf{c}\}$ are either pairwise equal or pairwise distinct;
- for all $i \in \{1,\ldots,\mathrm{p}^\mathsf{c}\}$, $\mathsf{B}_i^\mathsf{c} = \mathsf{B}$;
- for all $i \in \{\mathrm{p}^\mathsf{c}+1,\ldots,\mathrm{q}^\mathsf{c}\}$ with $\alpha_i^\mathsf{c} \in \mathtt{V}$, B_i^c occurs in T_i;
- for all $i \in \{\mathrm{p}^\mathsf{c}+1,\ldots,\mathrm{q}^\mathsf{c}\}$ with $\alpha_i^\mathsf{c} \in \mathtt{V}$, $\mathrm{Pos}(\mathsf{B}_i^\mathsf{c}, T_i) \subseteq \mathrm{Pos}^+(T_i)$;
- $\sigma^\mathsf{c} \in \overline{\mathtt{A}}$;
- 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 monotone wrt. every α_i^c);
- for all $i \in \{1,\ldots,\mathrm{p}^\mathsf{c}\}$, $\alpha_i^\mathsf{c} <_\mathtt{A} \sigma^\mathsf{c}$ (σ^c is strictly extensive wrt. recursive arguments).

The semantics of these annotations is given by 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 $[\mathsf{B}_i^\mathsf{c} : \mathscr{S}^{\mathsf{B}_i^\mathsf{c}}]T_i$.

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 16 (Interpretation of size-annotated types) First, for each constructor c with $\overline{\Theta}(\mathsf{c})$ as in Definition 15, we define a size function Σ^c (see Definition 7) as follows:

$$\Sigma^\mathsf{c}(\vec{\mathfrak{a}}) = \begin{cases} 0 \text{ if } \sigma^\mathsf{c} = \infty \\ \sigma^\mathsf{c}\nu \text{ otherwise where } \nu(\alpha) = \begin{cases} \mathfrak{a}_i \text{ if } \alpha = \alpha_i^\mathsf{c} \text{ and all the } \alpha_i^\mathsf{c} \in \mathtt{V} \text{ are distinct} \\ \sup\{\mathfrak{a}_i \mid i \in \{1,\ldots,\mathrm{q}^\mathsf{c}\}, \alpha_i^\mathsf{c} \in \mathtt{V}\} \text{ otherwise} \end{cases} \end{cases}$$

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

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

Note that Σ^c is monotone wrt. every argument and strictly extensive wrt. recursive arguments since σ^c so is.

Note also that, by definition of sup, if α is distinct from every α_i^c, then $\nu(\alpha) = 0$.

In the successor algebra, a constructor c can always be annotated as in Definition 15 above by taking:

Example 2 (Canonical annotations in the successor algebra) The *canonical type of a constructor* c *in the successor algebra* is obtained by taking:

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

In this case, we get $\Sigma^{\mathsf{c}}(\mathfrak{a}_1,\ldots,\mathfrak{a}_{\mathsf{q}^{\mathsf{c}}}) = \sup\{\mathfrak{a}_1+1,\ldots,\mathfrak{a}_{\mathsf{p}^{\mathsf{c}}}+1\}$, that is, the size is the constructor height, the size of a constant being 0.

For the constructors of the sort O of Howard's constructive ordinals, we get:

- $\mathsf{zero} : \mathsf{O}_\alpha$, $\sigma^{\mathsf{zero}} = \alpha$ and $\Sigma^{\mathsf{zero}} = 0$;
- $\mathsf{succ} : \mathsf{O}_\alpha \Rightarrow \mathsf{O}_{\mathtt{s}\alpha}$, $\mathsf{B}_1^{\mathsf{succ}} = \mathsf{O}$, $\alpha_1^{\mathsf{succ}} = \alpha$, $\sigma^{\mathsf{succ}} = \mathtt{s}\,\alpha$ and $\Sigma^{\mathsf{succ}}(\mathfrak{a}) = \mathfrak{a}+1$;
- $\mathsf{lim} : (\mathsf{N} \Rightarrow \mathsf{O}_\alpha) \Rightarrow \mathsf{O}_{\mathtt{s}\alpha}$, $\mathsf{B}_1^{\mathsf{lim}} = \mathsf{O}$, $\alpha_1^{\mathsf{lim}} = \alpha$, $\sigma^{\mathsf{lim}} = \mathtt{s}\,\alpha$ and $\Sigma^{\mathsf{lim}}(\mathfrak{a}) = \mathfrak{a}+1$.

Remark that we could have zero of size 2 by simply taking $\mathsf{zero} : \mathsf{N}_{\mathtt{s}(\mathtt{s}\,\alpha)}$ instead.

For the constructors of the sort T of binary trees with labels in a sort $\mathsf{B} <_{\mathbb{S}} \mathsf{T}$, we get:

- $\mathsf{leaf} : \mathsf{B} \Rightarrow \mathsf{T}_\alpha$, $\mathsf{B}_1^{\mathsf{leaf}} = \mathsf{B}$, $\alpha_1^{\mathsf{leaf}} = \infty$, $\sigma^{\mathsf{leaf}} = \alpha$, and $\Sigma^{\mathsf{leaf}}(\mathfrak{a}) = 0$;
- $\mathsf{node} : \mathsf{T}_\alpha \Rightarrow \mathsf{T}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{T}_{\mathtt{s}\alpha}$, $\mathsf{B}_1^{\mathsf{node}} = \mathsf{B}_2^{\mathsf{node}} = \mathsf{T}$, $\alpha_1^{\mathsf{node}} = \alpha_2^{\mathsf{node}} = \alpha$, $\sigma^{\mathsf{node}} = \mathtt{s}\,\alpha$ and $\Sigma^{\mathsf{node}}(\mathfrak{a},\mathfrak{b},\mathfrak{c}) = \sup\{\mathfrak{a}+1,\mathfrak{b}+1\}$.

Note that, in the successor algebra, constructors with at least two accessible arguments (*e.g.* node) cannot have functional annotated types (because there is only one non-nullary symbol, namely $\mathtt{s}$).

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 sizes are compared:

Definition 17 (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 every $\mathsf{f} : T_1 \Rightarrow \ldots \Rightarrow T_{\mathrm{r}^{\mathsf{f}}} \Rightarrow \mathsf{B}$:
  - a number q^{f} such that, for all $i \in \{1,\ldots,\mathrm{q}^{\mathsf{f}}\}$, T_i is a sort $\mathsf{B}_i^{\mathsf{f}}$ (the first q^{f} arguments of f are the arguments that will be taken into account for proving termination);
  - an annotated type $\overline{\Theta}(\mathsf{f}) = \overline{T_1} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}^{\mathsf{f}}}} \Rightarrow \mathsf{B}_{\sigma^{\mathsf{f}}}$ such that:
    - for all $i \in \{1,\ldots,\mathrm{q}^{\mathsf{f}}\}$, $\overline{T_i} = \mathrm{Annot}(\mathsf{B}_i^{\mathsf{f}}, \mathsf{B}_i^{\mathsf{f}}, \alpha_i^{\mathsf{f}})$;
    - for all $i \in \{\mathrm{q}^{\mathsf{f}}+1,\ldots,\mathrm{r}^{\mathsf{f}}\}$, $\overline{T_i} = T_i$;
    - $\vec{\alpha}^{\mathsf{f}}$ are distinct variables;
    - $\sigma^{\mathsf{f}} \in \overline{\mathtt{A}}$;
    - $\mathrm{Var}(\sigma^{\mathsf{f}}) \subseteq \{\vec{\alpha}^{\mathsf{f}}\}$;
  - for each $X \in \{\mathtt{A},\mathfrak{h}\}$, a set $\mathbb{D}_X^{\mathsf{f}}$, a quasi-order $\leq_X^{\mathsf{f}}$ on $\mathbb{D}_X^{\mathsf{f}}$, a well-founded relation $<_X^{\mathsf{f}} \subseteq \leq_X^{\mathsf{f}}$ and a map $\zeta_X : X^{\mathrm{q}^{\mathsf{f}}} \to \mathbb{D}_X^{\mathsf{f}}$ such that:
    - $(\mathbb{D}_X^{\mathsf{f}}, \leq_X^{\mathsf{f}}, <_X^{\mathsf{f}}) = (\mathbb{D}_X^{\mathsf{g}}, \leq_X^{\mathsf{g}}, <_X^{\mathsf{g}})$ 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 $\mu : \mathtt{V} \to \mathfrak{h}$;

– $\vec{a} <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{c}$ whenever $\vec{a} (\leq_{\mathtt{A}}^{\infty})_{\mathrm{prod}} \vec{b}$ and $\vec{b} <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{c}$, that is, $(\leq_{\mathtt{A}}^{\infty})_{\mathrm{prod}} \circ <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \subseteq <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}}$;
– $\vec{\mathfrak{a}} <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{\mathfrak{c}}$ whenever $\vec{\mathfrak{a}} <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \vec{\mathfrak{b}}$ and $\vec{\mathfrak{b}} \leq_{\mathrm{prod}} \vec{\mathfrak{c}}$, that is, $<_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \circ \leq_{\mathrm{prod}} \subseteq <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}}$;

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

The condition $<_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}} \circ \leq_{\mathrm{prod}} \subseteq <_{\mathfrak{h}}^{\mathsf{g},\mathsf{f}}$ is used in Theorem 1 (in the case (app-decr)). On the other hand, the condition $(\leq_{\mathtt{A}}^{\infty})_{\mathrm{prod}} \circ <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \subseteq <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}}$ is only used in Lemma 22. Note that, because $<_{\mathtt{A}}^{\mathsf{g},\mathsf{f}}$ is only defined on terms of $\mathtt{A}$, if $\vec{a} (\leq_{\mathtt{A}}^{\infty})_{\mathrm{prod}} \vec{b}$ and $\vec{b} <_{\mathtt{A}}^{\mathsf{g},\mathsf{f}} \vec{c}$, then $\vec{a}$ must be in $\mathtt{A}$ too since, by definition, $a \leq_{\mathtt{A}}^{\infty} b$ iff $a \leq_{\mathtt{A}} b$ or $b = \infty$.

In the following, we may drop the exponent f when there is no ambiguity.

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

We therefore define a quasi-ordering on pairs (f,φ) as follows. Given $h \in \mathbb{V} \cup \mathbb{C} \cup \mathbb{F}$, $\psi : \{\vec{\alpha}^h\} \to \overline{\mathtt{A}}$ (with $\{\vec{\alpha}^h\} = \emptyset$ if $h \in \mathbb{V} \cup \mathbb{C}$), $\mathsf{f} \in \mathbb{F}$, $\varphi : \{\vec{\alpha}^{\mathsf{f}}\} \to \overline{\mathtt{A}}$, let

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

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

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_X^{f}, 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 7, we use a different function. When ζ_X^{f} is the identity, one can for instance take for $\leq_{\mathtt{A}}^{\mathsf{f}}$ ($\leq_{\mathfrak{h}}^{\mathsf{f}}$ resp.) the lexicographic or multiset extension (Dershowitz & Manna, 1979) of $\leq_{\mathtt{A}}$ ($\leq$ resp.), or some combination thereof, for which one can easily prove the compatibility of $\leq_{\mathtt{A}}^{\mathsf{f}}$ ($\leq_{\mathfrak{h}}^{\mathsf{f}}$ resp.) with $\leq_{\mathtt{A}}^{\infty}$ ($\leq$ resp.). Indeed, we have $(\leq_{\mathtt{A}}^{\infty})_{\mathrm{prod}} \circ (<_{\mathtt{A}})_{\mathrm{lex}} \subseteq (<_{\mathtt{A}})_{\mathrm{lex}}$ since $\leq_{\mathtt{A}}^{\infty} \circ \leq_{\mathtt{A}} \subseteq \leq_{\mathtt{A}}$, and $<_{\mathrm{lex}} \circ \leq_{\mathrm{prod}} \subseteq <_{\mathrm{lex}}$.

We can now state our general termination theorem. In Section 6, we will provide many examples of rewrite systems whose termination is implied by it.

Theorem 1 (Termination criterion) Assume that constructor types are annotated as in Definition 15. By Definition 16 and 8, this provides us with a size function Σ and a stratification $\mathscr{S}$. Assume moreover that $\to_{\mathscr{R}}$ is finitely branching and no σ^{c} can be interpreted by an infinite limit ordinal.

Then, the relation $\to = \to_\beta \cup \to_{\mathscr{R}}$ terminates on the set $\mathbb{T}$ of well-typed terms if, for each rule $l \to r \in \mathscr{R} \subseteq \mathbb{T}^2$, l is of the form $\mathsf{f}\vec{l}$, the type of f is annotated as in Definition 17, $|\vec{l}| \geq \mathrm{q}^{\mathsf{f}}$ 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 such that x occurs in l_{k^x}, a sort B^x occurring only positively in $|U|$ and a size variable α^x such that $U = \mathrm{Annot}(|U|,\mathsf{B}^x,\alpha^x)$, indicating how to measure the size of x;[6]
– *finite* symbolic size upper bounds $\varphi : \{\vec{\alpha}^{\mathsf{f}}\} \to \mathtt{A}$ for $l_1,\ldots,l_{\mathrm{q}}$;

such that:

[6] Note that, if we do not care about the size of x, or if no sort occurs only positively in U, then we can always take for B^x any sort *not occurring* in U.

Fig. 3. Computability closure of (f,φ)

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

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

Fig. 4. Subtyping rules

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

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

- **Monotony.** For all $i\in\{1,\dots,\mathrm{q}^{\mathsf{f}}\}$, $\mathrm{Pos}(\alpha^{\mathsf{f}}_i,\sigma^{\mathsf{f}})\subseteq\mathrm{Pos}^{+}(\sigma^{\mathsf{f}})$;
- **Accessibility.** For every $(x,U)\in\Gamma$, one of the following holds:
  - $x=l_{k^x}$ and $U=\overline{T_{k^x}}\varphi$,
  - T_{k^x} is a sort and $(x,|U|,\mathsf{B}^x)\trianglelefteq_{\mathrm{a}}(l_{k^x},T_{k^x},T_{k^x})$;
- **Minimality.**[7] For all substitutions θ with $\vec{l}\theta\in\vec{T}$, there exists a valuation ν such that:
  - for all $(x,U)\in\Gamma$, $o_{[\mathsf{B}^x:\mathscr{S}^{\mathsf{B}^x}]|U|}(x\theta)\leq\alpha^x\nu$,
  - for all $i\in\{1,\dots,\mathrm{q}^{\mathsf{f}}\}$, $\alpha^{\mathsf{f}}_i\varphi\nu=o_{\mathscr{S}^{\mathsf{B}_i}}(l_i\theta)$;
- **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.

Proof. **Computability of constructors.** We first prove that, for all $(\mathsf{c},\mu,\vec{t})$ with $\Theta(\mathsf{c})=T_1\Rightarrow\ldots\Rightarrow T_{\mathrm{r}}\Rightarrow\mathsf{B}$, $\overline{\Theta}(\mathsf{c})=\overline{T_1}\Rightarrow\ldots\Rightarrow\overline{T_{\mathrm{r}}}\Rightarrow\mathsf{B}_\sigma$ as in Definition 15 (we drop the c's in exponents), $|\vec{t}|=\mathrm{r}$ and $(\forall i)t_i\in\overline{T_i}\mu$, we have $\mathsf{c}\vec{t}\in\mathsf{B}_\sigma\mu$. 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, by Proposition 1, for every $i\in\{1,\dots,\mathrm{q}\}$, we have $\overline{T_i}\mu\subseteq T_i$ since $\overline{T_i}=\mathrm{Annot}(T_i,\mathsf{B}_i,\alpha_i)$ and $\mathrm{Pos}(\mathsf{B}_i,T_i)\subseteq\mathrm{Pos}^{+}(T_i)$. Therefore, $\mathsf{c}\vec{t}\in\mathsf{B}$.

[7] Lemma 17 provides a syntactic condition for checking minimality in the successor algebra.

Now, if $\sigma = \infty$, then we are done. Otherwise, we are left to prove that $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{t}) \leq \sigma\mu$. By Corollary 2, $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\vec{t}) = \Sigma(o_{\mathscr{J}}(\vec{t}))$. By definition, $\Sigma(o_{\mathscr{J}}(\vec{t})) = \sigma\nu$ where ν is defined in Definition 16. Since σ is monotone and $\mathrm{Var}(\sigma) \subseteq \{\vec{\alpha}\}$, it suffices to prove that, for all i such that $\alpha_i \in \mathtt{V}$, $\alpha_i\nu \leq \alpha_i\mu$. If all the $\alpha_i \in \mathtt{V}$ are distinct, then $\alpha_i\nu = o_{\mathscr{J}^i}(t_i) \leq \alpha_i\mu$ since $t_i \in \overline{T_i}\mu$ and α_i occurs only positively in $\overline{T_i}$. Otherwise, all the $\alpha_i \in \mathtt{V}$ are equal. If there is no $\alpha_i \in \mathtt{V}$, then the property holds trivially. Otherwise, all the $\alpha_i \in \mathtt{V}$ are equal to some variable α and $\alpha\nu = \sup\{o_{\mathscr{J}^i}(t_i) \mid \alpha_i = \alpha\} \leq \alpha\mu$ since, for all i such that $\alpha_i = \alpha$, $t_i \in \overline{T_i}\mu$ and α occurs only positively in $\overline{T_i}$.

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 17 (we drop the f's in exponents), $|\vec{t}| = \mathrm{r}$ and $(\forall i) t_i \in \overline{T_i}\mu$, 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 u such that $\mathsf{f}\vec{t} \to u$, we have $u \in \mathsf{B}_\sigma\mu$. There are two cases:

(a) $u = \mathsf{f}\vec{u}$ and $\vec{t} \to_{\mathrm{prod}} \vec{u}$. Since computability is preserved by reduction, $(\forall i) u_i \in \overline{T_i}\mu$. Therefore, by induction hypothesis (1), $u \in \mathsf{B}_\sigma\mu$.

(b) $\vec{t} = \vec{l}\theta\vec{u}$, $\mathsf{f}\vec{l} \to r \in \mathscr{R}$ and $u = r\theta\vec{u}$. Let ν be the valuation given by minimality. For all $i \leq \mathrm{q}$, $\alpha_i\varphi\nu = o_{\mathscr{J}^{\mathsf{B}_i}}(l_i\theta)$. Since $l_i\theta \in \overline{T_i}\mu$ and $T_i = \mathsf{B}_{i\alpha_i}$, we have $\varphi\nu \leq \mu$ (*).

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

– (app-decr) By induction hypothesis (2), $w_i\theta \in V_i\psi\nu$. There are 3 cases:

· $h \in \mathbb{V}$. Then, $h\theta\vec{w}\theta \in V\psi\nu$ since $\psi = \emptyset$ and $h\theta \in (\vec{V} \Rightarrow V)\nu$ by assumption.

· $h \in \mathbb{C}$ and $V = \vec{U} \Rightarrow \mathsf{C}_\sigma$. For all $\vec{u} \in \vec{U}\psi\nu$, we have $h\vec{w}\theta\vec{u} \in \mathsf{C}_\sigma\psi\nu$ by computability of constructors. Therefore, by Definition 2, $h\vec{w}\theta \in V\psi\nu$.

· $h \in \mathbb{F}$ and $V = \vec{U} \Rightarrow \mathsf{C}_\sigma$. Since $(h,\psi) <_\mathtt{A} (\mathsf{f},\varphi)$ and $\vec{a}\nu <^{h,\mathsf{f}}_\mathfrak{h} \vec{b}\nu$ whenever $\vec{a} <^{h,\mathsf{f}}_\mathtt{A} \vec{b}$, we have $(h,\psi\nu) <_\mathfrak{h} (\mathsf{f},\varphi\nu)$. Since $\varphi\nu \leq \mu$ and $<^{h,\mathsf{f}}_\mathfrak{h} \circ \leq_{\mathrm{prod}} \subseteq <^{h,\mathsf{f}}_\mathfrak{h}$, we have $(h,\psi\nu) <_\mathfrak{h} (\mathsf{f},\mu)$. Hence, for all $\vec{u} \in \vec{U}\psi\nu$, we have $h\vec{w}\theta\vec{u} \in \mathsf{C}_\sigma\psi\nu$ by induction hypothesis (1). Therefore, by Definition 2, $h\vec{w}\theta \in V\psi\nu$.

– (lam) Wlog. we can assume that $x \notin \mathrm{dom}(\theta) \cup \mathrm{FV}(\theta)$. We have $(\lambda x^U w)\theta = \lambda x^U(w\theta) \in U\nu \Rightarrow V\nu$ because, for all $u \in U\nu$, $(w\theta)\{(x,u)\} \in V\nu$ (cf. remarks after Definition 1) since $(w\theta)\{(x,u)\} = w\theta'$ where $\theta' = \theta \cup \{(x,u)\}$ and $w\theta' \in V\nu$ 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 $\mathsf{B}_a\nu = \mathscr{S}^{\mathsf{B}}_{a\nu} \subseteq \mathscr{S}^{\mathsf{B}}_{b\nu} = \mathsf{B}_b\nu$ since $\mathscr{S}^{\mathsf{B}}$ 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), $V\nu \subseteq V\nu'$. Thus, $tu' \in V'\nu$.

· (refl) Immediate.

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

(ii) **Computability of the matching substitution:** for all $(x,U) \in \Gamma$, $x\theta \in U\nu$. By assumption, there is k such that x occurs in l_k, and $l_k\theta \in \overline{T_k}\mu$. After the accessibility condition, there are two cases:

– $x = l_k$ and $U = \overline{T_k}\varphi$. If $k > q$, then $\overline{T_k} = T_k$ and $\overline{T_k}\mu = U\nu$. Therefore, $x\theta \in U\nu$ since $l_k\theta \in \overline{T_k}\mu$. If $k \leq q$, then $\overline{T_k} = \mathsf{B}_{\alpha_k}$ for some sort B. By minimality, $\alpha_k\varphi\nu = o_{\mathscr{S}^{\mathsf{B}}}(l_k\theta)$. Therefore, $x\theta \in U\nu$ since $U = \mathsf{B}_{\alpha_k}\varphi$.

– T_k is a sort and $(x, |U|, \mathsf{B}^x) \trianglelefteq_{\mathrm{a}} (l_k, T_k, T_k)$. By Lemma 13, $x\theta \in |U|$ since, by assumption, $l_k\theta \in T_k$. By assumption, $U = \mathrm{Annot}(|U|, \mathsf{B}^x, \alpha^x)$ and $\mathrm{Pos}(\mathsf{B}^x, |U|) \subseteq \mathrm{Pos}^+(|U|)$. By minimality, $o_{[\mathsf{B}^x:\mathscr{S}^{\mathsf{B}^x}]|U|}(x\theta) \leq \alpha^x\nu$. Therefore, $x\theta \in U\nu$.

(iii) We can now end the proof that $u \in \mathsf{B}_\sigma\mu$. Since $\Gamma \vdash^{\mathsf{f}}_{\varphi} r : V\varphi$ with $V = \overline{T_{|\vec{l}|+1}} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}}} \Rightarrow \mathsf{B}_\sigma$, and $x\theta \in U\nu$ whenever $(x, U) \in \Gamma$ by (ii), we have $r\theta \in V\varphi\nu$ by (i). Hence, $u = r\theta\vec{u} \in \mathsf{B}_\sigma\varphi\nu$. We now prove that $\mathsf{B}_\sigma\varphi\nu \subseteq \mathsf{B}_\sigma\mu$. If $\sigma = \infty$, then $\mathsf{B}_\sigma\varphi\nu = \mathsf{B}_\sigma\mu$. Otherwise, we have $\varphi\sigma \neq \infty$ since $\varphi : \{\vec{\alpha}\} \to \mathtt{A}$. Moreover, we have seen in (ii) that, for all $k \leq q$, $\overline{T_k} = \mathsf{B}_{\alpha_k}$ for some sort B and $\alpha_k\varphi\nu = o_{\mathscr{S}^{\mathsf{B}}}(l_k\theta)$. Since $l_k\theta \in \overline{T_k}\mu$, $\alpha_k\varphi\nu \leq \alpha_k\mu$. Now, by monotony, $\mathrm{Pos}(\alpha_k, \sigma) \subseteq \mathrm{Pos}^+(\sigma)$. Therefore, by Proposition 1, $\mathsf{B}_\sigma\varphi\nu \subseteq \mathsf{B}_\sigma\mu$.

Computability of well-typed terms. Now, it is easy to prove that every well-typed term is computable by proceeding as for the correctness of the computability closure: if $\Gamma \vdash t : T$ and $x\theta \in U$ whenever $(x, U) \in \Gamma$, then $t\theta \in T$. 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 $\overline{\Theta}(\mathsf{f}) = \overline{T_1} \Rightarrow \ldots \Rightarrow \overline{T_{\mathrm{r}}} \Rightarrow \mathsf{B}_\sigma$. After Definition 2, $\mathsf{f} \in \Theta(\mathsf{f})$ iff, for all $\vec{t} \in \vec{T}$ such that $|\vec{t}| = \mathrm{r}$, $\mathsf{f}\vec{t} \in \mathsf{B}$. By assumption, for all $i \in \{1, \ldots, \mathrm{q}\}$, T_i is a sort B_i. Let μ be the valuation mapping, for every $i \in \{1, \ldots, \mathrm{q}\}$, α_i to the smallest ordinal $\mathfrak{h}_i < \mathfrak{h}$ such that $\mathscr{S}^{\mathsf{B}_i}_{\mathfrak{h}_i} = \mathsf{B}_i$. Then, $t_i \in \overline{\mathsf{B}_i}\mu$ and, by computability of function symbols, $\mathsf{f}\vec{t} \in \mathsf{B}_\sigma\mu \subseteq \mathsf{B}$. Finally, we conclude by noting that the identity substitution is computable (cf. remark after Definition 1). ■

It is worth remarking that this criterion is modular since the above conditions are for each rule. Hence, if both $\mathscr{R}_1$ and $\mathscr{R}_2$ satisfy the criterion with the same parameters, then $\mathscr{R}_1 \cup \mathscr{R}_2$ satisfies the criterion too.

We now discuss each condition in turn.

Accessibility. The accessibility is easy to check. 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 (Lemma 13). The accessibility condition ensures that each free variable x of the right hand-side is accessible in some l_i. Hence, every instance of x is computable if the arguments of f so are. Now, when x is accessible in a termination argument ($k^x \leq \mathrm{q}^{\mathsf{f}}$), there must be a sort B^x with respect to which the size of the instances of x are measured. Since x can be instantiated by terms of any size, the type of x should be of the form $\mathrm{Annot}(|U|, \mathsf{B}^x, \alpha^x)$, that is, every occurrence of B^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. This subject-reduction property is required since the interpretation of a type has to be stable by reduction. So, there should be no rule $\mathsf{f}\vec{l} \to r$ such that the size of r is strictly bigger than the size of $\mathsf{f}\vec{l}$. Second, by (app-decr), in every function call $h\vec{t}$, the symbolic 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.

In (app-decr), ψ is any size substitution of the size variables of $\vec{V}$. This rule works like the rule for type instantiation in Hindley-Milner type system (Hindley, 1969; Milner, 1978) 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. This means that, 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 and function symbols. 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, are 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 $<_{\mathtt{A}}$.

In the examples below, we will however use (var), (cons) and (prec) to denote the rule (app-decr) when h is variable, a constructor or a function symbol smaller than f respectively.

Note that the typability of r may require two variables x and y to have the same size over-approximation, that is, to have $\alpha^x = \alpha^y$. This will always be the case in the successor algebra when x and y are two recursive arguments of a constructor because, in this algebra, the types of constructor arguments are annotated by the same variable. For instance, if x and y are the first two arguments of $\mathsf{node} : \mathsf{T}_\alpha \Rightarrow \mathsf{T}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{T}_{\mathsf{s}\alpha}$, we must have $\alpha^x = \alpha^y$.

Note also 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 $\mathsf{f}(\mathsf{b}x_1(\mathsf{c}x_2)) \to \mathsf{f}(\mathsf{b}x_1x_2)$. Indeed, in this case, we can have $o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{b}x_1\theta\, x_2\theta) = o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{b}x_1\theta\,(\mathsf{c}x_2\theta))$ if $o(x_2\theta) < o(x_1\theta)$: the height is not a decreasing measure in this case.

The relation $\vdash^{\mathsf{f}}_{\varphi}$ is similar to the notion of computability closure introduced in (Blanqui *et al.*, 2002; Blanqui, 2016) 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 (Manna & Ness, 1970; van de Pol, 1996; Hamana, 2006), semantic labeling (Zantema, 1995; Hamana, 2007) or the notion of size-change principle (Lee *et al.*, 2001; Hyvernat, 2014). Now, as remarked in (Blanqui, 2006a; Kusakari & Sakai, 2007), the notion of computability closure itself has strong connections with the notion of dependency pair (Arts & Giesl, 2000). It is also a tool for defining and

strengthening the higher-order recursive path ordering (Blanqui, 2006b; Jouannaud & Rubio, 2007; Blanqui *et al.*, 2015). Finally, some relations between these notions have been formally established: size-change principle and dependency pairs (Thiemann & Giesl, 2005), semantic labeling and recursive path ordering (Kamin & Lévy, 1980), dependency pairs and recursive path ordering (Dershowitz, 2013), and size-based termination and semantic labeling (Blanqui & Roux, 2009).

The decidability of $\vdash^{\mathsf{f}}_{\varphi}$ will be studied in Section 7 and following.

Monotony. The monotony condition is easy to check. It 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 $\sigma^{\mathsf{f}} = \infty$. It is also satisfied if $\mathtt{A}$ is monotone. This condition also appears in (Abel, 2004; Barthe *et al.*, 2004). 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 defined by the rules of Figure 1 and assume that $\mathsf{sub} : \mathsf{N}_{\alpha} \Rightarrow \mathsf{N}_{\beta} \Rightarrow \mathsf{N}_{\alpha-\beta}$ in the size algebra with $\leq_{\mathtt{A}} = \leq_{\mathrm{ext}}$ and $\mathtt{0}$, $\mathtt{s}$ and $-$ interpreted by 0, successor and minus respectively. Then, given $\mathsf{f} : \mathsf{N}_{\alpha} \Rightarrow \mathsf{N}$ with $\mathsf{sub} <_{\mathbb{F}} \mathsf{f}$, the rule $\mathsf{f}\ (\mathtt{s}\ x) \to \mathsf{f}\ (\mathsf{sub}\ (\mathtt{s}\ x)\ x)$ satisfies the other conditions. Indeed, take $\Gamma = [x : \mathsf{N}_x]$ and $\varphi = \{(\alpha, \mathtt{s}\,x)\}$. By (var), $\vdash^{\mathsf{f}}_{\varphi} x : \mathsf{N}_x$. By (cons), $\vdash^{\mathsf{f}}_{\varphi} \mathtt{s}\,x : \mathsf{N}_{\mathtt{s}\,x}$. By (sub), $\vdash^{\mathsf{f}}_{\varphi} x : \mathsf{N}_{\mathtt{s}\,x}$. By (prec), $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{sub}\ (\mathtt{s}\ x)\ x : \mathsf{N}_{\mathtt{s}\,x-\mathtt{s}\,x}$. By (sub), $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{sub}\ (\mathtt{s}\ x)\ x : \mathsf{N}_{\mathtt{0}}$ (while $o_{\mathscr{D}^{\mathsf{N}}}(\mathsf{sub}\ (\mathtt{s}\ x)\ x) > 0$!). Therefore, $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{f}\ (\mathsf{sub}\ (\mathtt{s}\ x)\ x) : \mathsf{N}$ since $\mathtt{0} <_{\mathtt{A}} \mathtt{s}\,x$, but the system does not terminate since $\mathsf{f}\ (\mathtt{s}\,x) \to \mathsf{f}\ (\mathsf{sub}\ (\mathtt{s}\,x)\ x) \to \mathsf{f}\ (\mathtt{s}\,x)$.

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 $\mathsf{f} : \mathsf{N}_{\alpha} \Rightarrow \mathsf{N}$ and $\Gamma = [x : \mathsf{N}_x]$. By taking $\alpha\varphi = \mathtt{s}\,x$, one has $\Gamma \vdash^{\mathsf{f}}_{\varphi} \mathsf{f}\ x : \mathsf{N}$ since $\Gamma \vdash^{\mathsf{f}}_{\varphi} x : \mathsf{N}_x$ and $x <_{\mathtt{A}} \mathtt{s}\ x$.

In Theorem 1, minimality is expressed in the most general way by using the interpretation of annotated types in computability predicates. With some acquaintance, it is not too difficult to check this condition by hand on simple systems as shown in Example 3. In fact, we think that it is always possible to find a minimal φ when the type of every constructor c is annotated in the max-successor algebra (extension of the successor algebra with a $\mathtt{max}$ operator) in the canonical way, that is, by taking $\sigma^{\mathsf{c}} \in \mathtt{V}$ if $\mathrm{p}^{\mathsf{c}} = 0$ and $\sigma^{\mathsf{c}} = \mathtt{s}(\mathtt{max}\alpha^{\mathsf{c}}_1 \dots \alpha^{\mathsf{c}}_{\mathrm{p}^{\mathsf{c}}})$ with distinct variables for $\alpha^{\mathsf{c}}_1, \dots, \alpha_{\mathrm{p}^{\mathsf{c}}}$ otherwise. However, in this paper, we want to focus on the successor algebra and, in this case, minimality may not be satisfiable whatever φ is. This is due to the fact that, in the successor algebra, one often needs to approximate the sizes of two distinct term variables by the same size variable. Indeed, in the successor algebra, there is no function symbol of arity ≥ 2. Hence, the annotated type of a binary constructor can only be of the form $\mathsf{B}_{\alpha} \Rightarrow \mathsf{B}_{\alpha} \Rightarrow \mathsf{B}_{\sigma}$ with the same size variable α for both arguments.

In the following section, we study in more details the size of constructor terms when the size is defined as the height like it is the case with the canonical annotations of constructor types in the successor and max-successor algebras. Then, we give a syntactic condition for minimality to be satisfied in the successor algebra.

5 Minimality property when the size is defined as the height

In this section, we provide sufficient conditions for the minimality condition of Theorem 1 to be satisfied when the notion of size is the height and the size of constants is 0, that is, when, for every constructor c, we have:

$$\Sigma^{\mathsf{c}}(\mathfrak{a}_1,\dots,\mathfrak{a}_{\mathsf{q}^{\mathsf{c}}}) = \sup\{\mathfrak{a}_1+1,\dots,\mathfrak{a}_{\mathsf{p}^{\mathsf{c}}}+1\}.$$

After Definition 16, this can be achieved in the successor algebra by taking the canonical annotation for constructor types (cf. Example 2).

To check the minimality condition, we need to know how the size of a term of the form $t\theta$ depends on the sizes of the subterms $x\theta$ where x is a variable of t. To this end, we introduce a number of definitions to express what are the subterms that contribute to the size of a term and how they contribute to it:

Definition 18 (Recursive subterms) Let $\mathbb{D}$ be the set of triples (u,U,k) made of a term u, a type U and a number $k \in \mathbb{N}$. Given a sort B and $(u,U,k) \in \mathbb{D}$, let

$$\mathrm{Sub}^1_{\mathsf{B}}(u,U,k) = \begin{cases} \{(u_i,U_i,k+1) \mid i \in \{1,\dots,\mathsf{p}^{\mathsf{c}}\}\} \\ \quad \text{if there is } (\mathsf{c},\vec{u},\vec{U}) \in \mathbb{C}^{\mathsf{B}} \text{ such that } u = \mathsf{c}\,\vec{u} \text{ and } U = \mathsf{B} \\ \emptyset \text{ otherwise} \end{cases}$$

Then, let $\to_{\mathsf{B}}$ be the relation on finite sets of triples such that $S \to_{\mathsf{B}} S'$ if there is $d \in S$ such that $\mathrm{Sub}^1_{\mathsf{B}}(d) \neq \emptyset$ and $S' = (S - \{d\}) \cup \mathrm{Sub}^1_{\mathsf{B}}(d)$ (we replace d by $\mathrm{Sub}^1_{\mathsf{B}}(d)$). We say that a set $S \subseteq \mathbb{D}$ is a *set of* B*-recursive subterms* of a term t if $\{(t,\mathsf{B},0)\} \to^*_{\mathsf{B}} S$.

For instance, if $\mathsf{a} : \mathsf{B}$, $\mathsf{c} : \mathsf{B} \Rightarrow \mathsf{B}$, $\mathsf{p}^{\mathsf{c}} = 1$, $\mathsf{b} : \mathsf{B} \Rightarrow \mathsf{B} \Rightarrow \mathsf{B}$, $\mathsf{p}^{\mathsf{b}} = 2$ and $t = \mathsf{b}(\mathsf{c}(\mathsf{ca}))x$, then $\{(t,\mathsf{B},0)\} \to_{\mathsf{B}} \{(\mathsf{c}(\mathsf{ca}),\mathsf{B},1),(x,\mathsf{B},1)\} \to_{\mathsf{B}} \{(\mathsf{ca},\mathsf{B},2),(x,\mathsf{B},1)\} \to_{\mathsf{B}} \{(\mathsf{a},\mathsf{B},3),(x,\mathsf{B},1)\}$.

Lemma 14 If S is a set of B-recursive subterms of $t \in \mathsf{B}$, then

$$o_{\mathscr{J}^{\mathsf{B}}}(t) = \sup\{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u)+k \mid (u,U,k) \in S\}.$$

Proof. Let $M(S) = \{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u)+k \mid (u,U,k) \in S\}$. The lemma trivially holds for $S = \{(t,\mathsf{B},0)\}$. Hence, if suffices to check that, if it holds for S and $S \to_{\mathsf{B}} S'$, then it holds for S' too. So, assume that there is $(\mathsf{c}\,\vec{u},\mathsf{B},k) \in S$ such that $\mathrm{Sub}^1_{\mathsf{B}}(\mathsf{c}\,\vec{u},\mathsf{B},k) \neq \emptyset$. Then, $M(S') = (M(S) - \{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u)+k\}) \cup \{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_{U_i}}(u_i)+k+1 \mid i \in I\}$ where $I = \{1,\dots,\mathsf{p}^{\mathsf{c}}\}$. But, by Corollary 2, $o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_{\mathsf{B}}}(\mathsf{c}\,\vec{u}) = o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\,\vec{u}) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c},1}}(u_1),\dots,o_{\mathscr{J}^{\mathsf{c},p^{\mathsf{c}}}}(u_{p^{\mathsf{c}}})) = \sup\{o_{\mathscr{J}^{\mathsf{c},i}}(u_i)+1 \mid i \in I\} = \sup\{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_{U_i}}(u_i)+1 \mid i \in I\}$. Therefore, $\sup M(S) = \sup M(S')$. ■

Lemma 15 If S is a set of B-recursive subterms of t and θ is a substitution, then $S\theta = \{(u\theta,U,k) \mid (u,U,k) \in S\}$ is a set of B-recursive subterms of $t\theta$.

Proof. The lemma holds for $S = \{(t,\mathsf{B},0)\}$. Hence, if suffices to check that, if it holds for S and $S \to_{\mathsf{B}} S'$, then it holds for S' too. But $\mathrm{Sub}^1_{\mathsf{B}}(u\theta,U,k) = \mathrm{Sub}^1_{\mathsf{B}}(u,U,k)\theta$. ■

Note that $\to_{\mathsf{B}}$ terminates (it acts on finite sets and replaces a term by smaller subterms) and is confluent (it is orthogonal). Hence, every finite set has a $\to_{\mathsf{B}}$-normal form.

Definition 19 (Simple terms) Given a sort B and a term t, let $\mathrm{Sub}_{\mathsf{B}}(t)$ be the $\to_{\mathsf{B}}$-normal form of $\{(t,\mathsf{B},0)\}$. A term t of sort B is *simple* if, for all $(u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)$, either $u \in \mathbb{V}$ or there is $(\mathsf{c},\vec{u},\vec{U}) \in \mathbb{C}^{\mathsf{B}}$ such that $u = \mathsf{c}\,\vec{u}$, $U = \mathsf{B}$ and $\mathrm{p}^{\mathsf{c}} = 0$ (c has no recursive argument).

Lemma 16 If t is a simple term of sort B and $t\theta \in \mathsf{B}$ then:

$$o_{\mathscr{J}^{\mathsf{B}}}(t\theta) = \sup(\{\mathrm{d}_{\mathsf{B}}(t)\} \cup \{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_V}(x\theta) + \mathrm{d}^x_{\mathsf{B}}(t) \mid (x,V) \in \mathrm{Var}_{\mathsf{B}}(t)\})$$

where:

- $\mathrm{Var}_{\mathsf{B}}(t) = \{(x,U) \mid \exists k, (x,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)\}$,
- $\mathrm{d}^x_{\mathsf{B}}(t) = \sup\{k \mid \exists U, (x,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)\}$,
- $\mathrm{d}_{\mathsf{B}}(t) = \sup\{k \mid \exists u, \exists U, (u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)\}$.

Proof. By Lemma 15, $\mathrm{Sub}_{\mathsf{B}}(t)\theta$ is a set of B-recursive subterms of $t\theta$. Hence, by Lemma 14, $o_{\mathscr{J}^{\mathsf{B}}}(t\theta) = \sup\{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u) + k \mid (u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)\theta\} = \sup\{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u\theta) + k \mid (u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)\}$. Let $(x,V) \in \mathrm{Var}_{\mathsf{B}}(t)$. Since t is well-typed, for all $(x,V') \in \mathrm{Var}_{\mathsf{B}}(t)$, we have $V' = V$. Hence, $\sup\{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u\theta) + k \mid (u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t), u = x\} = o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_V}(x\theta) + \mathrm{d}^x_{\mathsf{B}}(t)$. Let now $(u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t)$ with $u \notin \mathbb{V}$. Since t is simple, there is $(\mathsf{c},\vec{u},\vec{U}) \in \mathbb{C}^{\mathsf{B}}$ such that $u = \mathsf{c}\,\vec{u}$, $U = \mathsf{B}$ and $\mathrm{p}^{\mathsf{c}} = 0$. By Corollary 2, $o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(\mathsf{c}\,\vec{u}\theta) = o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}\,\vec{u}\theta) = \Sigma^{\mathsf{c}}(o_{\mathscr{J}^{\mathsf{c},1}}(u_1\theta),\ldots,o_{\mathscr{J}^{\mathsf{c},p^{\mathsf{c}}}}(u_{p^{\mathsf{c}}}\theta)) = \sup\{o_{\mathscr{J}^{\mathsf{c},i}}(u_i\theta) + 1 \mid i \in \{1,\ldots,\mathrm{p}^{\mathsf{c}}\}\} = 0$. Therefore, $\sup\{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_U}(u\theta) + k \mid (u,U,k) \in \mathrm{Sub}_{\mathsf{B}}(t), u \notin \mathbb{V}\} = \mathrm{d}_{\mathsf{B}}(t)$ and $o_{\mathscr{J}^{\mathsf{B}}}(t\theta) = \sup(\{\mathrm{d}_{\mathsf{B}}(t)\} \cup \{o_{[\mathsf{B}:\mathscr{J}^{\mathsf{B}}]_V}(x\theta) + \mathrm{d}^x_{\mathsf{B}}(t) \mid (x,V) \in \mathrm{Var}_{\mathsf{B}}(t)\}$. ■

To carry on with the previous example, $t = \mathsf{b}(\mathsf{c}(\mathsf{ca}))x$ is simple and we have $o_{\mathscr{J}^{\mathsf{B}}}(t\theta) = \sup\{o_{\mathscr{J}^{\mathsf{B}}}(\mathsf{c}(\mathsf{ca})) + 1, o_{\mathscr{J}^{\mathsf{B}}}(x\theta) + 1\} = \sup\{3, o_{\mathscr{J}^{\mathsf{B}}}(x\theta) + 1\} = \sup\{\mathrm{d}_{\mathsf{B}}(t), o_{\mathscr{J}^{\mathsf{B}}}(x\theta) + \mathrm{d}^x_{\mathsf{B}}(t)\}$.

Assume now that we are under the conditions of Theorem 1 for some rule $\mathsf{f}\,\vec{l} \to r \in \mathscr{R}$, typing environment $\Gamma = [x_1 : U_1, \ldots, x_n : U_n]$ and substitution $\varphi : \{\vec{\alpha}\} \to \mathtt{A}$. In particular:

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

with $\vec{\alpha}$ distinct variables, $\sigma \in \overline{\mathtt{A}}$ and $\mathrm{Var}(\sigma) \subseteq \{\vec{\alpha}\}$.

Assume moreover that, for all $j \in \{1,\ldots,\mathrm{q}\}$, l_j is a simple term of sort B_j and there are $n_j \in \mathbb{N}$ and $\gamma_j \in \mathtt{V}$ such that $\alpha_j\varphi = \mathtt{s}^{n_j}\gamma_j$.

Then, after Lemma 16, the minimality property is 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), there are $\mathfrak{b}_1,\ldots,\mathfrak{b}_n$ (for $\alpha^{x_1}\nu,\ldots,\alpha^{x_n}\nu$ respectively) and $\mathfrak{c}_1,\ldots,\mathfrak{c}_{\mathrm{q}}$ (for $\gamma_1\nu,\ldots,\gamma_{\mathrm{q}}\nu$ 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. $(\forall j)\ \mathfrak{a}_j \leq \mathfrak{b}_j$,
5. $(\forall j)\ \mathfrak{c}_j + n_j = \sup(\{\mathrm{d}_{\mathsf{B}_j}(l_j)\} \cup \{\mathfrak{a}_m + \mathrm{d}^{x_m}_{\mathsf{B}_j}(l_j) \mid x_m \in \mathrm{dom}(\mathrm{Sub}_{\mathsf{B}_j}(l_j))\}$.

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

One of the problems for these inequations to be satisfied is when two arguments of f, say l_1 and l_2, share some variable but have distinct sets of variables, or when shared

variables occur at different depths. Take for instance $l_1 = x_1$ and $l_2 = \mathsf{b}\,(\mathsf{c}\,x_1)\,(\mathsf{c}\,x_2)$ with constructors annotated in the canonical way in the successor algebra, that is, $\mathsf{c} : \mathsf{B}_\alpha \Rightarrow \mathsf{B}_{\mathsf{s}\alpha}$ and $\mathsf{b} : \mathsf{B}_\alpha \Rightarrow \mathsf{B}_\alpha \Rightarrow \mathsf{B}_{\mathsf{s}\alpha}$. Then, for having $\mathsf{b}\,x_1\,x_2$ in the right hand-side, we need to take $\alpha^{x_1} = \alpha^{x_2} = \gamma_1 = \gamma_2$. In this case, the minimality condition says that, for all $\mathfrak{a}_1, \mathfrak{a}_2$, there is $\mathfrak{b}$ such that $\mathfrak{a}_1 \leq \mathfrak{b}$, $\mathfrak{a}_2 \leq \mathfrak{b}$, $\mathfrak{a}_1 = \mathfrak{b} + n_1$ and $\sup\{\mathfrak{a}_1 + 2, \mathfrak{a}_2 + 2\} = \mathfrak{b} + n_2$, which is not possible. Take now $l_1 = \mathsf{b}\,(\mathsf{c}\,x_1)\,(\mathsf{c}\,x_2)$ and $l_2 = \mathsf{b}\,(\mathsf{c}\,(\mathsf{c}\,x_1))\,(\mathsf{c}\,x_2)$. Again, for having $\mathsf{b}\,x_1\,x_2$ in the right hand-side, we need to take $\alpha^{x_1} = \alpha^{x_2} = \gamma_1 = \gamma_2$. In this case, the minimality condition says that, for all $\mathfrak{a}_1, \mathfrak{a}_2$, there is $\mathfrak{b}$ such that $\mathfrak{a}_1 \leq \mathfrak{b}$, $\mathfrak{a}_2 \leq \mathfrak{b}$, $\sup\{\mathfrak{a}_1 + 2, \mathfrak{a}_2 + 2\} = \mathfrak{b} + n_1$ and $\sup\{\mathfrak{a}_1 + 3, \mathfrak{a}_2 + 2\} = \mathfrak{b} + n_2$, which is not possible either.

We now give sufficient conditions for the above set of inequations to be satisfied:

Lemma 17 Under the conditions of Theorem 1, assume that constructor types are annotated in the canonical way in the successor algebra (cf. Example 2). Then, the minimality property is satisfied if, for all $j \in \{1, \ldots, \mathrm{q}\}$:

(a) l_j is a simple term of sort B_j;
(b) there are $n_j \in \mathbb{N}$ and $\gamma_j \in \mathsf{V}$ such that $\alpha_j \varphi = \mathsf{s}^{n_j} \gamma_j$;
(c) $n_j \leq \inf(\mathrm{range}(D_j))$;
(d) for all $k \in \{1, \ldots, \mathrm{q}\}$, if $\gamma_j = \gamma_k$, then $n_j = n_k$, $\mathrm{d}_{\mathsf{B}_j}(l_j) = \mathrm{d}_{\mathsf{B}_k}(l_k)$ and $D_j = D_k$;
(e) for all $x \in \mathrm{dom}(\Gamma)$, if $\gamma_j = \alpha^x$ then $x \in \mathrm{dom}(D_j)$;

where $D_j = \{(x, \mathrm{d}^x_{\mathsf{B}_j}(l_j)) \mid x \in \mathrm{dom}(\mathrm{Sub}_{\mathsf{B}_j}(l_j))\}$, $\mathrm{Sub}_{\mathsf{B}}(l)$ is introduced in Definition 19, d_{B} and $\mathrm{d}^x_{\mathsf{B}}$ are defined in Lemma 16.

Proof. Let $\mathfrak{c}_i = \sup(\{\mathrm{d}_{\mathsf{B}_i}(l_i)\} \cup \{\mathfrak{a}_p + d \mid (x_p, d) \in D_i\}) - n_i$. It is well defined since $n_i \leq \inf(\mathrm{range}(D_i)) \leq \mathrm{d}_{\mathsf{B}_i}(l_i)$. 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 $\mathfrak{c}_j = \mathfrak{c}_k$ because $n_j = n_k$, $\mathrm{d}_{\mathsf{B}_j}(l_j) = \mathrm{d}_{\mathsf{B}_k}(l_k)$ and $D_j = D_k$. 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, $\mathfrak{a}_j \leq \mathfrak{b}_j$. Indeed, if $\alpha^{x_j} = \gamma_m$, then $\mathfrak{b}_j = \mathfrak{c}_m \geq \mathfrak{a}_j$ since $x_j \in \mathrm{dom}(D_m)$.
   Otherwise, $\mathfrak{b}_j = \sup\{\mathfrak{a}_p \mid \alpha^{x_p} = \alpha^{x_j}\} \geq \mathfrak{a}_j$.
5. For all j, $\mathfrak{c}_j + n_j = \sup(\{\mathrm{d}_{\mathsf{B}_j}(l_j)\} \cup \{\mathfrak{a}_m + \mathrm{d}^{x_m}_{\mathsf{B}_j}(l_j) \mid x_m \in \mathrm{dom}(\mathrm{Sub}_{\mathsf{B}_j}(l_j))\})$ by definition of $\mathfrak{c}_j$. ■

For instance, with the last rule of Figure 1, $\mathsf{div}\ (\mathsf{s}\,x)\ (\mathsf{s}\,y) \to \mathsf{s}\ (\mathsf{div}\ (\mathsf{sub}\,x\,y)\ (\mathsf{s}\,y))$, if we take $\mathsf{div} : \mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}_\alpha$, $\Gamma = [x : \mathsf{N}_x, y : \mathsf{N}_y]$, $\alpha^x = x$, $\alpha^y = y$ and $\varphi = \{(\alpha, \mathsf{s}\,x)\}$, we have $n_1 = 1$, $\gamma_1 = x = \alpha^x$ and $D_1 = \{(x, 1)\}$. So, the conditions above are satisfied.

On the contrary, if $l_1 = \mathsf{c}\,x_1$, $l_2 = \mathsf{b}\,(\mathsf{c}\,x_1)\,(\mathsf{c}\,x_2)$, $\alpha^{x_1} = \alpha^{x_2} = \gamma_1 = \gamma_2$, $n_1 = 1$ and $n_2 = 2$, then (d) is not satisfied because $\gamma_1 = \gamma_2$ but $D_1 = \{(x_1, 1)\}$ and $D_2 = \{(x_1, 2), (x_2, 2)\}$.

6 Examples

In this section, we show various examples whose termination can be established by using Theorem 1. In proofs of $\vdash^{\mathsf{f}}_{\varphi}$ judgments, (var), (cons) and (prec) will refer to the specialization of (app-decr) to variables, constructors and function symbols smaller than f respectively.

We will use the following sorts and constructors with $\mathsf{N} <_{\mathbb{S}} \mathsf{L}$ and $\mathsf{N} <_{\mathbb{S}} \mathsf{O}$:

- 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 $0 : \mathsf{N}$ 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 $\mathsf{cons} : \mathsf{L} \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}$[8];

Unless stated otherwise, we always use the successor algebra (Definition 12) and, for constructor types, the canonical annotations (Example 2).

Example 3 (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.

We take $\mathsf{sub} <_{\mathbb{F}} \mathsf{div}$. For annotated types, we take, for each $f \in \{\mathsf{sub}, \mathsf{div}\}$, $\overline{\Theta}(f) = \mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}_\alpha$, $\mathsf{q}^f = 1$, $\mathsf{B}^f_1 = \mathsf{N}$, $\alpha^f_1 = \alpha$, which expresses the fact that these functions are not size-increasing. And, for ζ^{sub}_X and ζ^{div}_X, we take the identity.

We now detail the conditions of Theorem 1 for each rule in turn (monotony is trivial).

1. $\mathsf{sub}\ x\ 0 \to x$. Take $\Gamma = [x : \mathsf{N}_x]$, $k^x = 1$, $\mathsf{B}^x = \mathsf{N}$, $\alpha^x = x$ and $\varphi = \{(\alpha, x)\}$. Then, $\mathsf{N}_x = \mathrm{Annot}(\mathsf{N}, \mathsf{B}^x, \alpha^x)$ and:
   - Accessibility. x is accessible since $x = l_{k^x}$ and $\mathsf{N}_x = \mathsf{N}_\alpha\varphi$.
   - Minimality. One could use Lemma 17. We give a direct proof instead. Let θ be such that $x\theta \in \mathsf{N}$. We have to prove that there exists ν such that $o_{\mathscr{J}^{\mathsf{N}}}(x\theta) \leq \alpha^x\nu$ and $\alpha\varphi\nu = o_{\mathscr{J}^{\mathsf{N}}}(x\theta)$. It suffices to take $\nu(x) = o_{\mathscr{J}^{\mathsf{N}}}(x\theta)$.
   - Subject-reduction. By (var), $\vdash^{\mathsf{sub}}_{\varphi} x : \mathsf{N}_x = \mathsf{N}_\alpha\varphi$.
2. $\mathsf{sub}\ 0\ y \to 0$. Take $\Gamma = \varphi = \emptyset$. Then:
   - Minimality. Let θ be such that $y\theta \in \mathsf{N}$. We have to prove that there exists ν such that $\alpha\varphi\nu = o_{\mathscr{J}^{\mathsf{N}}}(0)$. It suffices to take $\nu(\alpha) = o_{\mathscr{J}^{\mathsf{N}}}(0)$.
   - Subject-reduction. By (cons), $\vdash^{\mathsf{sub}}_{\varphi} 0 : \mathsf{N}_\alpha = \mathsf{N}_\alpha\varphi$.
3. $\mathsf{sub}\ (\mathsf{s}\,x)\ (\mathsf{s}\,y) \to \mathsf{sub}\,x\,y$. Take $\Gamma = [x : \mathsf{N}_x, y : \mathsf{N}_y]$, $k^x = 1$, $\mathsf{B}^x = \mathsf{N}$, $\alpha^x = x$, $k^y = 2$, $\mathsf{B}^y = \mathsf{N}$, $\alpha^y = y$, $\varphi = \{(\alpha, \mathsf{s}\,x)\}$. Then, $\mathsf{N}_x = \mathrm{Annot}(\mathsf{N}, \mathsf{B}^x, \alpha^x)$, $\mathsf{N}_y = \mathrm{Annot}(\mathsf{N}, \mathsf{B}^y, \alpha^y)$ and:
   - Accessibility. x is accessible since $(x, \mathsf{N}, \mathsf{N}) \trianglelefteq_{\mathrm{a}} (l_{k^x}, \mathsf{N}, \mathsf{N})$. y is accessible since $(y, \mathsf{N}, \mathsf{N}) \trianglelefteq_{\mathrm{a}} (l_{k^y}, \mathsf{N}, \mathsf{N})$.
   - Minimality. Let θ be such that $\mathsf{s}x\theta \in \mathsf{N}$ and $\mathsf{s}y\theta \in \mathsf{N}$. We have to prove that there exists ν such that $o_{\mathscr{J}^{\mathsf{N}}}(x\theta) \leq \alpha^x\nu$, $o_{\mathscr{J}^{\mathsf{N}}}(y\theta) \leq \alpha^y\nu$ and $\alpha\varphi\nu = o_{\mathscr{J}^{\mathsf{N}}}(\mathsf{s}x\theta) = o_{\mathscr{J}^{\mathsf{N}}}(x\theta) + 1$. It suffices to take $\nu(x) = o_{\mathscr{J}^{\mathsf{N}}}(x\theta)$ and $\nu(y) = o_{\mathscr{J}^{\mathsf{N}}}(y\theta)$.
   - Subject-reduction. Let $\vdash = \vdash^{\mathsf{sub}}_{\varphi}$. By (var), $\vdash x : \mathsf{N}_x$ and $\vdash y : \mathsf{N}_y$. By (app-decr), $\vdash \mathsf{sub}xy : \mathsf{N}_x$ since $x <_{\mathtt{A}} \mathtt{s}x$. Therefore, by (sub), $\vdash \mathsf{sub}xy : \mathsf{N}_{\mathtt{s}x} = \mathsf{N}_\alpha\varphi$.

[8] We permuted the usual order of the arguments of cons so that its type conforms to Definition 4.

4. $\mathsf{div}\ 0\ (\mathsf{s}\ y) \to 0$. Like for rule (2).
5. $\mathsf{div}\ (\mathsf{s}\ x)\ (\mathsf{s}\ y) \to \mathsf{s}\ (\mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y))$. Take $\Gamma = [x : \mathsf{N}_x, y : \mathsf{N}_y]$, $k^x = 1$, $\mathsf{B}^x = \mathsf{N}$, $\alpha^x = x$, $k^y = 2$, $\mathsf{B}^y = \mathsf{N}$, $\alpha^y = y$ and $\varphi = \{(\alpha, \mathsf{s}\,x)\}$. Then, $\mathsf{N}_x = \mathrm{Annot}(\mathsf{N}, \mathsf{B}^x, \alpha^x)$, $\mathsf{N}_y = \mathrm{Annot}(\mathsf{N}, \mathsf{B}^y, \alpha^y)$ and:
   - Accessibility and minimality like for rule (3).
   - Subject-reduction. Let $\vdash = \vdash_\varphi^{\mathsf{div}}$. By (var), $\Gamma \vdash x : \mathsf{N}_x$ and $\Gamma \vdash y : \mathsf{N}_y$. By (prec), $\Gamma \vdash \mathsf{sub}\ x\ y : \mathsf{N}_x$. By (cons), $\Gamma \vdash \mathsf{s}\ y : \mathsf{N}_{\mathsf{s}\,y}$. 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 4 (Map and filter) Consider the function symbols $\mathsf{map} : \mathsf{L} \Rightarrow (\mathsf{N} \Rightarrow \mathsf{N}) \Rightarrow \mathsf{L}$,[9] $\mathsf{if} : \mathsf{L} \Rightarrow \mathsf{L} \Rightarrow \mathsf{B} \Rightarrow \mathsf{L}$ and $\mathsf{filter} : \mathsf{L} \Rightarrow (\mathsf{N} \Rightarrow \mathsf{B}) \Rightarrow \mathsf{L}$ defined by the rules:

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

For annotated types, we could take in the successor algebra, $\mathsf{map} : \mathsf{L}_\alpha \Rightarrow (\mathsf{N} \Rightarrow \mathsf{N}) \Rightarrow \mathsf{L}_\alpha$, $\mathrm{q}^{\mathsf{map}} = 1$, $\mathsf{B}_1^{\mathsf{map}} = \mathsf{L}$, $\alpha_1^{\mathsf{map}} = \alpha$, $\mathsf{if} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{B} \Rightarrow \mathsf{L}_\alpha$, $\mathrm{q}^{\mathsf{if}} = 2$, $\mathsf{B}_1^{\mathsf{if}} = \mathsf{B}_2^{\mathsf{if}} = \mathsf{L}$, $\alpha_1^{\mathsf{f}} = \alpha_2^{\mathsf{f}} = \alpha$, $\mathsf{filter} : \mathsf{L}_\alpha \Rightarrow (\mathsf{N} \Rightarrow \mathsf{B}) \Rightarrow \mathsf{L}_\alpha$ and $\mathrm{q}^{\mathsf{filter}} = 1$, expressing the fact that these functions are not size-increasing.

Unfortunately, the annotated type of if does not satisfy the conditions of Definition 17 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:

1. Annotate if in the max-successor algebra by taking $\mathsf{if} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\beta \Rightarrow \mathsf{B} \Rightarrow \mathsf{L}_{\mathtt{max}\alpha\beta}$.
2. 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:

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

One can easily check the conditions on annotated types and the monotony condition.

For the other conditions, we only detail the case of the last rule of filter by taking $\Gamma = [f : \mathsf{N} \Rightarrow \mathsf{B}, x : \mathsf{N}, l : \mathsf{L}_l]$, $\varphi = \{(\alpha, \mathsf{s}\,l)\}$, $k^f = 2$, any sort distinct from N and B for B^f (we do not care about the size of f), $\alpha^f = f$, $k^x = 1$, any sort distinct from N for B^x (we do not care about the size of x), $\alpha^x = x$, $k^l = 1$, $\mathsf{B}^l = \mathsf{L}$, $\alpha^l = l$, $\mathsf{newif} <_{\mathbb{F}} \mathsf{filter}$, $\mathsf{cond} <_{\mathbb{F}} \mathsf{filter}$ and the identity for ζ^{filter}.

One can easily check the accessibility and minimality conditions.

[9] We permuted the usual order of the arguments of map so that its type conforms to Definition 17.

We now check subject-reduction. Let $\vdash = \vdash_{\varphi}^{\mathsf{filter}}$. By (var), $\Gamma \vdash x : \mathsf{N}$ and $\Gamma \vdash l : \mathsf{L}_l$. By (var), $\Gamma \vdash f\ x : \mathsf{B}$. 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 (cons), $\Gamma \vdash \mathsf{cond}\ (\mathsf{cons}\ (\mathsf{filter}\ l\ f)\ x)\ (\mathsf{filter}\ l\ f)\ (f\ x) : \mathsf{L}_{\mathtt{s}\, l}$. Therefore, 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 5 (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 (Gödel, 1958), 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 (Howard, 1972) defined by the following rules:

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

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

For the annotated types of function symbols, take $\mathsf{rec}_T^{\mathsf{N}} : \mathsf{N}_{\alpha} \Rightarrow T \Rightarrow (\mathsf{N} \Rightarrow T \Rightarrow T) \Rightarrow T$ and $\mathsf{rec}_T^{\mathsf{O}} : \mathsf{O}_{\alpha} \Rightarrow T \Rightarrow (\mathsf{O} \Rightarrow T \Rightarrow T) \Rightarrow ((\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T) \Rightarrow T$.

We now detail the subject-reduction condition for the last rule of $\mathsf{f} = \mathsf{rec}_T^{\mathsf{O}}$ with $\Gamma = [f : \mathsf{N} \Rightarrow \mathsf{O}_{\beta}, u : T, v : \mathsf{O} \Rightarrow T \Rightarrow T, w : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T]$, $\varphi = \{(\alpha, \mathtt{s}\,\beta)\}$ 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 $\Delta \vdash f : \mathsf{N} \Rightarrow \mathsf{O}_{\beta}$, $\Delta \vdash u : T$, $\Delta \vdash v : \mathsf{O} \Rightarrow T \Rightarrow T$, $\Gamma \vdash w : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T$ and $\Delta \vdash w : (\mathsf{N} \Rightarrow \mathsf{O}) \Rightarrow (\mathsf{N} \Rightarrow T) \Rightarrow T$. By (var), $\Delta \vdash f\ n : \mathsf{O}_{\beta}$. By (app-decr), $\Delta \vdash \mathsf{rec}_T^{\mathsf{O}}\ (f\ n)\ u\ v\ w : T$ since $\beta <_{\mathtt{A}} \mathtt{s}\,\beta$. By (lam), $\Gamma \vdash \lambda n^{\mathsf{N}}.\mathsf{rec}_T^{\mathsf{O}}\ (f\ n)\ u\ v\ w : \mathsf{N} \Rightarrow T$. By (sub), $\Gamma \vdash f : N \Rightarrow \mathsf{O}$ since $\mathsf{N} \Rightarrow \mathsf{O}_{\beta} \leq \mathsf{N} \Rightarrow \mathsf{O}$. Finally, by (var), $\Gamma \vdash w\ f\ (\lambda n^{\mathsf{N}}.\mathsf{rec}_T^{\mathsf{O}}\ (f\ n)\ u\ v\ w) : T$. ■

Example 6 (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:

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

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

For the annotated types of constructors, we take the canonical annotations except for $\mathsf{pair} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha \Rightarrow \mathsf{P}_\alpha$ 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 $\mathsf{fst}, \mathsf{snd} : \mathsf{P}_\alpha \Rightarrow \mathsf{L}_\alpha$, $\mathsf{if} : \mathsf{C}_\alpha \Rightarrow \mathsf{P}_\alpha$, $\mathsf{pivot} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{P}_\alpha$, which expresses the fact that these functions are not size-increasing, and $\mathsf{le} : \mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{B}$, $\mathsf{qs} : \mathsf{L}_\alpha \Rightarrow \mathsf{L} \Rightarrow \mathsf{L}$ and $\mathsf{qsort} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}$.

We now detail the subject-reduction condition for the case of the last rule of qs by taking $\Gamma = [x : \mathsf{N}, l : \mathsf{L}_l, m : \mathsf{L}]$, $\varphi = \{(\alpha, \mathsf{s}\, l)\}$, $\mathsf{fst}, \mathsf{snd}, \mathsf{pivot} <_{\mathbb{F}} \mathsf{qs}$ 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 $l <_{\mathtt{A}} \mathsf{s}\, l$, by (app-decr), $\Gamma \vdash \mathsf{qs}\ p_2\ m : \mathsf{L}$. By (cons), $\Gamma \vdash \mathsf{cons}\ (\mathsf{qs}\ p_2\ m)\ x : \mathsf{L}$. Finally, since $l <_{\mathtt{A}} \mathsf{s}\, l$, by (app-decr) again, $\Gamma \vdash \mathsf{qs}\ p_1\ (\mathsf{cons}\ (\mathsf{qs}\ p_2\ m)\ x) : \mathsf{L}$.

We proved the termination of this system. However, we cannot express that qsort is not size-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:

$$(\forall \alpha)\mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow (\exists \beta)(\exists \gamma)(\alpha = \beta + \gamma)\mathsf{L}_\beta \times \mathsf{L}_\gamma$$ (Blanqui & Riba, 2006). ■

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

Example 7 (Reverse) List reversal can be defined as follows (Huet & Hullot, 1982):

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

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

Since we have a first-order data type, we can assume that $\mathfrak{h} = \omega$. Let $\mathtt{A}$ be the size algebra with the constant $\mathtt{1}$ interpreted by 1 and the binary function symbol $+$ interpreted by the addition. Let $\leq_{\mathtt{A}}$ and $<_{\mathtt{A}}$ be $\leq_{\mathrm{ext}}$ and $<_{\mathrm{ext}}$ respectively (cf. remark after Definition 11).

Consider the 4th rule. Take $\mathsf{cons} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}_{\alpha+1}$, $\mathsf{rev} : \mathsf{L}_\alpha \Rightarrow \mathsf{L}_\alpha$, $\mathsf{revremlast} : \mathsf{L}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{L}_\alpha$, $\Gamma = [x : \mathsf{N}, y : \mathsf{N}_y, l : \mathsf{L}_l]$ and $\varphi = \{(\alpha, l+1)\}$. One can easily check monotony, accessibility and minimality. We now check subject-reduction. Let $\vdash = \vdash^{\mathsf{revremlast}}_{\varphi}$. For comparing termination arguments, take $\mathsf{rev} \simeq_{\mathbb{F}} \mathsf{revremlast}$, $\zeta^{\mathsf{rev}}(a) = 2a$ (formally $a + a$) 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), we get $\Gamma \vdash r : \mathsf{L}_{l+1}$, where $r = \mathsf{rev}\ (\mathsf{cons}\ (\mathsf{rev}\ (\mathsf{revremlast}\ l\ y))\ x)$, since $\zeta^{\mathsf{rev}}(l+1) = 2l + 2 < 2l + 3$. ■

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 (Boyer & Moore, 1979), 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 (Paulson, 1986) 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: $\mathsf{at} : \mathsf{C}_\alpha$, $\mathsf{if} : \mathsf{C}_\alpha \Rightarrow \mathsf{C}_\beta \Rightarrow \mathsf{C}_\gamma \Rightarrow \mathsf{C}_{(\alpha+1)(\beta+\gamma+3)}$ 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 $a = ((u+1)(v+w+3)+1)(y+z+3)) = uvy+uvz+uwy+uwz+3uv+3uw+3uy+3uz+vy+wy+vz+wz+9u+3v+3w+4y+4z+12$, ζ^{nm} be the identity, and $\vdash = \vdash^{\mathsf{nm}}_{\varphi}$. One can easily check monotony, accessibility and minimality. We now check subject-reduction. 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 $b = uvy+uvz+uwy+uwz+2uy+2uz+vy+vz+wy+wz+9u+2y+2z+9 <_{\mathtt{A}} a$. So, by (sub), $\Gamma \vdash \mathsf{nm}\,(\mathsf{if}\, u\,(\mathsf{nm}\,(\mathsf{if}\, v\, y\, z))\,(\mathsf{nm}\,(\mathsf{if}\, w\, y\,))) : \mathsf{C}_a$. ■

7 Decidability of $\vdash^{\mathsf{f}}_{\varphi}$

In this section, we provide an algorithm for deciding the relation $\vdash^{\mathsf{f}}_{\varphi}$ used in Theorem 1 and defined in Figures 3 and 4, under general conditions on the size algebra $\mathtt{A}$. We will prove in Section 9 that these conditions are satisfied by the successor algebra.

The differences between $\vdash^{\mathsf{f}}_{\varphi}$ and the usual typing relation for simply-typed λ-calculus are the following. First, the set of typable symbols is restricted to those smaller than or equivalent to 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:

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

that is (app-decr) without the decreasingness condition $(h, \psi) <_{\mathtt{A}} (\mathsf{f}, \varphi)$. 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 (app) node, the decreasingness condition is satisfied.

In Section 7.1, we provide an algorithm for deciding $\vdash^{\mathsf{f}}$. Then, in Section 7.2, we show how to use this algorithm to decide $\vdash^{\mathsf{f}}_{\varphi}$.

7.1 Decidability of $\vdash^{\mathsf{f}}$

First note that, in a given typing environment Γ, a typable term t may have several and even infinitely many types for two reasons. First, in (app), the size variables of function types can be instantiated arbitrarily. Second, subtyping is generally not bounded. For instance, in the successor algebra, $\mathsf{N}_\alpha \leq \mathsf{N}_{\mathtt{s}\alpha} \leq \ldots$

The relation $\vdash^{\mathsf{f}}$ differs from Curry and Feys' typing relation with functional characters or type-schemes (a type with type variables) (Curry & Feys, 1958) in two points. First, our type-schemes are not built from type variables but from size variables. Second, we have a subtyping relation. We will however see that some techniques developed for Curry and Feys' type system or, more generally, Milner's type system (Milner, 1978) and its extensions, can be adapted to our framework.

The decidability of type-checking in Curry and Feys' system has been proved by Hindley in (Hindley, 1969). Hindley's algorithm is based on the fact the set of types of a typable term has a smallest element wrt some ordering $\sqsubseteq$. Hence, to decide whether $\Gamma \vdash t : T$, the algorithm proceeds in two steps. First, it computes the smallest type of t, say U, and then checks whether $U \sqsubseteq T$.

In Curry and Feys' system, $\sqsubseteq$ is the instantiation ordering: a type-scheme U is an instance of a type-scheme T, or T is more general than U, written $T \sqsubseteq U$, if $T\theta = U$ for some substitution θ. In (Huet, 1976), Huet proved that every non-empty set of terms has a greatest lower bound wrt. $\sqsubseteq$. So, in particular, $\{T \mid \Gamma \vdash t : T\}$ has a greatest lower bound if t is typable in Γ.

For computing the most general type, Hindley uses an algorithm based on unification (Herbrand, 1930; Robinson, 1965). Unifying two terms T and U consists in solving the equation $T = U$, that is, in finding a substitution θ such that $T\theta = U\theta$. In (Huet, 1976), 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 terms has a least upper bound wrt. $\sqsubseteq$. Hence, every solvable unification problem has a most general solution.

Hindley's 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 generalizations seems to be Sulzmann's HM(X) system (Sulzmann, 2001), 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.

But, since we also have subtyping, we define $\sqsubseteq$ as follows:

Definition 20 (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 21 (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 $T \leq^? U$.

Fig. 5. Type inference algorithm

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

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

It has a solution $\varphi : \mathtt{V} \to \overline{\mathtt{A}}$ 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 $\mathrm{Sol}_{\overline{\mathtt{A}}}(P)$ 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 $\varphi\theta \leq^{\infty}_{\mathtt{A}} \psi$, *i.e.* there is θ such that, for all α, $\alpha\varphi\theta \leq^{\infty}_{\mathtt{A}} \alpha\psi$. Finally, let $\equiv$ be the equivalence relation $\sqsubseteq \cap \sqsupseteq$.

Again, one can easily check that the ordering $\sqsubseteq$ on substitutions is a quasi-ordering.
In order to compute the most general type of a term, we make the following assumptions:

- every solvable subtyping problem P has a most general solution $\mathrm{mgs}(P)$;
- 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 9 that these assumptions are satisfied when types are annotated in the successor algebra. On the other hand, they are not generally satisfied in an algebra with addition.

Now, following (Hindley, 1969), the computation of the most general type is defined by the rules of Figure 5 where $\Gamma \vdash^{\mathsf{f}} t \uparrow U$ means that, in the typing environment Γ, the most general type of t is U. In the case of an application $h\vec{w}$, the algorithm proceeds as follows:

1. Check that h is declared. Let T be its declared type.
2. Check that h can take $n = |\vec{w}|$ arguments, *i.e.* T is of the form $\vec{V} \Rightarrow V$ with $|\vec{V}| = n$.
3. If h is a function symbol equivalent to f, check that $|\vec{V}| \geq \mathrm{q}^h$.
4. Try to infer the types of every w_i.
5. If this succeeds with U_i for the type of w_i, then rename the variables of every U_i using a permutation ρ_i, so that, for all i, $U_i\rho_i$ has no variable in common with T and, for all $i \neq j$, $U_i\rho_i$ and $U_j\rho_j$ have no variable in common.

6. Finally, try 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 return $V\eta$ in case of success.

Example 9 To carry on with Example 3, let $r = \mathsf{s}\ (\mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y))$ be the right hand-side of the last rule of Figure 1. We would like to infer the type of r in $\Gamma = [x : \mathsf{N}_x, y : \mathsf{N}]$ when $\mathsf{s} : \mathsf{N}_\alpha \Rightarrow \mathsf{N}_{\mathsf{s}\alpha}$, $\mathsf{sub} : \mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}_\alpha$ and $\mathsf{div} : \mathsf{N}_\alpha \Rightarrow \mathsf{N} \Rightarrow \mathsf{N}_\alpha$. Let $\vdash = \vdash^{\mathsf{div}}$ and assume wlog that x is a constant of the successor algebra.

By (inf-app), we get (1) $\Gamma \vdash x \uparrow \mathsf{N}_x$ and (2) $\Gamma \vdash y \uparrow \mathsf{N}$. From (1) and (2), by (inf-app), we get (3) $\Gamma \vdash \mathsf{sub}\ x\ y \uparrow \mathsf{N}_x$ since, as we shall see in Section 9.2, $\mathrm{mgs}\{\mathsf{N}_x \leq^? \mathsf{N}_\alpha, \mathsf{N} \leq^? \mathsf{N}\} = \{(\alpha, x)\}$. From (2), by (inf-app), we get (4) $\Gamma \vdash \mathsf{s}\ y \uparrow \mathsf{N}$ since $\mathrm{mgs}\{\mathsf{N} \leq^? \mathsf{N}_\alpha\} = \{(\alpha, \infty)\}$. From (3) and (4), we get (5) $\Gamma \vdash \mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y) \uparrow \mathsf{N}_x$ since $\mathrm{mgs}\{\mathsf{N}_x \leq^? \mathsf{N}_\alpha, \mathsf{N} \leq^? \mathsf{N}\} = \{(\alpha, x)\}$. From (5), by (inf-app), we get $\Gamma \vdash r \uparrow \mathsf{N}_{\mathsf{s}x}$ since $\mathrm{mgs}\{\mathsf{N}_x \leq^? \mathsf{N}_\alpha\} = \{(\alpha, x)\}$. ■

We now prove that this algorithm is correct and complete wrt $\vdash^{\mathsf{f}}$, when the size algebra is monotone and the algorithm is applied to an environment Γ having no size variables. To extend in the next section this result to $\vdash^{\mathsf{f}}_\varphi$, we need to make derivations explicit:

Definition 22 (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 $\Gamma \vdash^{\mathsf{f}} w_i : V_i\psi$, written $\pi_i \rhd \Gamma \vdash^{\mathsf{f}} w_i : V_i\psi$, then $\mathrm{a}(\Gamma, h\vec{w}, \psi, \vec{\pi})$ is the derivation of $\Gamma \vdash^{\mathsf{f}} h\vec{w} : V\psi$ 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).

Simarly, 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, $\pi_i \rhd \Gamma \vdash^{\mathsf{f}} w_i \uparrow U_i$, then $\mathrm{i}(\Gamma, h\vec{w}, \vec{\rho}, \vec{\pi})$ is the derivation of $\Gamma \vdash^{\mathsf{f}} h\vec{w} \uparrow V\eta$ 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 π for $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$, we write $\pi \rhd \Gamma \vdash^{\mathsf{f}}_\varphi t : T$ if, at every (app) node in π, the decreasingness condition of (app-decr) is satisfied, that is, if $\Gamma \vdash^{\mathsf{f}}_\varphi t : T$.

Note that $\Gamma \vdash^{\mathsf{f}} t \uparrow T$ has at most one derivation.

Lemma 18 If $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$ then, for every size substitution φ, $\pi\varphi \rhd \Gamma\varphi \vdash^{\mathsf{f}} t : T\varphi$.

Proof. Straightforward induction using the fact that $\leq_{\mathtt{A}}$ and thus $\leq^\infty_{\mathtt{A}}$ and $\leq$ are stable by substitution. In the case of (app), by induction hypothesis, we have $\Gamma\varphi \vdash^{\mathsf{f}} w_i : V_i\psi'$ with $\psi' = \psi\varphi$. Therefore, by (app), $\Gamma\varphi \vdash^{\mathsf{f}} h\vec{w} : V\psi' = (V\psi)\varphi$. ■

Lemma 19 (Correctness wrt. $\vdash^{\mathsf{f}}$) If $\pi \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow T$ and $\mathrm{Var}(\Gamma) = \emptyset$, then there is $|\pi|$ such that $|\pi| \rhd \Gamma \vdash^{\mathsf{f}} t : T$. In particular, for (inf-app), $|\mathrm{i}(\Gamma, h\vec{w}, \vec{\rho}, \vec{\pi})| = \mathrm{a}(\Gamma, h\vec{w}, \eta, \vec{\upsilon})$ where $\upsilon_i = \mathrm{s}(|\pi_i|\rho_i\eta, V_i\eta)$.

Proof. By induction on $\Gamma \vdash^{\mathsf{f}} t \uparrow T$. We only detail the case (inf-app). By induction hypothesis, $\Gamma \vdash^{\mathsf{f}} w_i : U_i$. By Lemma 18, $\Gamma \vdash^{\mathsf{f}} w_i : U_i\rho_i\eta$ since $\mathrm{Var}(\Gamma) = \emptyset$. Since $U_i\rho_i\eta \leq V_i\eta$, by (sub), $\Gamma \vdash^{\mathsf{f}} w_i : V_i\eta$. Therefore, by (app), $\Gamma \vdash^{\mathsf{f}} h\vec{w} : V\eta$. ■

Lemma 20 If $\varphi \leq^{\infty}_{\mathtt{A}} \psi$ and, for all α, $\mathrm{Pos}(\alpha, T) \subseteq \mathrm{Pos}^{+}(T)$, then $T\varphi \leq T\psi$.

Proof. We say that $T \in \mathbb{T}_{\mathtt{A}} \cup \mathtt{A}$ is $\delta \in \{+,-\}$ if, for all α, $\mathrm{Pos}(\alpha, T) \subseteq \mathrm{Pos}^{\delta}(T)$. We first prove that (*) if a is δ then $a\varphi(\leq^{\infty}_{\mathtt{A}})^{\delta}a\psi$ where, for any relation R, $R^{+} = R$ and $R^{-} = R^{-1}$. We proceed by induction on a:

- a is a variable α. Then, $\delta = +$ and $\alpha\varphi \leq^{\infty}_{\mathtt{A}} \alpha\psi$ since $\varphi \leq^{\infty}_{\mathtt{A}} \psi$.
- $a = \mathtt{f}a_1 \ldots a_n$. Let $i \in \{1,\ldots,n\}$. If $i \in \mathrm{Mon}^{\varepsilon}(\mathtt{f})$ (cf. Definition 13), then a_i is $\delta\varepsilon$ and, by induction hypothesis, $a_i\varphi(\leq^{\infty}_{\mathtt{A}})^{\delta\varepsilon}a_i\psi$. If $i \notin \mathrm{Mon}^{+}(\mathtt{f}) \cup \mathrm{Mon}^{-}(\mathtt{f})$ then a_i contains no variable and $a_i\varphi = a_i\psi$. Therefore, by monotony of $\mathtt{f}$ in every $i \in \mathrm{Mon}^{+}(\mathtt{f})$, anti-monotony of $\mathtt{f}$ in every $i \in \mathrm{Mon}^{-}(\mathtt{f})$, and transitivity, we get $a\varphi(\leq^{\infty}_{\mathtt{A}})^{\delta}a\psi$.

We now prove that, if T is δ, then $T\varphi \leq^{\delta} T\psi$, by induction on T.

- $T = U \Rightarrow V$. Then, U is $-\delta$ and V is δ. So, by induction hypothesis, $U\varphi \leq^{-\delta} U\psi$ and $V\varphi \leq^{\delta} V\psi$. Therefore, by (prod), $(U \to V)\varphi \leq^{\delta} (U \Rightarrow V)\psi$.
- $T = \mathsf{B}_a$. Then, a is δ and, by (*), $a\varphi \leq^{\infty}_{\mathtt{A}} a\psi$. Therefore, by (size), $T\varphi \leq T\psi$. ■

Lemma 21 (Completeness wrt. $\vdash^{\mathsf{f}}$) In monotone algebras, if $\pi \rhd \Gamma \vdash^{\mathsf{f}} t : T$ and $\mathrm{Var}(\Gamma) = \emptyset$, then there are U and $\pi\uparrow$ such that $\pi\uparrow \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow U$ and $U \sqsubseteq T$. In particular, $\mathrm{a}(\Gamma, h\vec{w}, \psi, \vec{\pi})\uparrow = \mathrm{i}(\Gamma, h\vec{w}, \vec{\rho}, \vec{\pi}\uparrow)$ where $\vec{\rho}$ are permutations satisfying the conditions of rule (inf-app).

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, $\Gamma \vdash^{\mathsf{f}} w_i \uparrow U_i$ and there is χ_i such that $U_i\chi_i \leq V_i\psi$. Wlog. we can assume that $\mathrm{dom}(\chi_i) \subseteq \mathrm{Var}(U_i)$. Let now $\rho_1, \ldots, \rho_n$ be permutations satisfying the conditions of (inf-app), and $\xi = \{(\alpha, \alpha\psi) \mid \alpha \in \mathrm{Var}(\vec{V} \Rightarrow V)\} \cup \{(\alpha, \alpha\rho_i^{-1}\chi_i) \mid \alpha \in \mathrm{Var}(U_i\rho_i), 1 \leq i \leq n\}$. Then, for all i, $U_i\rho_i\xi = U_i\chi_i \leq V_i\psi = V_i\xi$. 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 χ such that $\eta\chi \leq^{\infty}_{\mathtt{A}} \xi$. Hence, by (inf-app), $\Gamma \vdash^{\mathsf{f}} h\vec{w} \uparrow V\eta$. By the monotony condition, variables occur only positively in V. Therefore, by Lemma 20, $V\eta\chi \leq V\xi = V\psi$. Hence, $V\eta \sqsubseteq V\psi$. ■

7.2 Decidability of $\vdash^{\mathsf{f}}_{\varphi}$

We now prove that, when the size algebra is monotone, for checking $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$, it is sufficient to check whether there are U and χ such that $\Gamma \vdash^{\mathsf{f}} t \uparrow U$, $U\chi \leq T$ and also that, if one denotes by υ the (unique) derivation of $\Gamma \vdash^{\mathsf{f}} t \uparrow U$, then $|\upsilon|\chi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : U\chi$, that is, at every (app) node in $|\upsilon|\chi$, the decreasingness condition is satisfied.

Lemma 22 In monotone algebras, 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 $(h, \vec{V} \Rightarrow V) \in \overline{\Theta}$, $\pi_i \rhd \Gamma \vdash^{\mathsf{f}} w_i : V_i\psi$, $\pi_i\xi' \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : V_i\psi\xi'$ and $\vec{\alpha}^{h}\psi\xi' <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$. By induction hypothesis, $\pi_i\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : V_i\psi\xi$. Since $\xi \leq^{\infty}_{\mathtt{A}} \xi'$ and the size algebra is

Fig. 6. Algorithm for deciding whether $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$.

1. Check whether there is U such that $\Gamma\gamma \vdash^{\mathsf{f}} t \uparrow U$, where γ is an injection from the set of size variables of Γ, T and φ to the set of constants $\mathtt{C}$ not occurring in Γ, T and φ. If it fails, then t is not typable in Γ.
2. If it succeeds, try to compute $\chi = \mathrm{mgs}\{U \leq^{?} T\gamma\}$. If it fails, $\Gamma \vdash^{\mathsf{f}} t : T$ does not hold.
3. If it succeeds, then check whether $|\upsilon|\chi \rhd \Gamma\gamma \vdash^{\mathsf{f}}_{\varphi\gamma} t : U\chi$ where υ is the unique derivation of $\Gamma\gamma \vdash^{\mathsf{f}} t \uparrow U$.
   If it succeeds, then $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ holds. Otherwise, $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ does not hold.

monotone, we have $\psi\xi \leq^{\infty}_{\mathtt{A}} \psi\xi'$. Since $(\leq^{\infty}_{\mathtt{A}})_{\mathrm{prod}} \circ <^{h,\mathsf{f}}_{\mathtt{A}} \subseteq <^{h,\mathsf{f}}_{\mathtt{A}}$ (cf. Definition 17), we have $\vec{\alpha}^h\psi\xi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$. Therefore, $\pi\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} h\vec{w} : V\psi\xi$. ■

Lemma 23 (Completeness wrt. $\vdash^{\mathsf{f}}_{\varphi}$) Let $\mathtt{A}$ be a monotone algebra. Assume that $\pi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ and $\mathrm{Var}(\Gamma) = \emptyset$. By lemma 21, there are U and χ such that $\pi\uparrow\Gamma \vdash^{\mathsf{f}} t \uparrow U$ and $U\chi \leq T$. Then, $|\pi\uparrow|\chi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : U\chi$.

Proof. We prove that if $\pi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$, $\mathrm{Var}(\Gamma) = \emptyset$, $\pi\uparrow \rhd \Gamma \vdash^{\mathsf{f}} t \uparrow U$ and $U\chi \leq T$, then $\mathrm{s}(|\pi\uparrow|\chi, T) \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$, by induction on $\pi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$. We only detail the case (app-decr) when $t = h\vec{w}$, $(h, \vec{V} \Rightarrow V) \in \overline{\Theta}$, $T = V\psi$, $\vec{\alpha}^h\psi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$ and $U = V\eta$ where η is given by the rule (inf-app). We have $\pi = \mathrm{a}(\Gamma, h\vec{w}, \psi, \vec{\pi})$, $\pi\uparrow = \mathrm{i}(\Gamma, h\vec{w}, \vec{\rho}, \vec{\pi}\uparrow)$, $|\pi\uparrow| = \mathrm{a}(\Gamma, h\vec{w}, \eta, \vec{\upsilon})$ where $\upsilon_i = \mathrm{s}(|\pi_i\uparrow|\rho_i\eta, V_i\eta)$, $|\pi\uparrow|\chi = \mathrm{a}(\Gamma, h\vec{w}, \eta\chi, \vec{\upsilon}\chi)$ and, for all i, $\pi_i \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : V_i\psi$, $\pi_i\uparrow \rhd \Gamma \vdash^{\mathsf{f}} w_i \uparrow U_i$ and $U_i\chi_i \leq V_i\psi$ for some χ_i. By induction hypothesis, $\mathrm{s}(|\pi_i\uparrow|\chi_i, V_i\psi) \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : V_i\psi$. In particular, $|\pi_i\uparrow|\chi_i \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : U_i\chi_i$, that is, $|\pi_i\uparrow|\rho_i\xi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : U_i\rho_i\xi$, where ξ is defined in the proof of Lemma 21. Since $\eta\chi \leq^{\infty}_{\mathtt{A}} \xi$, by Lemma 22, we get $|\pi_i\uparrow|\rho_i\eta\chi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : U_i\rho_i\eta\chi$. Hence, $\upsilon_i\chi \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} w_i : V_i\eta\chi$. Moreover, since $\vec{\alpha}^h\eta\chi \leq^{\infty}_{\mathtt{A}} \vec{\alpha}^h\xi = \vec{\alpha}^h\psi$ and $\vec{\alpha}^h\psi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$, we get $\vec{\alpha}^h\eta\chi <^{h,\mathsf{f}}_{\mathtt{A}} \vec{\alpha}^{\mathsf{f}}\varphi$ by assumption on $<^{h,\mathsf{f}}_{\mathtt{A}}$. Therefore, $\mathrm{s}(|\pi\uparrow|\chi, T) \rhd \Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$. ■

The previous lemmas assume that there are no size variables in Γ. So, to use these lemmas, we need to be able to replace size variables by constants (*aka* eigenvariables). Under this assumption, we can conclude:

Theorem 2 (Decidability of $\vdash^{\mathsf{f}}_{\varphi}$) Assume that $\mathtt{A}$ is an algebra such that:

- $\mathtt{A}$ is monotone;
- $\mathtt{A}$ contains an infinite set of constants $\mathtt{C}$ such that, if $a \leq_{\mathtt{A}} b$ ($\vec{a} <^{\mathsf{g},\mathsf{f}}_{\mathtt{A}} \vec{b}$ resp.) then, for all $\mathtt{c} \in \mathtt{C}$ and $e \in \mathtt{A}$, $a\delta \leq_{\mathtt{A}} b\delta$ ($\vec{a}\delta <^{\mathsf{g},\mathsf{f}}_{\mathtt{A}} \vec{b}\delta$ resp.), where δ replaces every $\mathtt{c}$ by e;
- $<_{\mathbb{F}}$ is decidable and, for all $\mathsf{g} \simeq_{\mathbb{F}} \mathsf{f}$, $<^{\mathsf{g},\mathsf{f}}_{\mathtt{A}}$ is decidable;
- the satisfiability of a subtyping problem is decidable;
- every satisfiable problem P has a most general solution $\mathrm{mgs}(P)$ that is computable.

Given Γ, t and T, one can decide whether $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ by using the algorithm of Figure 6.

Proof.

- Correctness. Assume that the algorithm succeeds. Then, $\Gamma\gamma \vdash^{\mathsf{f}}_{\varphi\gamma} t : U\chi$ and $U\chi \leq T\gamma$. By (sub), $\Gamma\gamma \vdash^{\mathsf{f}}_{\varphi\gamma} t : T\gamma$. Then, by applying $\delta = \gamma^{-1}$, we get $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$.
- Completeness. Assume that the algorithm fails in step 1 or 2 then, by Lemma 21, t is not typable in $\Gamma\gamma$. Therefore, it is not typable in Γ either. Finally, if the algorithm fails in step 3 then, by Lemma 23, there is no derivation of $\Gamma\gamma \vdash^{\mathsf{f}}_{\varphi\gamma} t : T\gamma$. Therefore, there is no derivation of $\Gamma \vdash^{\mathsf{f}}_{\varphi} t : T$ either. ■

That the successor algebra satisfies the first two conditions follows from Lemma 25.

Example 10 To carry on with Example 9, we now would like to check whether $\Gamma \vdash r : \mathsf{N}_\alpha\varphi$ where $\vdash = \vdash^{\mathsf{div}}_{\varphi}$ and $\varphi = \{(\alpha, \mathsf{s}\,x)\}$. We have seen that $\Gamma \vdash^{\mathsf{div}} r \uparrow \mathsf{N}_{\mathsf{s}\,x}$. Hence, χ is the identity and we are left to check that, in every (app) node with $h \simeq_{\mathbb{F}} \mathsf{div}$, the decreasingness condition is satisfied. Here, it amounts to check that, in the (app) node for $(\mathsf{div}\ (\mathsf{sub}\ x\ y)\ (\mathsf{s}\ y))$, the size annotation for the type of $\mathsf{sub}\ x\ y$, that is x, is smaller than $\alpha\varphi = \mathsf{s}\,x$, which is indeed the case. ■

8 Reducing subtyping problems to size problems

For the type inference algorithm we just saw, we assumed the existence of an algorithm to compute the most general solution of a subtyping problem. In this section, we show how a subtyping problem can be reduced to solving constraints in $\overline{\mathtt{A}}$. 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 rules (refl) and (trans) are redundant, that is, they can be eliminated, following a proof technique used by Curien and Ghelli in (Curien & Ghelli, 1992):

Theorem 3 $T \leq U$ iff $T \leq_{\mathrm{a}} U$, where $\leq_{\mathrm{a}}$ is defined by the rules (size) and (prod) only.

Proof. We first prove that (refl) can be eliminated, hence that $\leq = \leq'$ where $\leq'$ is the relation defined by (size), (prod) and (trans) only. Indeed, using the reflexivity of $\leq^{\infty}_{\mathtt{A}}$, one can easily prove that $T \leq_{\mathrm{a}} T$, by induction on T.

We now prove that, in turn, (trans) can be eliminated, hence that $\leq = \leq_{\mathrm{a}}$. More precisely, we prove that, if π is a derivation of $A \leq' B$ of height n, then $A \leq_{\mathrm{a}} B$, by induction on n. We proceed by case on the last rule:

(size) Immediate.

(prod) Assume that $U \Rightarrow V \leq' U' \Rightarrow V'$ ends with (prod). By induction hypothesis, $U' \leq_{\mathrm{a}} U$ and $V \leq_{\mathrm{a}} V'$. Hence, by (prod), $U \Rightarrow V \leq_{\mathrm{a}} U' \Rightarrow V'$.

(trans) Assume that $T \leq' U$ and $U \leq' V$. By induction hypothesis, $T \leq_{\mathrm{a}} U$ and $U \leq_{\mathrm{a}} V$. If $T \leq_{\mathrm{a}} U$ ends with (size), then $T = \mathsf{B}_a$, $U = \mathsf{B}_b$ and $a \leq^{\infty}_{\mathtt{A}} b$. Therefore, $U \leq_{\mathrm{a}} V$ ends with (size) too, $V = \mathsf{B}_c$ and $b \leq^{\infty}_{\mathtt{A}} c$. Hence, by transitivity of $\leq^{\infty}_{\mathtt{A}}$, $T \leq_{\mathrm{a}} V$. Similarly, if $U \leq_{\mathrm{a}} V$ ends with (size), then $T \leq_{\mathrm{a}} U$ ends with (size) and $T \leq_{\mathrm{a}} V$. So, we are left with the case where both $T \leq_{\mathrm{a}} U$ and $U \leq_{\mathrm{a}} V$ ends with (prod):

$$\dfrac{\dfrac{\dfrac{\pi_{11}}{A' \leq_{\mathrm{a}} A} \quad \dfrac{\pi_{12}}{B \leq_{\mathrm{a}} B'}}{A \Rightarrow B \leq_{\mathrm{a}} A' \Rightarrow B'}\,(\mathrm{prod}) \quad \dfrac{\dfrac{\pi_{21}}{A'' \leq_{\mathrm{a}} A'} \quad \dfrac{\pi_{22}}{B' \leq_{\mathrm{a}} B''}}{A' \Rightarrow B' \leq_{\mathrm{a}} A'' \Rightarrow B''}\,(\mathrm{prod})}{A \Rightarrow B \leq' A'' \Rightarrow B''}\,(\mathrm{trans})$$

But $A \Rightarrow B \leq' A'' \Rightarrow B''$ can also be proved as follows:

$$\dfrac{\dfrac{\dfrac{\pi_{21}}{A'' \leq_{\mathrm{a}} A'} \quad \dfrac{\pi_{11}}{A' \leq_{\mathrm{a}} A}}{A'' \leq' A}\,(\text{trans}) \qquad \dfrac{\dfrac{\pi_{12}}{B \leq_{\mathrm{a}} B'} \quad \dfrac{\pi_{22}}{B' \leq_{\mathrm{a}} B''}}{B \leq' B''}\,(\text{trans})}{A \Rightarrow B \leq' A'' \Rightarrow B''}\,(\text{prod})$$

The derivation heights of $A'' \leq' A$ and $B \leq' B''$ are strictly smaller than the derivation height of $A \Rightarrow B \leq' A'' \Rightarrow B''$. Therefore, by induction hypothesis, $A'' \leq_{\mathrm{a}} A$ and $B \leq_{\mathrm{a}} B''$. Hence, by (prod), $A \Rightarrow B \leq_{\mathrm{a}} A'' \Rightarrow B''$. ■

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

Definition 23 (Size problem) A size constraint is a pair of size expressions (a,b), written $a \leq^? b$. A *size problem* P is either $\bot$ or a finite set of size constraints. It has a solution $\varphi : \mathtt{V} \to \overline{\mathtt{A}}$ if $P \neq \bot$, $\mathrm{dom}(\varphi) \subseteq \mathrm{Var}(P)$ and, for all $a \leq^? b \in P$, $a\varphi \leq^{\infty}_{\mathtt{A}} b\varphi$. A solution φ is *finite* if $\varphi : \mathtt{V} \to \mathtt{A}$. Let $\mathrm{Sol}_{\overline{\mathtt{A}}}(P)$ ($\mathrm{Sol}_{\mathtt{A}}(P)$ resp.) be the set of the (*finite* resp.) solutions 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 24 $\mathrm{Sol}(P) = \mathrm{Sol}_{\overline{\mathtt{A}}}(|P|)$.

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

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

9 Solving size problems in the successor algebra

We have seen in the previous section that solving a subtyping problem can be reduced to solving inequalities in $\overline{\mathtt{A}}$. In this section, we consider a specific size algebra $\mathtt{A}$, the successor algebra, and prove that, in this algebra, the solvability of a size problem is decidable in polynomial time, and that solvable size problems have a most general solution that can be computed in polynomial time too.

The relations $\leq_{\mathtt{A}}$ and $<_{\mathtt{A}}$ of the successor algebra (Definition 12) are equivalently defined by the rules of Figure 7. We start by proving basic properties of $\leq_{\mathtt{A}}$, the quasi-ordering $\sqsubseteq$ and its associated equivalence relation $\equiv$ on size substitutions introduced in Definition 21.

Fig. 7. Ordering in the successor algebra

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

Lemma 25

- $a \leq_{\mathtt{A}} b$ ($a <_{\mathtt{A}} b$ resp.) iff there is $k \geq 0$ ($k > 0$ resp.) such that $b = \mathtt{s}^k a$.
- $\mathtt{s}a <_{\mathtt{A}} \mathtt{s}b$ iff $a <_{\mathtt{A}} b$.

Proof.

- One can easily check $a \leq_{\mathtt{A}} \mathtt{s}^k a$ by induction on $k \geq 0$. We have $a \leq_{\mathtt{A}} a$ by definition. Assume now that $a \leq_{\mathtt{A}} \mathtt{s}^k a$. Since $\mathtt{s}^k a <_{\mathtt{A}} \mathtt{s}^{k+1} a$ holds by definition, we get $a \leq_{\mathtt{A}} \mathtt{s}^{k+1} a$ by transitivity.
  Similarly, once can easily check $a <_{\mathtt{A}} \mathtt{s}^k a$ by induction on $k \geq 1$. We have $a <_{\mathtt{A}} \mathtt{s}a$ by definition. Assume now that $a <_{\mathtt{A}} \mathtt{s}^k a$. Since $\mathtt{s}^k a <_{\mathtt{A}} \mathtt{s}^{k+1} a$ by definition, we get $a <_{\mathtt{A}} \mathtt{s}^{k+1} a$ by transitivity.
  We now prove that, if $a <_{\mathtt{A}} b$, then there is b' such that $b = \mathtt{s}b'$ and $a \leq_{\mathtt{A}} b'$, 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, $a \leq_{\mathtt{A}} b'$ since $\leq_{\mathtt{A}}$ is the reflexive closure of $<_{\mathtt{A}}$ and $<_{\mathtt{A}}$ is transitive.
  We finally prove that there is $k \geq 0$ whenever $a \leq_{\mathtt{A}} b$, by induction on b. If $a = b$, this is immediate. If $a <_{\mathtt{A}} b$, then there is b' such that $b = \mathtt{s}b'$ and $a \leq_{\mathtt{A}} b'$. By induction hypothesis, $b' = \mathtt{s}^k a$ for some $k \geq 0$. Therefore, $b = \mathtt{s}^{k+1} a$.
- If $\mathtt{s}a <_{\mathtt{A}} \mathtt{s}b$, then $\mathtt{s}b = \mathtt{s}^{k+1}\mathtt{s}a$ for some k. Therefore, $b = \mathtt{s}^{k+1} a$. Conversely, if $a <_{\mathtt{A}} b$, then $b = \mathtt{s}^{k+1} a$ for some k. Therefore, $\mathtt{s}b = \mathtt{s}^{k+1}\mathtt{s}a$. ■

It follows that the successor algebra is monotone and also that $\leq_{\mathtt{A}}$ and $\leq_{\mathtt{A}}^{\infty}$ are orderings, as well as their pointwise extensions to substitutions.

Definition 24 (Successor and head parts of a substitution) To a substitution $\varphi : \mathtt{V} \to \overline{\mathtt{A}}$, we associate two unique maps $\varphi_s : \mathtt{V} \to \mathbb{N}$ and $\varphi_h : \mathtt{V} \to \mathtt{V} \cup \mathtt{C} \cup \{\infty\}$ such that, for all α, $\alpha\varphi = \mathtt{s}^{\alpha\varphi_s}\alpha\varphi_h$ with $\alpha\varphi_s = 0$ if $\alpha\varphi_h = \infty$.

Lemma 26 $\varphi \sqsubseteq \psi$ iff there is $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{C} \cup \{\infty\}$ such that $\varphi\rho \leq_{\mathtt{A}}^{\infty} \psi$.

Proof. The "if" part is immediate. We now prove the "only if" part. Assume that there is θ such that $\varphi\theta \leq_{\mathtt{A}}^{\infty} \psi$. Let $\rho = \theta_h|_{\mathrm{Var}(\varphi_h)}$, where $\mathrm{Var}(\varphi_h) = \bigcup\{\mathrm{Var}(\alpha\varphi_h) \mid \alpha \in \mathrm{dom}(\varphi_h)\}$. We now check that $\varphi\rho \leq_{\mathtt{A}}^{\infty} \psi$. If $\alpha\varphi_h \notin \mathtt{V}$, then $\alpha\varphi\rho = \alpha\varphi\theta \leq_{\mathtt{A}}^{\infty} \alpha\psi$. Otherwise, $\alpha\varphi\rho = \mathtt{s}^{\alpha\varphi_s}\alpha\varphi_h\theta_h \leq_{\mathtt{A}}^{\infty} \mathtt{s}^{\alpha\varphi_s+\alpha\varphi_h\theta_s}\alpha\varphi_h\theta_h = \alpha\varphi\theta \leq_{\mathtt{A}}^{\infty} \alpha\psi$. ■

Lemma 27 Let V be a set, and V_1 and V_2 be subsets of V. If $\rho_1 : V_1 \to V_2$ and $\rho_2 : V_2 \to V_1$ are injections, then there is a permutation $\xi : V \to V$ such that $\xi|_{V_1} = \rho_1$.

Proof. By Cantor-Bernstein theorem, V_1 and V_2 are equipotent. Hence, $V_1 - V_2$ and $V_2 - V_1$ are equipotent as well. Let ν be any bijection from $V_2 - V_1$ to $V_1 - V_2$, and $\xi = \{(\alpha, \alpha\rho_1) \mid \alpha \in V_1\} \cup \{(\alpha, \alpha\nu) \mid \alpha \in V_2 - V_1\}$. The function ξ is a bijection on $V_1 \cup V_2$ and $\xi|_{V_1} = \rho_1$. ■

Lemma 28 $\varphi_2 \equiv \varphi_1$ iff $\varphi_2 = \varphi_1\xi$ for some permutation $\xi : \mathtt{V} \to \mathtt{V}$.

Proof. If "if" part is immediate. We now prove the "only if" part. In (Huet, 1976), Huet proved this result when $\leq^\infty_{\mathtt{A}}$ is the equality. His proof can be adapted to our more general situation since $\alpha \leq^\infty_{\mathtt{A}} \beta$ iff $\alpha = \beta$. By assumption and Lemma 26, there are $\rho_1, \rho_2 : \mathtt{V} \to \mathtt{V} \cup \mathtt{C} \cup \{\infty\}$ such that $\varphi_1\rho_1 \leq^\infty_{\mathtt{A}} \varphi_2$ and $\varphi_2\rho_2 \leq^\infty_{\mathtt{A}} \varphi_1$. By stability, we have $\varphi_1\rho_1\rho_2 \leq^\infty_{\mathtt{A}} \varphi_2\rho_2$. Hence, by transitivity, $\varphi_1\rho_1\rho_2 \leq^\infty_{\mathtt{A}} \varphi_1$. Similarly, $\varphi_2\rho_2\rho_1 \leq^\infty_{\mathtt{A}} \varphi_2$.

We now prove that ρ_1 is an injection from V_1 to V_2, where $V_i = \bigcup\{\mathrm{Var}(\beta\varphi_i) \mid \beta \in V\}$ and $V = \mathrm{dom}(\varphi_1) \cup \mathrm{dom}(\varphi_2)$. Let $\alpha \in V_1$. Then, there is $\beta \in V$ such that $\alpha \in \mathrm{Var}(\beta\varphi_1)$. Hence, $\beta\varphi_1 = \mathtt{s}^k\alpha$ for some $k \in \mathbb{N}$. Since $\varphi_1\rho_1\rho_2 \leq^\infty_{\mathtt{A}} \varphi_1$, we have $\beta\varphi_1\rho_1\rho_2 = \mathtt{s}^k\alpha\rho_1\rho_2 \leq^\infty_{\mathtt{A}} \beta\varphi_1 = \mathtt{s}^k\alpha$. Therefore, $\alpha\rho_1\rho_2 = \alpha$ and ρ_1 is an injection on V_1. We now prove that $\gamma = \alpha\rho_1 \in V_2$. Since $\varphi_1\rho_1 \leq^\infty_{\mathtt{A}} \varphi_2$, we have $\beta\varphi_1\rho_1 = \mathtt{s}^k\gamma \leq^\infty_{\mathtt{A}} \beta\varphi_2$. We now prove (*) for all $\delta \in \mathtt{V}$, if $\delta\varphi_1 \neq \infty$, then $\delta\varphi_2 \neq \infty$. Indeed, if $\delta\varphi_2 = \infty$ then, since $\varphi_2\rho_2 \leq^\infty_{\mathtt{A}} \varphi_1$, we have $\delta\varphi_2\rho_2 = \infty \leq^\infty_{\mathtt{A}} \delta\varphi_1$ which is not possible since $\delta\varphi_1 \neq \infty$. Applying (*) with $\delta = \beta$, we get $\beta\varphi_2 = \mathtt{s}^{k+l}\gamma$ for some l, and $\gamma \in V_2$.

Similarly, ρ_2 is an injection from V_2 to V_1. So, by Lemma 27, there is a permutation $\xi : V \to V$ with $\xi|_{V_1} = \rho_1$. We now prove that, for all α, $\alpha\varphi_1\xi = \alpha\varphi_2$. If $\alpha \notin V$, this is immediate. Otherwise, we proceed by case on $\alpha\varphi_1$:

- $\alpha\varphi_1 = \infty$. Since $\alpha\varphi_1\rho_1 \leq^\infty_{\mathtt{A}} \alpha\varphi_2$, we have $\alpha\varphi_1\xi = \alpha\varphi_2 = \infty$.
- $\alpha\varphi_1 = \mathtt{s}^k\beta$. Then, $\beta \in V_1$ and $\alpha\varphi_1\xi = \mathtt{s}^k\beta\rho_1$. Since $\varphi_1\rho_1 \leq^\infty_{\mathtt{A}} \varphi_2$, we have $\mathtt{s}^k\beta\rho_1 \leq^\infty_{\mathtt{A}} \alpha\varphi_2$. By (*), we have $\alpha\varphi_2 \neq \infty$ since $\alpha\varphi_1 \neq \infty$. So, $\alpha\varphi_2 = \mathtt{s}^{k+l}\beta\rho_1$ for some l. Since $\varphi_2\rho_2 \leq^\infty_{\mathtt{A}} \varphi_1$, we have $\alpha\varphi_2\rho_2 = \mathtt{s}^{k+l}\beta\rho_1\rho_2 \leq^\infty_{\mathtt{A}} \mathtt{s}^k\beta$. Thus, $l = 0$ and $\alpha\varphi_1\xi = \alpha\varphi_2$.
- $\alpha\varphi_1 = \mathtt{s}^k\mathtt{c}$. Since $\varphi_1\rho_1 \leq^\infty_{\mathtt{A}} \varphi_2$, we have $\mathtt{s}^k\mathtt{c} \leq^\infty_{\mathtt{A}} \alpha\varphi_2$. By (*), we have $\alpha\varphi_2 \neq \infty$ since $\alpha\varphi_1 \neq \infty$. Hence, $\alpha\varphi_2 = \mathtt{s}^{k+l}\mathtt{c}$ for some l. Since $\varphi_2\rho_2 \leq^\infty_{\mathtt{A}} \varphi_1$, we have $\mathtt{s}^{k+l}\mathtt{c} \leq^\infty_{\mathtt{A}} \mathtt{s}^k\mathtt{c}$. Therefore, $l = 0$ and $\alpha\varphi_1\xi = \alpha\varphi_2$. ■

9.1 Satisfiability

To check whether a problem is satisfiable, we are going to introduce a terminating rewrite system that will put the problem into some normal form whose satisfiability is easy to establish. To do so, we first need to extend the successor algebra as follows:

Definition 25 (Successor-iterator algebra) Let $\mathtt{B}$ be the following multi-sorted algebra:

- Sorts: $\mathtt{A}$ interpreted by $\mathfrak{h}$, and $\mathtt{N}$ interpreted by ω.
- Function symbols: $0 : \mathtt{N}$ interpreted by 0, $\mathtt{s}_{\mathtt{N}} : \mathtt{N} \to \mathtt{N}$ and $\mathtt{s} : \mathtt{A} \to \mathtt{A}$ interpreted by the successor function, $\mathtt{c} : \mathtt{A}$ for every $\mathtt{c} \in \mathtt{C}$, $\mathtt{s} : \mathtt{N} \times \mathtt{A} \to \mathtt{A}$, with $\mathtt{s}(a,b)$ written $\mathtt{s}^a b$, interpreted as the iteration of the successor function: $(\mathtt{s}^a b)\mu = b\mu + a\mu$.

- Variables: the variables $\alpha, \beta, \ldots \in \mathtt{V}$ are of sort $\mathtt{A}$. In addition, we assume given a set $\mathtt{V}_\mathtt{N}$, disjoint from $\mathtt{V} \cup \mathtt{C}$, of variables $x, y, \ldots$ of sort $\mathtt{N}$, and an injection $\mathrm{x} : \mathtt{V} \to \mathtt{V}_\mathtt{N}$.
- $<_\mathtt{B} = <_\mathtt{A}$.
- $\leq_\mathtt{B} = <_\mathtt{A} \cup \simeq_\mathtt{A}$ where $\simeq_\mathtt{A}$ is the smallest congruence satisfying the following semantically valid equations on terms of sort $\mathtt{A}$:

$$\begin{array}{rcl} \mathtt{s}^0\alpha & \simeq_\mathtt{A} & \alpha \\ \mathtt{s}^{\mathtt{s}_\mathtt{N} x}\alpha & \simeq_\mathtt{A} & \mathtt{s}(\mathtt{s}^x\alpha) \\ \mathtt{s}^x(\mathtt{s}\alpha) & \simeq_\mathtt{A} & \mathtt{s}(\mathtt{s}^x\alpha) \end{array}$$

In the top-extension of $\mathtt{B}$, $\overline{\mathtt{B}}$, the symbol ∞ is of sort $\mathtt{A}$. Let $\mathrm{Var}_s(a)$ be the variables of sort s occurring in a. A problem is *constant-free* if it contains no constants $\mathtt{c} \in \mathtt{C}$.

Note that, in a multi-sorted algebra, substitutions map a variable of sort s to a term of sort s (hence a substitution cannot map a variable of sort $\mathtt{N}$ to ∞), and constraints are pairs of terms of the same sort. A problem is of sort s if all its constraints are of sort s.

One can easily check that, when oriented from left to right, the equations defining $\simeq_\mathtt{A}$ form a confluent and terminating rewrite system. Hence, every term has a unique normal form and two equivalent terms have the same normal form. So, wlog, we can always assume that terms are in normal form, in which case $\simeq_\mathtt{A}$ is the equality and $\leq_\mathtt{B}$ is $\leq_\mathtt{A}$.

In the following, we use the letters e and f (k and l resp.) to denote arbitrary (closed resp.) terms of sort $\mathtt{N}$. Closed terms of sort $\mathtt{N}$ are isomorphic to natural numbers. Hence, we identify $\mathtt{s}_\mathtt{N} \ldots \mathtt{s}_\mathtt{N} \mathtt{0}$ (k times $\mathtt{s}_\mathtt{N}$) with k, denote $\mathtt{s}_\mathtt{N} \ldots \mathtt{s}_\mathtt{N} x$ (k times $\mathtt{s}_\mathtt{N}$) by $x+k$, and call a problem of sort $\mathtt{N}$ an *integer problem*. However, $\mathtt{s}^k\alpha$ will not denote $\mathtt{s}^{\mathtt{s}_\mathtt{N} \cdots \mathtt{s}_\mathtt{N} \mathtt{0}}\alpha$ (k times $\mathtt{s}_\mathtt{N}$) but its normal form $\mathtt{s} \ldots \mathtt{s}\alpha$ (k times $\mathtt{s}$).

Given a problem P in $\overline{\mathtt{A}}$, since $\mathrm{Sol}_{\overline{\mathtt{B}}}(P)$ may contain solutions not expressible in $\overline{\mathtt{A}}$, we consider the following subset instead:

Definition 26 ($\mathtt{N}$-closed solutions) A term $a \in \overline{\mathtt{B}}$ is $\mathtt{N}$-closed if $\mathrm{Var}_\mathtt{N}(a) = \emptyset$. A solution to a problem P is $\mathtt{N}$-closed if it maps every $\alpha \in \mathrm{Var}(P)$ to an $\mathtt{N}$-closed term. Let $\mathrm{Sol}^\emptyset_{\overline{\mathtt{B}}}(P)$ ($\mathrm{Sol}^\emptyset_\mathtt{B}(P)$ resp.) be the set of all the $\mathtt{N}$-closed (finite resp.) solutions of P.

Lemma 29

- A term of sort $\mathtt{A}$ belongs to $\overline{\mathtt{A}}$ iff it is $\mathtt{N}$-closed.
- Given a problem P in $\overline{\mathtt{A}}$, $\mathrm{Sol}_{\overline{\mathtt{A}}}(P) = \mathrm{Sol}^\emptyset_{\overline{\mathtt{B}}}(P)$.

Proof.

- This is immediate if $a = \infty$. Otherwise, $a = \mathtt{s} \ldots \mathtt{s}\mathtt{s}^{x_1} \ldots \mathtt{s}^{x_n} b$ with $b \in \mathtt{V} \cup \mathtt{C}$. If $a \in \mathtt{A}$, then $n = 0$ and a is $\mathtt{N}$-closed. Conversely, if a is $\mathtt{N}$-closed, then $n = 0$ and $a \in \mathtt{A}$.
- Immediate consequence of the previous property. ■

Note that a $\mathtt{N}$-closed solution maps every variable of sort $\mathtt{N}$ to an integer. Hence, for an integer problem P, $\mathrm{Sol}^\emptyset_{\overline{\mathtt{B}}}(P) = \mathrm{Sol}^\emptyset_\mathtt{B}(P)$ (solutions to integer problems are always finite) and, given $\varphi, \psi \in \mathrm{Sol}^\emptyset_\mathtt{B}(P)$, $\varphi \sqsubseteq \psi$ iff $\varphi \leq_\mathbb{N} \psi$, *i.e.* for all $x \in \mathrm{Var}(P)$, $x\varphi \leq_\mathbb{N} x\psi$.

Now, to a problem in $\overline{\mathtt{B}}$, we associate a graph as follows:

Definition 27 (Graph associated to a problem in $\overline{\mathtt{B}}$) Let $\mathtt{H} = \mathtt{V} \cup \mathtt{V_N} \cup \mathtt{C} \cup \{\mathtt{0}\}$. To a problem P in $\overline{\mathtt{B}}$, we associate a directed graph $G(P)$ on $\mathtt{H} \cup \{\infty\}$ with the following labeled edges:

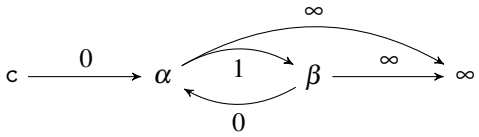
- $x \xrightarrow{k-l} y$ for each constraint $x+k \leq^? y+l \in P$ with $x,y \in \mathtt{V_N} \cup \{\mathtt{0}\}$;
- $\mathtt{0} \xrightarrow{0} y$ for each variable $y \in \mathrm{Var}_{\mathtt{N}}(P)$;
- $\alpha \xrightarrow{k-l} \beta$ for each constraint $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in P$;
- $\alpha \xrightarrow{\infty} \infty$ for each $\alpha \in \mathrm{Var}_{\mathtt{A}}(P)$;
- $\infty \xrightarrow{0} \beta$ for each constraint $\infty \leq^? \mathtt{s}^l\beta \in P$;
- $\mathtt{c} \xrightarrow{0} \beta$ for each constraint $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in P$.

The *weight* of a path $a_1 \xrightarrow{k_1} \ldots \xrightarrow{k_n} a_{n+1}$ is $\Sigma_{i=1}^n k_i$, where $k+\infty = \infty$. A *cycle* (*i.e.* when $a_{n+1} = a_1$) is *positive* if its weight is > 0.

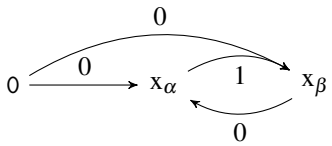
Let $\leq_P$ be the smallest quasi-ordering on $\mathtt{H}$ (we exclude ∞) such that $a \leq_P b$ iff there is a path from a to b in $G(P)$.

A triple $(\alpha, \mathtt{c}, \mathtt{d})$ such that $\mathtt{c} \leq_P \alpha$, $\mathtt{d} \leq_P \alpha$ and $\mathtt{c} \neq \mathtt{d}$, is called *incompatible*.

For instance, the graph of the problem $P = \{\mathtt{c} \leq^? \alpha, \mathtt{s}\alpha \leq^? \beta, \beta \leq^? \alpha\}$ is:



If we replace α by $\mathrm{x}_\alpha \in \mathtt{V_N}$, β by $\mathrm{x}_\beta \in \mathtt{V_N}$ and $\mathtt{c}$ by $\mathtt{0}$, we get the integer problem $I(P) = \{0 \leq^? \mathrm{x}_\alpha, \mathrm{x}_\alpha + 1 \leq^? \mathrm{x}_\beta, \mathrm{x}_\beta \leq^? \mathrm{x}_\alpha\}$ whose graph is:



Following Pratt (Pratt, 1977), an integer problem P has an integer solution iff $G(P)$ has no positive cycles, which 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".

In the graph of $I(P)$, the cycle $\mathrm{x}_\alpha \xrightarrow{1} \mathrm{x}_\beta \xrightarrow{0} \mathrm{x}_\alpha$ has weight 1 and thus is positive. So, $I(P)$ has no integer solution. On the other hand, P can be solved by taking $\alpha = \beta = \infty$.

Next, we introduce a data structure used to transform an arbitrary problem into a problem in normal form using the rules of Figure 8:

Definition 28 (Configuration) A term is admissible if it contains at most one variable. A constraint $a \leq^? b$ is admissible if both a and b are admissible.

A configuration C is $\bot$ or a tuple $(C_0, C_1, C_2, C_3, C_4)$ with:

- $C_0 \subseteq \mathtt{V}$,
- $C_1 \subseteq \mathtt{V}$,
- C_2 is a finite map from $\mathtt{V}$ to $\mathtt{C}$,

- C_3 is a set of admissible constraints of sort $\mathtt{N}$,
- C_4 is a set of admissible constraints of sort $\mathtt{A}$,
- C_0, C_1, $\mathrm{dom}(C_2)$ and $\mathrm{Var}(C_4)$ are pairwise disjoint,
- $\mathrm{Var}_{\mathtt{N}}(C_3) = \{\mathrm{x}_\alpha \mid \alpha \in \mathrm{dom}(C_2)\}$,
- $\mathrm{Var}_{\mathtt{N}}(C_4) \subseteq \{\mathrm{x}_\alpha \mid \alpha \in \mathrm{dom}(C_2)\}$.

Let $\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C) = \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(\pi(C))$ and $\mathrm{Var}(C) = \mathrm{Var}(\pi(C))$, where $\pi(\bot) = \bot$ and $\pi(C_0,\dots,C_4)$ is the union of:

- $\{\alpha \leq^? \infty \mid \alpha \in C_0\}$,
- $\{\infty \leq^? \alpha \mid \alpha \in C_1\}$,
- $\{\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \mid (\alpha,\mathtt{c}) \in C_2\} \cup \{\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \leq^? \alpha \mid (\alpha,\mathtt{c}) \in C_2\}$,
- $C_3 \cup C_4$.

C is *normal* if there is no D such that $C \rightsquigarrow D$ where $\rightsquigarrow$ is defined in Figure 8.

Finally, given C and ψ, let:

- $\sigma_0(C,\psi) = \{(\alpha, \alpha\psi) \mid \alpha \in C_0\}$,
- $\sigma_1(C) = \{(\alpha,\infty) \mid \alpha \in C_1\}$,
- $\sigma_2(C,\psi) = \{(\alpha, \mathtt{s}^{\mathrm{x}_\alpha\psi}\mathtt{c}) \mid (\alpha,\mathtt{c}) \in C_2\}$,
- $\sigma_{3,4}(C,\psi) = \{(\alpha,\alpha\psi) \mid \alpha \in \mathrm{Var}(C_3 \cup C_4)\}$,
- $\sigma_{4_{\mathtt{A}}}(C,\psi) = \{(\alpha,\alpha\psi) \mid \alpha \in \mathrm{Var}_{\mathtt{A}}(C_4)\}$.

C_0 records the variables with no constraints, C_1 records the variables that must be set of ∞, C_2 records the variables that must be set to a value of the form $\mathtt{s}^k\mathtt{c}$, C_3 contains the constraints on integer variables, and C_4 contains all the other constraints.

Note that Figure 8 describes an infinite set of rules since a stands for an arbitrary size expression of sort $\mathtt{A}$, e and f for arbitrary size expressions of sort $\mathtt{N}$, k for an arbitrary natural number, α for an arbitrary size variable of sort $\mathtt{A}$, $\mathtt{c}$ and $\mathtt{d}$ for arbitrary constants, and $P \uplus Q$ for an arbitrary set with two disjoint parts, P and Q.

($\_\infty$) removes the constraints of the form $a \leq^? \infty$ that are always satisfied, and records in C_0 variables not occurring elsewhere.

($\infty\alpha_1$) detects variables that must be set to ∞ because they belong to a positive cycle.

($\infty\alpha_2$) detects variables α that must be set to ∞ because some constraints imply that it should otherwise be set to a term of the form $\mathtt{s}^k\mathtt{c}$ and some other constraints that it should be set to a term of the form $\mathtt{s}^l\mathtt{d}$ with $\mathtt{c} \neq \mathtt{d}$.

($\infty\mathtt{c}$) detects an unsatisfiable constraint of the form $\infty \leq^? \mathtt{s}^e\mathtt{c}$.

($\mathtt{cd}$) detects an unsatisfiable constraint of the form $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^f\mathtt{d}$ with $\mathtt{c} \neq \mathtt{d}$.

($\mathtt{cc}$) replaces a constraint of the form $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^f\mathtt{c}$ by the integer constraint $e \leq^? f$.

($\alpha\mathtt{c}$) replaces a constraint of the form $\mathtt{s}^k\alpha \leq^? \mathtt{s}^e\mathtt{c}$ by recording in C_2 that α must be set to a term of the form $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$, propagating it in other constraints, and recording in integer constraints the fact that $\mathrm{x}_\alpha + k \leq^? e$.

The rule ($\infty\alpha_2$) is not necessary for deciding the satisfiability of a problem. It is included here because it is useful to compute a most general solution in next section.

Fig. 8. Rules for computing the normal form of a problem

$$
\begin{array}{lrcl}
(\_\infty) & C_0,C_1,C_2,C_3,C_4 \uplus \{a \leq^? \infty\} & \leadsto & C_0 \cup (\mathrm{Var}_{\mathtt{A}}(a) - \mathrm{Var}(C_4)),C_1,C_2,C_3,C_4 \\
(\infty\alpha_1) & C_0,C_1,C_2,C_3,C_4 \uplus Q & \leadsto & C_0,C_1 \cup \mathrm{Var}(Q),C_2, \\
 & & & \quad C_3,C_4\{(\alpha,\infty) \mid \alpha \in \mathrm{Var}(Q)\} \\
 & & \text{if} & Q \text{ is constant-free and } G(Q) \text{ is a positive cycle} \\
(\infty\alpha_2) & C_0,C_1,C_2,C_3,C_4 & \leadsto & C_0,C_1 \cup \{\alpha\},C_2,C_3,C_4\{(\alpha,\infty)\} \\
 & & & \text{if } \mathtt{c} \leq_{C_4} \alpha, \mathtt{d} \leq_{C_4} \alpha, \mathtt{c} \neq \mathtt{d} \\
(\infty\mathtt{c}) & C_0,C_1,C_2,C_3,C_4 \uplus \{\infty \leq^? \mathtt{s}^e\mathtt{c}\} & \leadsto & \bot \\
(\mathtt{cd}) & C_0,C_1,C_2,C_3,C_4 \uplus \{\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^f\mathtt{d}\} & \leadsto & \bot \text{ if } \mathtt{c} \neq \mathtt{d} \\
(\mathtt{cc}) & C_0,C_1,C_2,C_3,C_4 \uplus \{\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^f\mathtt{c}\} & \leadsto & C_0,C_1,C_2,C_3 \cup \{e \leq^? f\},C_4 \\
(\alpha\mathtt{c}) & C_0,C_1,C_2,C_3,C_4 \uplus \{\mathtt{s}^k\alpha \leq^? \mathtt{s}^e\mathtt{c}\} & \leadsto & C_0,C_1,C_2 \cup \{(\alpha,\mathtt{c})\}, \\
 & & & \quad C_3 \cup \{\mathrm{x}_\alpha + k \leq^? e\},C_4\{(\alpha,\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c})\}
\end{array}
$$

Lemma 30

1. $\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C) = \{\sigma_0(C,\varphi) \cup \sigma_1(C) \cup \sigma_2(C,\psi) \cup \sigma_{3,4}(C,\psi) \mid \varphi\, \mathtt{N}\text{-closed}, \psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C_3 \cup C_4)\}$.
2. For all problems P in $\overline{\mathtt{A}}$, $(\emptyset,\emptyset,\emptyset,\emptyset,P)$ is a configuration and $\mathrm{Sol}_{\overline{\mathtt{A}}}(P) = \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(\emptyset,\emptyset,\emptyset,\emptyset,P)$.
3. In a configuration, every term of sort $\mathtt{A}$ is of the form ∞, $\mathtt{s}^\alpha$ or $\mathtt{s}^e\mathtt{c}$.
4. If C is a configuration and $C \leadsto D$, then:
   (a) D is a configuration.
   (b) If $D \neq \bot$, then $\mathrm{Var}(C) \subseteq \mathrm{Var}(D)$.
   (c) Correctness: if $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(D)$, then $\varphi|_{\mathrm{Var}(C)} \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C)$.
   (d) Completeness: if $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C)$, then $\psi = \varphi|_{\mathrm{Var}(C)}$ for some $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(D)$.
5. The relation $\leadsto$ terminates.
6. If $(\emptyset,\emptyset,\emptyset,\emptyset,P) \leadsto^* C \neq \bot$, then $\mathrm{Var}(C) = \mathrm{Var}(P) \cup \mathrm{Var}(C_3)$.

Proof.

1. Let $S(C) = \{\sigma_0(C,\varphi) \cup \sigma_1(C) \cup \sigma_2(C,\psi) \cup \sigma_{3,4}(C,\psi) \mid \varphi\, \mathtt{N}\text{-closed}, \psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C_3 \cup C_4)\}$. One can easily check that $S(C) \subseteq \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C)$. Assume now that $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C)$. Then, $\varphi = \sigma_0(C,\varphi) \cup \sigma_1(C) \cup \sigma_2(C,\psi) \cup \sigma_{3,4}(C,\psi)$ where $\psi = \varphi|_{\mathrm{Var}(C_3 \cup C_4)}$. Indeed, if $\alpha \in C_1$, then $\infty \leq^? \alpha \in \pi(C)$. Hence, $\alpha\varphi = \infty$. Now, if $(\alpha,\mathtt{c}) \in C_2$, then $\pi(C)$ contains $\alpha \leq^? \mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$ and $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c} \leq^? \alpha$. Hence, $\alpha\varphi = \mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c}$ and $\mathrm{x}_\alpha\varphi = \mathrm{x}_\alpha\psi$ since $\{\mathrm{x}_\alpha \mid \alpha \in \mathrm{dom}(C_2)\} \subseteq \mathrm{Var}_{\mathtt{N}}(C_3)$.
2. One can easily check that $(\emptyset,\emptyset,\emptyset,\emptyset,P)$ is a configuration. The fact that $\mathrm{Sol}_{\overline{\mathtt{A}}}(P) = \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(\emptyset,\emptyset,\emptyset,\emptyset,P)$ directly follows from the previous property.
3. Straightforward.

4(a) One can easily check that all the conditions defining what is a configuration are preserved by each rule. In particular, $(\alpha\mathtt{c})$ replaces α by $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$, hence every term of D is admissible if every term of C so is.

(b) Straightforward.

(c) Straightforward.

(d) We only detail the following cases:

- Rule $(\alpha\mathtt{c})$. We have $(\mathtt{s}^k\alpha)\psi \leq^\infty_{\mathtt{A}} (\mathtt{s}^e\mathtt{c})\psi = \mathtt{s}^{e\psi}\mathtt{c}$. So, $\alpha\psi \neq \infty$ and $(\mathtt{s}^k\alpha)\psi = \mathtt{s}^k(\alpha\psi) \leq_{\mathtt{A}} \mathtt{s}^{e\psi}\mathtt{c}$. By Lemma 25, there is l such that $\mathtt{s}^{e\psi}\mathtt{c} = \mathtt{s}^l\mathtt{s}^k(\alpha\psi)$. Hence, there is m such that $\alpha\psi = \mathtt{s}^m\mathtt{c}$ and $e\psi = l+k+m$. Let now $\varphi = \psi \cup \{(\mathrm{x}_\alpha, m)\}$. We have $\alpha\varphi = \alpha\psi = \mathtt{s}^m\mathtt{c} = (\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c})\varphi$ and $(\mathrm{x}_\alpha + k)\varphi = m+k \leq l+k+m = e\psi = e\varphi$. Therefore, $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(D)$ and $\varphi|_{\mathrm{Var}(C)} = \psi$.
- Rule $(\infty\alpha_1)$. We first prove that, if $\alpha_1 \xrightarrow{k_1} \ldots \xrightarrow{k_n} \alpha_{n+1}$ is a path in $G(Q)$, $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(Q)$ and $k = \Sigma_{i=1}^n k_i \geq 0$ ($k < 0$ resp.), then $\mathtt{s}^k\alpha_1\psi \leq^\infty_{\mathtt{A}} \alpha_{n+1}\psi$ ($\alpha_1\psi \leq^\infty_{\mathtt{A}} \mathtt{s}^{-k}\alpha_{n+1}\psi$ resp.) (*), by induction on n. If $n = 1$, this is immediate. We now prove it for $n+1$.
  - Case $k \geq 0$. By induction hypothesis, $\mathtt{s}^k\alpha_1\psi \leq^\infty_{\mathtt{A}} \alpha_{n+1}\psi$.
    - Case $k_{n+1} \geq 0$. Then, $\mathtt{s}^{k_{n+1}}\alpha_{n+1}\psi \leq^\infty_{\mathtt{A}} \alpha_{n+2}\psi$.
      - Case $k + k_{n+1} \geq 0$. By monotony and transitivity, $\mathtt{s}^{k+k_{n+1}}\alpha_1\psi \leq^\infty_{\mathtt{A}} \alpha_{n+2}\psi$.
      - Case $k + k_{n+1} < 0$. Impossible.
    - Case $k_{n+1} < 0$. Then, $\alpha_{n+1}\psi \leq^\infty_{\mathtt{A}} \mathtt{s}^{-k_{n+1}}\alpha_{n+2}\psi$ and, by transitivity, $\mathtt{s}^k\alpha_1\psi \leq^\infty_{\mathtt{A}} \mathtt{s}^{-k_{n+1}}\alpha_{n+2}\psi$.
      - Case $k + k_{n+1} \geq 0$. Since $-k_{n+1} \leq k$, $\mathtt{s}^{k+k_{n+1}}\alpha_1\psi \leq^\infty_{\mathtt{A}} \alpha_{n+2}\psi$.
      - Case $k + k_{n+1} < 0$. Since $k < -k_{n+1}$, $\alpha_1\psi \leq^\infty_{\mathtt{A}} \mathtt{s}^{-k-k_{n+1}}\alpha_{n+2}\psi$.
  - Case $k < 0$. Symmetric to previous case.

  Assume now that Q is constant-free and $G(Q)$ is a positive cycle. If $G(Q)$ contains ∞, then $\alpha\psi = \infty$ for all $\alpha \in \mathrm{Var}(Q)$. Otherwise, $G(Q)$ is $\alpha_1 \xrightarrow{k_1} \ldots \xrightarrow{k_n} \alpha_{n+1} = \alpha_1$. Hence, $\mathtt{s}^k\alpha_1\psi \leq^\infty_{\mathtt{A}} \alpha_1\psi$ with $k = \Sigma_{i=1}^n k_i > 0$. Therefore, $\alpha_1\psi = \infty$ and $\alpha\psi = \infty$ for all $\alpha \in \mathrm{Var}(Q)$.
- Rule $(\infty\alpha_2)$. We first prove that (a) for any problem P, if $\beta \leq_P \alpha$ by a path of length n, $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ and $\beta\varphi = \infty$, then $\alpha\varphi = \infty$, by induction on n. If $n = 0$, this is immediate. Otherwise, there is $\mathtt{s}^p\beta \leq^? \mathtt{s}^q\gamma \in P$ with $\gamma \leq_P \alpha$ by a path of length $n-1$. Since $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ and $\beta\varphi = \infty$, we have $\infty \leq^\infty_{\mathtt{A}} \mathtt{s}^q\gamma\varphi$. Therefore $\gamma\varphi = \infty$ and, by induction hypothesis, $\alpha\varphi = \infty$.
  We now prove that (b) if $\beta \leq_P \alpha$ by a path of length n, $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ and $\beta\varphi = \mathtt{s}^k\mathtt{c}$ for some k, then either $\alpha\varphi = \infty$ or $\alpha\varphi = \mathtt{s}^i\mathtt{c}$ for some i, by induction on n. If $n = 0$, this is immediate. Otherwise, there is $\mathtt{s}^p\beta \leq^? \mathtt{s}^q\gamma \in P$ with $\gamma \leq_P \alpha$ by a path of length $n-1$. Since $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ and $\beta\varphi = \mathtt{s}^k\mathtt{c}$, we have $\mathtt{s}^{p+k}\mathtt{c} \leq^\infty_{\mathtt{A}} \mathtt{s}^q\gamma\varphi$. If $\gamma\varphi = \infty$ then, by (a), $\alpha\varphi = \infty$. Otherwise, $\gamma\varphi = \mathtt{s}^l\mathtt{c}$ for some l and, by induction hypothesis, either $\alpha\varphi = \infty$ or $\alpha\varphi = \mathtt{s}^i\mathtt{c}$ for some i.
  Hence, if $(\alpha, \mathtt{c}, \mathtt{d})$ is incompatible in C_4 and $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C_4)$, then $\alpha\varphi = \infty$.

5. Every rule decreases the number of constraints in C_4 except rule $(\infty\alpha_2)$. In $(\infty\alpha_2)$, this number is unchanged but the number of variables decreases. Since the number of variables in C_4 never increases, the system terminates.
6. Straightforward. ■

The properties 4(c) and 4(d) give $\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C) = \{\varphi|_{\mathrm{Var}(C)} \mid \varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(D)\}$ whenever $C \rightsquigarrow D$.

Definition 29 (Affine problem) A constraint is *affine* if it is of sort $\mathtt{N}$, of the form $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta$ or of the form $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta$. A problem is affine if all its constraints are affine.

Lemma 31 In any normal configuration $C \neq \bot$, C_4 is an affine problem with no positive cycles and no incompatible triples.

Proof. By Lemma 30, every term of sort $\mathtt{A}$ occurring in C is of the form ∞, $\mathtt{s}^k\alpha$ or $\mathtt{s}^e\mathtt{c}$. Now, C_4 cannot contain a constraint of the form:

- $a \leq^? \infty$ because of rule $(\_\infty)$,
- $\infty \leq^? \mathtt{s}^l\beta$ because of rule $(\infty\alpha_1)$,
- $\infty \leq^? \mathtt{s}^f\mathtt{d}$ because of rule $(\infty\mathtt{c})$,
- $\mathtt{s}^k\alpha \leq^? \mathtt{s}^f\mathtt{d}$ because of rule $(\alpha\mathtt{c})$,
- $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^f\mathtt{d}$ because of rules $(\mathtt{cc})$ and $(\mathtt{cd})$.

Therefore, a constraint in C_4 can only be either of the form $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta$ or of the form $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta$. Moreover, $G(C_4)$ cannot have positive cycles because of rule $(\infty\alpha_1)$, and cannot have incompatible triples because of rule $(\infty\alpha_2)$. ■

Since affine problems of sort $\mathtt{A}$ are always satisfiable (by setting their variables to ∞), we can conclude:

Theorem 4 (Satisfiability) The satisfiability of a size problem in the successor algebra is decidable in polynomial time wrt. the number of symbols by the algorithm of Figure 9.

Proof. Let $|P|$ be the number of symbols in P. Constructing $G(P)$ requires at most $\sharp\mathrm{Var}(P) + \sharp P$ steps, where $\sharp X$ is the cardinal of X. But $\mathrm{Var}(P) \leq 2\sharp P$ since there are at most 2 variables per constraint, and $2\sharp P \leq |P|$ since every constraint is of size 2 at least. Therefore, constructing $G(P)$ requires at most $3|P|/2$ steps.

Whether there is a positive cycle in a graph is decidable in polynomial time (Pratt, 1977). Whether there is an incompatible triple in a graph can be done in polynomial time too. Hence, whether a rule can be applied is decidable in polynomial time. Now, since $\rightsquigarrow$ terminates, the algorithm describes a computable function.

We now prove that it is correct and complete. If $C = \bot$ then, by completeness, P is unsatisfiable. Otherwise, $C = (C_0, C_1, C_2, C_3, C_4)$. If $G(C_3)$ has a positive cycle then, by completeness, P is unsatisfiable. Otherwise, let $\varphi_3 \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C_3)$. Then, one can easily check that $\varphi = \varphi_3 \cup \{(\alpha,\infty) \mid \alpha \in \mathrm{Var}(C_1) \cup \mathrm{Var}(C_4)\} \cup \{(\alpha, \mathtt{s}^{\mathrm{x}_\alpha\varphi_3}\mathtt{c}) \mid (\alpha,\mathtt{c}) \in C_2\} \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(C)$. Therefore, by correctness, $\varphi|_{\mathrm{Var}(P)} \in \mathrm{Sol}_{\mathtt{B}}(P) = \mathrm{Sol}_{\mathtt{A}}(P)$ and P is satisfiable.

Finally, to prove that the complexity for computing C is polynomial, it suffices to show that the number of rewrite steps and the size $|C| = |\pi(C)|$ of intermediate configurations C are polynomially bounded by $|P|$.

By definition of $\rightsquigarrow$, $\mathrm{Var}(C) \subseteq \mathrm{Var}(P) \cup \{\mathrm{x}_\alpha \mid \alpha \in \mathrm{Var}(P)\}$ and $\sharp\mathrm{Var}(C) \leq 2\sharp\mathrm{Var}(P) \leq 2|P|$. So, after the termination proof, the number of rewrite steps is $\leq \sharp P \times 2|P| \leq |P|^2$.

Let $\|C\|_\infty$ be the maximum size of a constraint in $\pi(C)$. No rule but $(\alpha\mathtt{c})$ can make $\|C\|_\infty$ increase. $\|C\|_\infty$ can be increased by at most 2 for each replacement of a variable α by $\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}$. However, there cannot be more than two such replacements in a constraint since, after two such replacements, there is no variable of sort $\mathtt{A}$ anymore. Therefore, $\|C\|_\infty \leq \|P\|_\infty + 4 \leq |P| + 4$. Now, $\sharp\pi(C) \leq 5\sharp P \leq 5|P|/2$ since $\sharp C_0 + \sharp C_1 + \sharp C_2 \leq 2\sharp\mathrm{Var}(P) \leq 4\sharp P$ and $\sharp C_3 + \sharp C_4 \leq \sharp P$. Therefore, $|C| \leq \|C\|_\infty \times \sharp C \leq (|P|+4) \times 5|P|/2$. ■

Fig. 9. Algorithm for deciding the satisfiability of a problem P in the successor algebra.

1. Compute a normal form C of $(\emptyset,\emptyset,\emptyset,\emptyset,P)$ wrt the rules of Figure 8.
2. If $C=\bot$, then P is not satisfiable. Otherwise, $C=(C_0,C_1,C_2,C_3,C_4)$.
3. If C_3 has a positive cycle, then P is not satisfiable. Otherwise, P is satisfiable.

Our procedure can be related to the one described in (Barthe *et al.*, 2005) where, like 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 as we shall see in next section.

Example 11 Let $P=\{\mathtt{c}\leq^?\alpha,\mathtt{s}\alpha\leq^?\beta,\beta\leq^?\alpha,\mathtt{d}\leq^?\beta\}$. We have $(\emptyset,\emptyset,\emptyset,\emptyset,P)$
$\rightsquigarrow(\emptyset,\{\alpha\},\emptyset,\emptyset,\{\mathtt{c}\leq^?\infty,\infty\leq^?\beta,\beta\leq^?\infty,\mathtt{d}\leq^?\beta\})$, by $(\infty\alpha_2)$ since $\mathtt{c}\leq_P\alpha$ and $\mathtt{d}\leq_P\alpha$;
$\rightsquigarrow(\emptyset,\{\alpha,\beta\},\emptyset,\emptyset,\{\mathtt{c}\leq^?\infty,\infty\leq^?\mathtt{d}\})$, by $(\infty\alpha_1)$ since $\infty\xrightarrow{0}\beta\xrightarrow{\infty}\infty$ is positive;
$\rightsquigarrow\bot$, by $(\infty\mathtt{c})$. ■

Example 12 Let $P=\{\alpha\leq^?\mathtt{sc},\beta\leq^?\alpha\}$. We have $(\emptyset,\emptyset,\emptyset,\emptyset,P)$
$\rightsquigarrow(\emptyset,\emptyset,\{(\alpha,\mathtt{c})\},\{\mathrm{x}_\alpha\leq^?1\},\{\beta\leq^?\mathtt{s}^{\mathrm{x}_\alpha}\mathtt{c}\})$, by $(\alpha\mathtt{c})$;
$\rightsquigarrow(\emptyset,\emptyset,\{(\alpha,\mathtt{c}),(\beta,\mathtt{c})\},\{\mathrm{x}_\alpha\leq^?1,\mathrm{x}_\beta\leq^?\mathrm{x}_\alpha\},\emptyset)$, by $(\alpha\mathtt{c})$ again. This is a normal form and the graph of $\{\mathrm{x}_\alpha\leq^?1,\mathrm{x}_\beta\leq^?\mathrm{x}_\alpha\}$ has no positive cycle, so it is satisfiable (the solutions for $(\mathrm{x}_\alpha,\mathrm{x}_\beta)$ are $(0,0)$, $(1,0)$ and $(1,1)$). ■

9.2 Computing the most general solution

We now turn to the problem of whether, in the successor algebra $\overline{\mathtt{A}}$, a satisfiable problem P has a most general solution and, if so, how to compute it.

Let $\mathrm{mgs}_{\overline{\mathtt{A}}}(P)$ ($\mathrm{mgs}_{\mathtt{A}}(P)$ resp.) be the set of most general (finite resp.) solutions of P, and $\mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ ($\mathrm{mgs}^{\emptyset}_{\mathtt{B}}(C)$ resp.) be the set of most general (finite resp.) $\mathtt{N}$-closed solutions of C.

We first prove a refinement of Lemma 26 to $\mathtt{N}$-closed solutions of a configuration:

Lemma 32 Given $\varphi,\psi\in\mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$, $\varphi\sqsubseteq\psi$ iff there is $\rho:\mathtt{V}\to\mathtt{V}\cup\mathtt{C}\cup\{\infty\}$ such that $\mathrm{dom}(\rho)\subseteq\mathrm{Var}_{\mathtt{A}}(C)$ and, for all $\alpha\in C_0\cup\mathrm{Var}(C_3)\cup\mathrm{Var}_{\mathtt{A}}(C_4)$, $\alpha\varphi\rho\leq^{\infty}_{\mathtt{A}}\alpha\psi$.

Proof.

- $\Rightarrow$ By Lemma 26, there is $\rho:\mathtt{V}\cup\mathtt{V}_{\mathtt{N}}\to\mathtt{V}\cup\mathtt{V}_{\mathtt{N}}\cup\mathtt{C}\cup\{0,\infty\}$ such that $\varphi\rho\leq^{\infty}_{\mathtt{A}}\psi$. Since φ and ψ are $\mathtt{N}$-closed, we also have $\varphi(\rho|_{\mathtt{V}})\leq^{\infty}_{\mathtt{A}}\psi$. Indeed, if $\alpha\in\mathtt{V}_{\mathtt{N}}$, then $\alpha\varphi(\rho|_{\mathtt{V}})=\alpha\varphi=\alpha\varphi\rho\leq^{\infty}_{\mathtt{A}}\alpha\psi$. Let now $\alpha\notin\mathrm{Var}(C)$. Then, $\alpha(\rho|_{\mathtt{V}})=\alpha\varphi(\rho|_{\mathtt{V}})\leq^{\infty}_{\mathtt{A}}\alpha\psi=\alpha$. Therefore, $\alpha(\rho|_{\mathtt{V}})=\alpha$ and $\mathrm{dom}(\rho|_{\mathtt{V}})\subseteq\mathrm{Var}_{\mathtt{A}}(C)$.
- $\Leftarrow$ After Lemma 26, it is enough to prove that, for all $\alpha\in\mathtt{V}\cup\mathtt{V}_{\mathtt{N}}$, $\alpha\varphi\rho\leq^{\infty}_{\mathtt{A}}\alpha\psi$. By assumption, the property holds if $\alpha\in C_0\cup\mathrm{Var}(C_3)\cup\mathrm{Var}_{\mathtt{A}}(C_4)$. If $\alpha\in C_1$, then $\alpha\varphi=\infty=\alpha\psi$ and $\alpha\varphi\rho\leq^{\infty}_{\mathtt{A}}\alpha\psi$. If $(\alpha,\mathtt{c})\in C_2$, then $\alpha\varphi=\mathtt{s}^{\mathrm{x}_\alpha\varphi}\mathtt{c}$, $\alpha\psi=\mathtt{s}^{\mathrm{x}_\alpha\psi}\mathtt{c}$. Since $\mathrm{x}_\alpha\in\mathrm{Var}(C_3)$ and φ is $\mathtt{N}$-closed, we have $\mathrm{x}_\alpha\varphi=\mathrm{x}_\alpha\varphi\rho\leq^{\infty}_{\mathtt{A}}\mathrm{x}_\alpha\psi$. Therefore, $\alpha\varphi\rho\leq^{\infty}_{\mathtt{A}}\alpha\psi$. Since $\mathrm{Var}_{\mathtt{N}}(C_4)\subseteq\mathrm{Var}(C_3)$, we are left with the case where $\alpha\notin\mathrm{Var}(C)$. But, in

this case, $\alpha\varphi = \alpha\psi = \alpha\rho = \alpha$ since $\mathrm{dom}(\varphi)$, $\mathrm{dom}(\psi)$ and $\mathrm{dom}(\rho)$ are all included in $\mathrm{Var}(C)$. ■

We now prove that the most general solutions of a problem P in $\overline{\mathtt{A}}$ can be obtained from the most general $\mathtt{N}$-closed solutions of the normal form of $(\emptyset,\emptyset,\emptyset,\emptyset,P)$.

Lemma 33 Assume that $(\emptyset,\emptyset,\emptyset,\emptyset,P) \leadsto^* C$.

- Correctness: if $\varphi \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C)$, then $\varphi|_{\mathrm{Var}(P)} \in \mathrm{mgs}_{\overline{\mathtt{A}}}(P)$.
- Completeness: if $\psi \in \mathrm{mgs}_{\overline{\mathtt{A}}}(P)$, then there is $\varphi \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ such that $\varphi|_{\mathrm{Var}(P)} = \psi$.

Proof. Note that $\mathrm{Var}(C) = \mathrm{Var}(P) \cup \mathrm{Var}(C_3)$.

- Let $\varphi \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C)$. By correctness of $\leadsto$, $\varphi|_{\mathrm{Var}(P)} \in \mathrm{Sol}_{\overline{\mathtt{A}}}(P)$. Let now $\psi \in \mathrm{Sol}_{\overline{\mathtt{A}}}(P)$. By completeness of $\leadsto$, there is $\varphi' \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ such that $\psi = \varphi'|_{\mathrm{Var}(P)}$. Since $\varphi = \mathrm{mgs}(C)$, $\varphi \sqsubseteq \varphi'$. By Lemma 32, $\varphi\rho \leq^{\infty}_{\mathtt{A}} \varphi'$ for some $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{C} \cup \{\infty\}$ such that $\mathrm{dom}(\rho) \subseteq \mathrm{Var}_{\mathtt{A}}(C) = \mathrm{Var}(P)$. Therefore, for all $\alpha \in \mathtt{V} \cup \mathtt{V}_{\mathtt{N}}$, $\alpha\varphi|_{\mathrm{Var}(P)}\rho \leq^{\infty}_{\mathtt{A}} \alpha\varphi'|_{\mathrm{Var}(P)}$ and $\varphi|_{\mathrm{Var}(P)} \sqsubseteq \varphi'|_{\mathrm{Var}(P)} = \psi$.
- Let $\psi \in \mathrm{mgs}_{\overline{\mathtt{A}}}(P)$. By completeness of $\leadsto$, there is $\varphi \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ such that $\psi = \varphi|_{\mathrm{Var}(P)}$. Assume now that there is $\varphi' \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ such that $\varphi \not\sqsubseteq \varphi'$. By correctness of $\leadsto$, $\varphi'|_{\mathrm{Var}(P)} \in \mathrm{Sol}_{\overline{\mathtt{A}}}(P)$. Since $\psi = \mathrm{mgs}(P)$, $\psi = \varphi|_{\mathrm{Var}(P)} \sqsubseteq \varphi'|_{\mathrm{Var}(P)}$, that is, there is ρ such that, $\varphi|_{\mathrm{Var}(P)}\rho \leq^{\infty}_{\mathtt{A}} \varphi'|_{\mathrm{Var}(P)}$. Since $\varphi \not\sqsubseteq \varphi'$, there is x such that $x\varphi\rho \not\leq^{\infty}_{\mathtt{A}} x\varphi'$. Since $\varphi|_{\mathrm{Var}(P)}\rho \leq^{\infty}_{\mathtt{A}} \varphi'|_{\mathrm{Var}(P)}$, $x = \mathrm{x}_\beta$ for some $\beta \in \mathrm{Var}(P)$. By definition of $\mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$, there is $\mathtt{c}$ such that $\beta\varphi = \mathtt{s}^{x\varphi}\mathtt{c}$ and $\beta\varphi' = \mathtt{s}^{x\varphi'}\mathtt{c}$. Hence, $\mathtt{s}^{x\varphi}\mathtt{c} \leq^{\infty}_{\mathtt{A}} \mathtt{s}^{x\varphi'}\mathtt{c}$ and $x\varphi \not\leq^{\infty}_{\mathtt{A}} x\varphi'$. Contradiction. ■

We now prove that the most general $\mathtt{N}$-closed solutions of (C_0,C_1,C_2,C_3,C_4) can be obtained from the most general $\mathtt{N}$-closed solutions of $C_3 \cup C_4$.

Lemma 34 Let $C = (C_0,C_1,C_2,C_3,C_4)$ be a configuration.

- Correctness: if $\psi \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C_3 \cup C_4)$ and, for all $\alpha \in \mathrm{Var}_{\mathtt{A}}(C_4)$, $\mathrm{Var}(\alpha\psi) \cap C_0 = \emptyset$,[10] then $\sigma_1(C) \cup \sigma_2(C,\psi) \cup \sigma_{3,4}(C,\psi) \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C)$.
- Completeness: if $\varphi \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C)$, then $\varphi|_{\mathrm{Var}(C_3 \cup C_4)} \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C_3 \cup C_4)$.

Proof.

- Let $\psi' = \sigma_1(C) \cup \sigma_2(C,\psi) \cup \sigma_{3,4}(C,\psi)$ and $\varphi \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$. We have $\varphi_{3,4} = \varphi|_{\mathrm{Var}(C_3 \cup C_4)} \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C_3 \cup C_4)$. Since $\psi \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C_3 \cup C_4)$, $\psi \sqsubseteq \varphi_{3,4}$. By applying Lemma 32 on $(\emptyset,\emptyset,\emptyset,C_3,C_4)$, there is $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{C} \cup \{\infty\}$ such that $\mathrm{dom}(\rho) \subseteq \mathrm{Var}_{\mathtt{A}}(C_4)$ and, for all $\alpha \in \mathrm{Var}(C_3) \cup \mathrm{Var}_{\mathtt{A}}(C_4)$, $\alpha\psi\rho \leq^{\infty}_{\mathtt{A}} \alpha\varphi_{3,4}$. Then, let $\rho' = \{(\alpha,\alpha\varphi) \mid \alpha \in C_0\} \cup \{(\alpha,\alpha\rho) \mid \alpha \in \mathrm{Var}_{\mathtt{A}}(C_4)\}$. We prove that $\psi' \sqsubseteq \varphi$ by using Lemma 32. We have $\mathrm{dom}(\rho') \subseteq \mathrm{Var}_{\mathtt{A}}(C)$ by definition. If $\alpha \in C_0$, then $\alpha\psi'\rho' = \alpha\rho' = \alpha\varphi$ by definition. If $\alpha \in \mathrm{Var}(C_3)$, then $\alpha\psi'\rho' = \alpha\psi\rho' = \alpha\psi\rho$ because ψ is $\mathtt{N}$-closed, and $\alpha\psi\rho \leq^{\infty}_{\mathtt{A}} \alpha\varphi_{3,4} = \alpha\varphi$. If $\alpha \in \mathrm{Var}_{\mathtt{A}}(C_4)$, then $\alpha\psi'\rho' = \alpha\psi\rho' = \alpha\psi\rho$ since $\mathrm{Var}(\alpha\psi) \cap C_0 = \emptyset$ by assumption, and $\alpha\psi\rho \leq^{\infty}_{\mathtt{A}} \alpha\varphi_{3,4} = \alpha\varphi$.

[10] Thanks to Lemma 28, this condition can always be satisfied by applying some permutation to ψ.

- We first check that $\varphi|_{C_0}$ maps variables to variables and is injective. Let $\alpha \in C_0$ and $\varphi' = \varphi|_{\mathrm{Var}(C)-\{\alpha\}}$. Then, $\varphi' \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ too since, by definition of configuration, $\alpha \notin \mathrm{Var}(C_i)$ for every $i > 0$. Hence, $\varphi \sqsubseteq \varphi'$, that is, there is ρ such that $\alpha\varphi\rho \leq^{\infty}_{\mathtt{A}} \alpha\varphi' = \alpha$. Therefore, $\alpha\varphi$ is a variable γ. Assume now that $\gamma = \beta\varphi$ for some $\beta \in C_0$. Then, $\varphi'' = \varphi|_{\mathrm{Var}(C)-\{\alpha,\beta\}} \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$ too. Hence, $\varphi \sqsubseteq \varphi'$, that is, there is ρ' such that $\gamma\rho' \leq^{\infty}_{\mathtt{A}} \alpha\varphi' = \alpha$ and $\gamma\rho' \leq^{\infty}_{\mathtt{A}} \beta\varphi' = \beta$. Therefore, $\alpha = \gamma\rho' = \beta$.
  So, by taking in Lemma 27 $V = \mathtt{V}$, $V_1 = C_0$, $V_2 = \varphi(C_0)$, $\rho_1 = \{(\alpha, \alpha\varphi) \mid \alpha \in C_0\}$ and $\rho_2 = \{(\alpha\varphi, \alpha) \mid \alpha \in C_0\}$ (the inverse of ρ_1), there is a permutation $\xi : \mathtt{V} \to \mathtt{V}$ such that $\xi|_{C_0} = \varphi|_{C_0}$. By Lemma 28, $\varphi\xi^{-1}$ is a mgs of C too. So, wlog, we can assume that $\varphi|_{C_0}$ is the identity.
  We now prove that $\varphi_{3,4} = \varphi|_{\mathrm{Var}(C_3 \cup C_4)} \in \mathrm{mgs}^{\emptyset}_{\overline{\mathtt{B}}}(C_3 \cup C_4)$. Let $\psi \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C_3 \cup C_4)$. By Lemma 30 (1), $\psi' = \sigma_1(C) \cup \sigma_2(C, \psi) \cup \sigma_{3,4}(C, \psi) \in \mathrm{Sol}^{\emptyset}_{\overline{\mathtt{B}}}(C)$. Hence, $\varphi \sqsubseteq \psi'$. By Lemma 32, there is $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{C} \cup \{\infty\}$ such that $\mathrm{dom}(\rho) \subseteq \mathrm{Var}_{\mathtt{A}}(C_4)$ and, for all $\alpha \in C_0 \cup \mathrm{Var}(C_3) \cup \mathrm{Var}_{\mathtt{A}}(C_4)$, $\alpha\varphi\rho \leq^{\infty}_{\mathtt{A}} \alpha\psi'$. For all $\alpha \in \mathrm{Var}(C_3) \cup \mathrm{Var}_{\mathtt{A}}(C_4)$, $\alpha\varphi_{3,4}\rho = \alpha\varphi\rho \leq^{\infty}_{\mathtt{A}} \alpha\psi' = \alpha\psi$. Therefore, by Lemma 32, $\varphi_{3,4} \sqsubseteq \psi$. ■

Next, we prove that, for all affine problems P with no incompatible triples (like $C_3 \cup C_4$ in a normal configuration C), the set of finite $\mathtt{N}$-closed solutions of P is in bijection with the set of finite $\mathtt{N}$-closed solutions of:

Definition 30 (Integer problem associated to an affine problem) Given an affine problem P, let $I(P)$ be the integer problem obtained by replacing in P every constraint $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta$ by $\mathrm{x}_\alpha + k \leq^? \mathrm{x}_\beta + l$, and every constraint $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta$ by $e \leq^? \mathrm{x}_\beta + l$.

Lemma 35 If P is an affine problem with no incompatible triples, then:

1. there is a strictly monotone map $\psi \mapsto \check{\psi}$ from $(\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P)), \sqsubseteq)$ to $(\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P), \sqsubseteq)$;
2. there is a monotone map $\varphi \mapsto \hat{\varphi}$ from $(\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P), \sqsubseteq)$ to $(\mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P)), \sqsubseteq)$;
3. for all $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P))$, $\check{\psi} = \psi$;
4. for all $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$, there is $\rho : \mathtt{V} \to \mathtt{V} \cup \mathtt{C}$ such that $\varphi = \widehat{\varphi}\rho$, hence $\widehat{\varphi} \sqsubseteq \varphi$;
5. correctness: if $\psi \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(I(P))$, then $\check{\psi} \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(P)$;
6. completeness: if $\varphi \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(P)$, then $\hat{\varphi} \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(I(P))$.

Proof. Let $\simeq_P$ be the symmetric and transitive closure of $\leq_P$ and $\eta : \mathtt{H}/\simeq_P \to \mathtt{H}$ be any function such that, for all equivalence classes X, $\eta(X) \in X$ ($\mathtt{H}$ and $\leq_P$ are introduced in Definition 27). Such a function always exists because equivalence classes are non-empty. Because P has no incompatible triples, an equivalence class modulo $\simeq_P$ cannot contain two different constants. Hence, we can assume that $\eta(X) = \mathtt{c}$ iff $\mathtt{c} \in X$.

Given $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P))$, let $\check{\psi} = \{(x, x\psi) \mid x \in \mathrm{Var}_{\mathtt{N}}(P)\} \cup \{(\alpha, \mathtt{s}^{\mathrm{x}_\alpha\psi}\alpha^*) \mid \alpha \in \mathrm{Var}_{\mathtt{A}}(P)\}$ where $\alpha^* = \eta([\alpha]_P)$ and $[\alpha]_P$ is the equivalence class of α modulo $\simeq_P$.

Given $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$, let $\hat{\varphi} = \{(x, x\varphi) \mid x \in \mathrm{Var}_{\mathtt{N}}(P)\} \cup \{(\mathrm{x}_\alpha, \alpha\varphi_s) \mid \alpha \in \mathrm{Var}_{\mathtt{A}}(P)\}$ (φ_s is introduced in Definition 24).

1. We first check that $\check{\psi} \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ whenever $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$, that is, $\check{\psi}$ satisfies every constraint of P. This is immediate for constraints of sort $\mathtt{N}$. Otherwise, since P is affine, there are two cases. If $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in P$, then $\alpha^* = \beta^*$ and $\mathrm{x}_\alpha\psi + k \leq_{\mathtt{A}} \mathrm{x}_\beta\psi + l$.

Hence, $(\mathtt{s}^k\alpha)\dot{\psi} = \mathtt{s}^{k+\alpha\psi}\alpha^* \leq_{\mathtt{A}} \mathtt{s}^{l+\beta\psi}\beta^* = (\mathtt{s}^l\alpha)\dot{\psi}$. If $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in P$, then $\beta^* = \mathtt{c}$ and $e\psi \leq_{\mathtt{A}} \mathtt{x}_\beta\psi + l$. Therefore, $(\mathtt{s}^e\mathtt{c})\dot{\psi} = \mathtt{s}^{e\psi}\mathtt{c} \leq_{\mathtt{A}} \mathtt{s}^{\beta\psi+l}\beta^* = (\mathtt{s}^l\beta)\dot{\psi}$.
Next, one can easily check that $\psi \mapsto \dot{\psi}$ is injective ($\psi_1 = \psi_2$ whenever $\dot{\psi_1} = \dot{\psi_2}$) and monotone wrt. $\leq_{\mathtt{A}}$ ($\dot{\varphi} \leq_{\mathtt{A}} \dot{\psi}$ whenever $\varphi \leq_{\mathtt{A}} \psi$) and thus wrt. $\sqsubseteq$. Therefore, $\psi \mapsto \dot{\psi}$ is strictly monotone wrt. $\sqsubseteq$.

2. We first check that $\dot{\varphi} \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P))$ whenever $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$. If $\mathtt{s}^k\alpha \leq^? \mathtt{s}^l\beta \in P$, then $(\mathtt{s}^k\alpha)\varphi = \mathtt{s}^{\alpha\varphi_s+k}\alpha\varphi_h \leq_{\mathtt{A}} (\mathtt{s}^l\beta)\varphi = \mathtt{s}^{\beta\varphi_s+l}\beta\varphi_h$. So, $\alpha\varphi_h = \beta\varphi_h$ and $\mathtt{x}_\alpha\dot{\varphi} + k \leq_{\mathtt{A}} \mathtt{x}_\beta\dot{\varphi} + l$. Assume now that $\mathtt{s}^e\mathtt{c} \leq^? \mathtt{s}^l\beta \in P$. Then, $(\mathtt{s}^e\mathtt{x})\varphi = \mathtt{s}^{e\varphi_s}\mathtt{c} \leq_{\mathtt{A}} (\mathtt{s}^l\beta)\varphi = \mathtt{s}^{\beta\varphi_s+l}\beta\varphi_h$. So, $\mathtt{c} = \beta\varphi_h$ and $e\dot{\varphi} \leq_{\mathtt{A}} \mathtt{x}_\beta\dot{\varphi} + l$.
We now check that $\varphi \mapsto \dot{\varphi}$ is monotone. Let $\varphi_1, \phi_2 \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ such that $\varphi_1 \sqsubseteq \varphi_2$. Hence, there is $\rho : \mathtt{V} \to \mathtt{C} \cup \mathtt{V}$ such that $\varphi\rho \leq_{\mathtt{A}} \psi$. Therefore, $\dot{\varphi} \leq_{\mathtt{A}} \dot{\psi}$.
3. Immediate.
4. Let ρ the map from $\mathtt{V}$ to $\mathtt{V} \cup \mathtt{C}$ such that, if $\alpha^* \in \mathtt{V}$, then $\alpha^*\rho = \alpha\varphi_h$. The map ρ is well defined since φ_h is invariant by $\simeq_P$: if $\alpha \simeq_P \beta$, then $\alpha\varphi_h = \beta\varphi_h$. Now, one can easily check that $\varphi = \widehat{\varphi}\rho$. If $\alpha^* = \mathtt{c}$, then there is a constraint $\mathtt{s}^k\mathtt{c} \leq^? \mathtt{s}^l\alpha \in P$. Since $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$, $\alpha\varphi_h = \mathtt{c}$ and $\alpha\widehat{\varphi}\rho = \mathtt{s}^{\alpha\varphi_s}\alpha^*\rho = \alpha\varphi$. Otherwise, $\alpha\widehat{\varphi}\rho = \mathtt{s}^{\alpha\varphi_s}\alpha^*\rho = \mathtt{s}^{\alpha\varphi_s}\alpha\varphi_h = \alpha\varphi$.
5. Let $\varphi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$. By 2, $\dot{\varphi} \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P))$ and $\psi \sqsubseteq \dot{\varphi}$. By 1, $\dot{\psi} \sqsubseteq \widehat{\varphi}$. By 4, $\widehat{\varphi} \sqsubseteq \varphi$. Therefore, $\dot{\psi} \sqsubseteq \varphi$.
6. Let $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(I(P))$. By 1, $\dot{\psi} \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ and $\varphi \sqsubseteq \dot{\psi}$. By 2, $\dot{\varphi} \sqsubseteq \ddot{\psi}$. By 3, $\ddot{\psi} = \psi$. Therefore, $\dot{\varphi} \sqsubseteq \psi$. ■

Lemma 36 Every satisfiable integer problem has a smallest $\mathtt{N}$-closed solution that can be computed in polynomial time.

Proof. Let P be a satisfiable integer problem whose variables are $x_1, \ldots, x_n$. 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 (Baccelli *et al.*, 1992).

Wlog. we can assume that P contains no constraints of the form $0 + k \leq 0$ (since P is satisfiable, these constraints are always satisfied and thus can be removed). Hence, P contains only constraints of the form $x_i + k \leq^? x_j$, $0 + k \leq^? x_j$ or $x_i + k \leq^? 0$, that is, in the syntax of $(\overline{\mathbb{Z}}_{\max}, \oplus, \otimes)$, $k \otimes x_i \leq x_j$, $k \leq x_j$ or $x_i \leq -k$.

Given a problem P, let $a_{ij} = \sup\{k \in \overline{\mathbb{Z}}_{\max} \mid x_j + k \leq^? x_i \in P\}$, $b_{ij} = \sup(\{0\} \cup \{k \in \overline{\mathbb{Z}}_{\max} \mid 0 + k \leq^? x_i \in P\})$ (we add 0 because solutions must be non-negative) and $c_{ij} = \inf\{-k \in \overline{\mathbb{Z}}_{\max} | x_i + k \leq^? 0 \in P\}$ with, as usual, $\sup\emptyset = -\infty$ and $\inf\emptyset = +\infty$. Note that, in b and c, every column is the same (b_{ij} and c_{ij} do not depend on j).

We now prove that, if $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$, then there is $x \in \overline{\mathbb{Z}}_{\max}^{n\times n}$ such that $(a \otimes x) \oplus b \leq x \leq c$ and, for all j, $x_{ij} = x_i\psi$ (the columns of x are equal). For all i and j, the set of inequations $\{k \otimes x_i \leq x_j \mid x_i + k \leq^? x_j \in P\}$ is equivalent to $a_{ji} \otimes x_i \leq x_j$ since $k \leq a_{ji}$ and $(-\infty) \otimes x_i = -\infty \leq x_j$. Hence, $\{a_{ji} \otimes x_i \leq x_j \mid i \in \{1, \ldots, n\}\}$ is equivalent to $\bigoplus_{i=1}^n a_{ji} \otimes x_i \leq x_j$. By taking $x_{il} = x_i$ for all l, we therefore get $(a \otimes x)_{jl} \leq x_{jl}$. Similarly, for all j and l, $\{k \leq x_j \mid 0 + k \leq^? x_j \in P\} \cup \{0 \leq x_j\}$ (implicit in P) is equivalent to $b_{jl} \leq x_{jl}$. Therefore, $(a \otimes x) \oplus b \leq x$. Finally, for all i, $\{x_i \leq -k \mid x_i + k \leq^? 0 \in P\}$ is equivalent to $x_{il} \leq c_{il}$ for all l, that is, $x \leq c$.

Fig. 10. Algorithm computing a most general solution in the successor algebra.

1. Apply the algorithm of Figure 9.
2. Compute the most general $\mathtt{N}$-closed solution ψ of $C_3 \cup I(C_4)$ using Lemma 36.
3. Compute $\acute{\psi}$ defined in Lemma 35.
4. Return $\sigma_1(C) \cup \sigma_2(C, \acute{\psi}) \cup \sigma_{4_\mathtt{A}}(C, \acute{\psi})$.

Because we proceeded by equivalence, we also have the converse: if $(a \otimes x) \oplus b \leq x \leq c$, then $\psi_l \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ where ψ_l is the substitution such that $x_i \psi_l = x_{il}$ (l-th column of x).

By Theorem 4.75 in (Baccelli *et al.*, 1992), $(a \otimes x) \oplus b \leq x$ has $a^* \otimes b$ as smallest solution, where $a^* = \bigoplus_{k \in \mathbb{N}} a^k$, $a^{k+1} = a^k \otimes a$, and a^0 is the matrix with 0 on the diagonal and $-\infty$ everywhere else. Since P is satisfiable, $G(P)$ has no positive cycles. Hence, $a^* = \bigoplus_{k=0}^{n} a^k$ (Cuninghame-Green, 1979) (Theorem 3.20 in (Baccelli *et al.*, 1992)). Therefore, $\psi \in \mathrm{Sol}^{\emptyset}_{\mathtt{B}}(P)$ iff $a^* \otimes b \leq c$, and the smallest solution of P is the function ψ such that $x_i \psi = \bigoplus_{k=1}^{n} a^*_{ik} \otimes b_{k1}$, which can be computed in polynomial time. ■

Therefore, we can now conclude:

Theorem 5 In the successor algebra, any satisfiable size problem has a most general solution that can be computed in polynomial time following the algorithm of Figure 10.

Proof. Correctness. By Lemma 36, $\psi \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(I(C_3 \cup C_4))$. By Lemma 35 (5), $\acute{\psi} \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(C_3 \cup C_4) \subseteq \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(C_3 \cup C_4)$. By Lemma 34 (1), $\varphi = \sigma_1(C) \cup \sigma_2(C, \acute{\psi}) \cup \sigma_{3,4}(C, \acute{\psi}) \in \mathrm{mgs}^{\emptyset}_{\mathtt{B}}(C)$. By Lemma 33 (1), $\varphi|_{\mathrm{Var}(P)} = \sigma_1(C) \cup \sigma_2(C, \acute{\psi}) \cup \sigma_{4_\mathtt{A}}(C, \acute{\psi}) \in \mathrm{mgs}_{\overline{\mathtt{A}}}(P)$.

Complexity. After Theorem 4, $C_3 \cup C_4$ is of polynomial size wrt. the size of P. The computation of $I(C_3 \cup C_4)$ is linear. After Lemma 36, the computation of ψ is polynomial. After Lemma 35 (1), the computation of $\acute{\psi}$ is polynomial. Therefore, the algorithm of Figure 10 is polynomial. ■

Example 13 In Example 12, we have seen that the normal form of $(\emptyset, \emptyset, \emptyset, \emptyset, P)$ where $P = \{\alpha \leq^? \mathtt{sc}, \beta \leq^? \alpha\}$ is $(\emptyset, \emptyset, \{(\alpha, \mathtt{c}), (\beta, \mathtt{c})\}, C_3, \emptyset)$ with $C_3 = \{\mathrm{x}_\alpha \leq^? 1, \mathrm{x}_\beta \leq^? \mathrm{x}_\alpha\}$. Following Lemma 36, by taking $x_1 = \mathrm{x}_\alpha$ and $x_2 = \mathrm{x}_\beta$, the corresponding max-linear system is $(a \otimes x) \oplus b \leq x \leq c$ where $a_{11} = \sup\{k \mid \mathrm{x}_\alpha + k \leq^? \mathrm{x}_\alpha \in C_3\} = \sup \emptyset = -\infty$, $a_{12} = \sup\{\mathrm{x}_\beta + k \leq^? \mathrm{x}_\alpha \in C_3\} = \sup\{0\} = 0$, $a_{21} = \sup\{k \mid \mathrm{x}_\alpha + k \leq^? \mathrm{x}_\beta \in C_3\} = \sup \emptyset = -\infty$, $a_{22} = \sup\{k \mid \mathrm{x}_\beta + k \leq^? \mathrm{x}_\beta \in C_3\} = \sup \emptyset = -\infty$, $b_1 = \sup(\{0\} \cup \{k \mid k \leq^? \mathrm{x}_\alpha \in C_3\}) = \sup\{0\} = 0$, $b_2 = \sup(\{0\} \cup \{k \mid k \leq^? \mathrm{x}_\alpha \in C_3\}) = \sup\{0\} = 0$, $c_1 = \inf\{k \mid \mathrm{x}_\alpha \leq^? k \in C_3\} = \inf\{1\} = 1$ and $c_2 = \inf\{k \mid \mathrm{x}_\beta \leq^? k \in C_3\} = \inf \emptyset = +\infty$. To summarize, we have: $a = \begin{pmatrix} -\infty & 0 \\ -\infty & -\infty \end{pmatrix}$, $b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and $c = \begin{pmatrix} 1 \\ +\infty \end{pmatrix}$. One can easily check that, if $x = \begin{pmatrix} \mathrm{x}_\alpha \\ \mathrm{x}_\beta \end{pmatrix}$, then $(a \otimes x) \oplus b = \begin{pmatrix} \mathrm{x}_\beta \oplus 0 \\ 0 \end{pmatrix}$, hence that $(a \otimes x) \oplus b \leq x \leq c$ is equivalent to $\mathrm{x}_\beta \oplus 0 \leq \mathrm{x}_\alpha \leq 1$ and $0 \leq \mathrm{x}_\beta \leq +\infty$, which is C_3. Now, $a^0 = \begin{pmatrix} 0 & -\infty \\ -\infty & 0 \end{pmatrix}$ and $a^2 =$

$\begin{pmatrix} -\infty & -\infty \\ -\infty & -\infty \end{pmatrix}$. Hence, $a^* = a^0 \oplus a = \begin{pmatrix} 0 & 0 \\ -\infty & 0 \end{pmatrix}$ and $a^* \otimes b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. So, the smallest solution of C_3 is $\psi = \{(\mathrm{x}_\alpha, 0), (\mathrm{x}_\beta, 0)\}$ and the smallest solution of P is $\sigma_1(C) \cup \sigma_2(C, \psi) \cup \sigma_{4_\mathrm{A}}(C, \psi) = \{(\alpha, \mathsf{c}), (\beta, \mathsf{c})\}$. ■

10 Conclusion

We have presented a general and modular 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 (Hughes *et al.*, 1996).

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 (HOT, 2012), which tries to detect size-preserving functions and, following (Abel & Altenkirch, 2002), to find a lexicographic ordering on arguments. Combined with other (non-)termination techniques (Jouannaud & Okada, 1991; Blanqui, 2000; Blanqui *et al.*, 2002), HOT won the 2012 international competition of termination provers (Termination competition, 2017) for higher-order rewriting against THOR (THOR, 2014) and WANDA (Wanda, 2015). It could be improved by replacing the lexicographic ordering by the size-change principle (Lee *et al.*, 2001; Hyvernat, 2014), and using abstract interpretation techniques for annotating function symbols (Telford & Turner, 2000; Chin & Khoo, 2001). 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 (Fuhs *et al.*, 2007; Ben-Amram & Codish, 2008; Codish *et al.*, 2011).

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 (Blanqui & Riba, 2006), it is also possible to consider full Presburger arithmetic (Presburger, 1929) and handle conditional rewrite rules, by extending the system with explicit quantifiers and constraints on size variables, in the spirit of HM(X) (Sulzmann, 2001). Simplification of constraints is then an important issue in practice (Pottier, 2001).

We have presented this criterion in Church' simply typed λ-terms but, following (Blanqui, 2005b), 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

(Blanqui, 2016), 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 (Klop *et al.*, 1993) or Nipkow's higher-order rewrite systems (Mayr & Nipkow, 1998).

Another interesting extension would be to consider size-annotated types in the computability path ordering (Blanqui *et al.*, 2015), following Kamin and Lévy's extension of Dershowitz' recursive path ordering (Dershowitz, 1979b; Kamin & Lévy, 1980), and Borralleras and Rubio's extension of Jouannaud and Okada's higher-order recursive path ordering (Jouannaud & Rubio, 1999; Borralleras & Rubio, 2001).

Acknowledgments. I would like to thank Christophe Raffalli for a short but useful discussion on max-plus algebra, and Nachum Dershowitz, Jean-Pierre Jouannaud and Sylvain Schmitz for their comments on the introduction and the conclusion. I also want to thank very much the anonymous referees for their very careful reading and the numerous remarks and suggestions they made. This greatly helped me to improve the article.

References

Abel, A. (2004). Termination checking with types. *Theoretical informatics and applications*, **38**(4), 277–319.

Abel, A. (2006). *A polymorphic lambda-calculus with sized higher-order types*. Ph.D. thesis, Ludwig-Maximilians-Universität München, Germany.

Abel, A. (2008). Semi-continuous sized types and termination. *Logical methods in computer science*, **4**(2), 1–33.

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

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

Abel, A., & Altenkirch, T. (2002). A predicative analysis of structural recursion. *Journal of functional programming*, **12**(1), 1–41.

Ackermann, W. (1925). Begründung des "tertium non datur" mittels der Hilbertschen Theorie der Widerspruchsfreiheit. *Mathematische annalen*, **93**, 1–36.

Agda. (2017). http://wiki.portal.chalmers.se/agda/pmwiki.php.

Amadio, R., & Coupet-Grimal, S. (1997). *Analysis of a guard condition in type theory (preliminary report)*. Tech. rept. 3300. INRIA, France.

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

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

Arts, T., & Giesl, J. (2000). Termination of term rewriting using dependency pairs. *Theoretical computer science*, **236**, 133–178.

ATS. (2018). http://www.ats-lang.org/.

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

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

Barbanera, F., Fernández, M., & Geuvers, H. (1997). Modularity of strong normalization in the algebraic-λ-cube. *Journal of functional programming*, **7**(6), 613–660.

Barthe, G., Frade, M. J., Giménez, E., Pinto, L., & Uustalu, T. (2004). Type-based termination of recursive definitions. *Mathematical structures in computer science*, **14**(1), 97–141.

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

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

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

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

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

Berger, U. (2008). A domain model characterising strong normalisation. *Annals of pure and applied logic*, **156**(1), 39–50.

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

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

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

Blanqui, F. (2005b). Definitions by rewriting in the calculus of constructions. *Mathematical structures in computer science*, **15**(1), 37–92.

Blanqui, F. (2006a). Higher-order dependency pairs. *8th International Workshop on Termination*.

Blanqui, F. (2006b). *(HO)RPO revisited*. Tech. rept. 5972. INRIA, France.

Blanqui, F. (2016). Termination of rewrite relations on λ-terms based on Girard's notion of reducibility. *Theoretical computer science*, **611**, 50–86.

Blanqui, F., & Riba, C. (2006). Combining typing and size constraints for checking the termination of higher-order conditional re *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, Lecture Notes in Computer Science 4246.

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

Blanqui, F., Jouannaud, J.-P., & Okada, M. (2002). Inductive-data-type systems. *Theoretical computer science*, **272**, 41–68.

Blanqui, F., Jouannaud, J.-P., & Rubio, A. (2015). The computability path ordering. *Logical methods in computer science*, **11**(4), 1–45.

Bonfante, G., Marion, J.-Y., & Péchoux, R. (2011). Quasi-interpretations a way to control resources. *Theoretical computer science*, 2776–2796.

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

Boyer, R., & Moore, J. (1979). *A computational logic*. Academic Press.

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

Burstall, R., Queen, D. Mac, & Sannella, D. (1980). HOPE: an experimental applicative language. *Proceedings of the ACM Symposium on Lisp and Fonctional Programming*.

Cheney, J. (2003). *First-class phantom types*. Tech. rept. TR2003-1901. Cornell University.

Cherifa, A. Ben, & Lescanne, P. (1987). Termination of rewriting systems by polynomial interpretations and its implementation. *Science of computer programming*, **9**(2), 137–159.

Chin, W. N., & Khoo, S. C. (2001). Calculating sized types. *Journal of higher-order and symbolic computation*, **14**(2-3), 261–300.

Church, A. (1940). A formulation of the simple theory of types. *Journal of symbolic logic*, **5**, 56–68.

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

cicminus. (2015). https://github.com/jsacchini/coq.

Codish, M., Giesl, J., Schneider-Kamp, P., & Thiemann, R. (2011). SAT solving for termination proofs with recursive path orders and dependency pairs. *Journal of automated reasoning*, **49**(1), 53–93.

Collins, G. (1975). Quantifier elimination for real closed fields by cylindrical algebraic decompostion. *Proceedings of the 2nd GI conference on Automata Theory and Formal Languages*. Lecture Notes in Computer Science, no. 33.

Contejean, E., Marché, C., Tomás, A. P., & Urbain, X. (2005). Mechanically proving termination using polynomial interpretations. *Journal of automated reasoning*, **34**(4), 325–363.

Coq. (2017). http://coq.inria.fr/.

Coquand, T., & Paulin-Mohring, C. (1988). Inductively defined types. *Proceedings of the International Conference on Computer Logic*, Lecture Notes in Computer Science 417.

Coquand, T., & Spiwack, A. (2007). A proof of strong normalization using domain theory. *Logical methods in computer science*, **3**(4), 1–16.

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

Cousot, P. (1996). Abstract interpretation. *Acm computing surveys*, **28**(2), 324–328.

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

Cousot, P., & Cousot, R. (1979). Constructive versions of Tarski's fixed point theorems. *Pacific journal of mathematics*, **82**(1), 43–57.

Cuninghame-Green, R. (1979). *Minimax algebra*. Lecture Notes in Economics and Mathematical Systems, no. 166. SV.

Curien, P.-L., & Ghelli, G. (1992). Coherence of subsumption, minimum typing and type-checking in $F_{\leq}$. *Logical methods in computer science*, **2**(1), 55–91.

Curry, H. B., & Feys, R. (1958). *Combinatory logic*. North-Holland.

de Bruijn, N. G. (1970). The mathematical language AUTOMATH, its usage, and some of its extensions. *Proceedings of the 1968 Symposium on Automatic Demonstration*. Lecture Notes in Mathematics, vol. 125.

de Vrijer, R. (1987). Exactly estimating functionals and strong normalization. *Indagationes mathematicae*, **90**(4), 479–493.

Dedukti. (2018). https://deducteam.github.io/.

Dershowitz, N. (1979a). A note on simplification orderings. *Information processing letters*, **9**(5), 212–215.

Dershowitz, N. (1979b). Orderings for term rewriting systems. *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*.

Dershowitz, N. (1982). Orderings for term rewriting systems. *Theoretical computer science*, **17**, 279–301.

Dershowitz, N. (2013). Dependency Pairs are a Simple Semantic Path Ordering. *13th International Workshop on Termination*.

Dershowitz, N., & Jouannaud, J.-P. (1990). Rewrite systems. *Chap. 6, pages 243–320 of:* van Leeuwen, J. (ed), *Handbook of Theoretical Computer Science. Volume B: formal models and methods*. North-Holland.

Dershowitz, N., & Manna, Z. (1979). Proving termination with multiset orderings. *Communications of the acm*, **22**(8), 465–476.

Dershowitz, N., & Okada, M. (1988). Proof-theoretic techniques for term rewriting. *Proceedings of the 3rd IEEE Symposium on Logic in Computer Science*.

Endrullis, J., Waldmann, J., & Zantema, H. (2008). Matrix interpretations for proving termination of term rewriting. *Journal of automated reasoning*, **40**(2-3), 195–220.

Fiore, M., Plotkin, G., & Turi, D. (1999). Abstract syntax and variable binding. *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*.

Fischer, M., & Rabin, M. (1974). Super-exponential complexity of Presburger arithmetic. *Proceedings of the SIAM-AMS Symposium in Applied Mathematics*.

Fuh, Y.-C., & Mishra, P. (1990). Type inference with subtypes. *Theoretical computer science*, **73**(2), 155–175.

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

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

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

Gandy, R. O. (1980a). An early proof of normalization by A. M. Turing. *Pages 453–455 of:* Hindley, R., & Seldin, J. P. (eds), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press.

Gandy, R. O. (1980b). Proofs of strong normalization. *Pages 457–477 of:* Hindley, R., & Seldin, J. P. (eds), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press.

Gentzen, G. (1935). Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische annalen*, **112**(1), 493–565. English translation in (Szabo, 1969).

Giesl, J. (1997). Termination of nested and mutually recursive algorithms. *Journal of automated reasoning*, **19**(1), 1–29.

Giesl, J., Arts, T., & Ohlebusch, E. (2002). Modular termination proofs for rewriting using dependency pairs. *Journal of symbolic computation*, **34**(1), 21–58.

Giesl, J., Thiemann, R., Schneider-Kamp, P., & Falke, S. (2006). Mechanizing and improving dependency pairs. *Journal of automated reasoning*, **37**(3), 155–203.

Giménez, E. (1996). *Un calcul de constructions infinies et son application à la vérification de systèmes communiquants*. Ph.D. thesis, ENS Lyon, France.

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

Girard, J.-Y. (1972). *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur*. Ph.D. thesis, Université Paris 7, France.

Girard, J.-Y., Lafont, Y., & Taylor, P. (1988). *Proofs and types*. Cambridge University Press.

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für mathematik und physik*, **38**, 173–198. English translation in (v. Heijenoort, 1977).

Gödel, K. (1958). Über einer bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, **12**(3-4), 280–287. Reprinted in (Gödel, 1990).

Gödel, K. (1990). *Collected works - vol. 2: publications 1938-1974*. Oxford University Press.

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

Hamana, M. (2006). An initial algebra approach to term rewriting systems with variable binders. *Journal of higher-order and symbolic computation*, **19**(2-3), 231–262.

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

Hardy, G. H. (1904). A theorem concerning the infinite cardinal numbers. *Quarterly journal of mathematics*, **35**, 87–94.

Hartogs, F. (1915). Über das Problem der Wohlordnung. *Mathematische annalen*, **76**, 438–443.

Haskell. (2017). https://www.haskell.org/.

Herbrand, J. (1930). *Recherches sur la théorie de la démonstration*. Ph.D. thesis, Faculté des sciences de Paris, France.

Hessenberg, G. (1909). Kettentheorie und Wohlordnung. *Journal für die reine und angewandte Mathematik*, **135**, 81–133.

Hindley, R. (1969). The principal type-scheme of an object in combinatory logic. *Transactions of the american mathematical society*, **146**, 29–60.

Hirokawa, N., & Middeldorp, A. (2005). Automating the dependency pair method. *Information and computation*, **199**(1-2), 172–199.

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

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

Hong, H., & Jakuš, D. (1998). Testing positiveness of polynomials. *Journal of automated reasoning*, **21**(1), 23–38.

HOT. (2012). http://rewriting.gforge.inria.fr/hot.html.

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

Howard, W. A. (1972). A system of abstract constructive ordinals. *Journal of symbolic logic*, **37**(2), 355–374.

Hrbacek, K., & Jech, T. (1999). *Introduction to set theory*. 3rd, revised and expanded edn. M. Dekker.

Huet, G. (1976). *Résolution d'équations dans les langages d'ordre 1, 2, . . . , ω*. Thèse d'État, Université Paris 7, France.

Huet, G., & Hullot, J.-M. (1982). Proofs by induction in equational theories with constructors. *Journal of computer and system sciences*, **25**(2), 239–266.

Hughes, J., Pareto, L., & Sabry, A. (1996). Proving the correctness of reactive systems using sized types. *Proceedings of the 23th ACM Symposium on Principles of Programming Languages*.

Hyvernat, P. (2014). The Size-Change Termination Principle for Constructor Based Languages. *Logical methods in computer science*, **10**(1), 1–30.

Jones, N. D., & Bohr, N. (2008). Call-by-value termination in the untyped λ-calculus. *Logical methods in computer science*, **4**(1), 1–39.

Jouannaud, J.-P., & Okada, M. (1991). A computation model for executable higher-order algebraic specification languages. *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*.

Jouannaud, J.-P., & Rubio, A. (1999). The higher-order recursive path ordering. *Proceedings of the 14th IEEE Symposium on Logic in Computer Science*.

Jouannaud, J.-P., & Rubio, A. (2007). Polymorphic higher-order recursive path orderings. *Journal of the acm*, **54**(1), 1–48.

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

Kamin, S., & Lévy, J.-J. (1980). *Attempts for generalizing the recursive path orderings*. Unpublished note.

Klop, J. W., van Oostrom, V., & van Raamsdonk, F. (1993). Combinatory reduction systems: introduction and survey. *Theoretical computer science*, **121**, 279–308.

Knaster, B., & Tarski, A. (1928). Un théorème sur les fonctions d'ensembles. *Annales de la société polonaise de mathématiques*, **6**, 133–134.

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

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

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

Kusakari, K., & Sakai, M. (2007). Enhancing dependency pair method using strong computability in simply-typed term rewriting *Applicable algebra in engineering communication and computing*, **18**(5), 407–431.

Kusakari, K., Isogai, Y., Sakai, M., & Blanqui, F. (2009). Static dependency pair method based on strong computability for higher-order rewrite systems. *Ieice transactions on information and systems*, **E92-D**(10), 2007–2015.

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

Lucas, S. (2005). Polynomials over the reals in proofs of termination: from theory to practice. *Theoretical informatics and applications*, **39**, 547–586.

Manna, Z., & Ness, S. (1970). On the termination of Markov algorithms. *Proceedings of the 3rd Hawaii International Conference on System Sciences*.

Martin-Löf, P. (1975). An intuitionistic theory of types: predicative part. Rose, H. E., & Shepherdson, J. C. (eds), *Proceedings of the 1973 Logic Colloquium*. Studies in Logic and the Foundations of Mathematics, vol. 80. North-Holland.

Matiyasevich, Y. V. (1970). Enumerable sets are diophantine. *Soviet mathematics. doklady*, **11**, 354–358.

Matiyasevich, Y. V. (1993). *Hilbert's tenth problem*. MIT Press.

Maude. (2015). http://maude.cs.uiuc.edu/.

Mayr, R., & Nipkow, T. (1998). Higher-order rewrite systems and their confluence. *Theoretical computer science*, **192**(2), 3–29.

Mendler, N. P. (1987). *Inductive definition in type theory*. Ph.D. thesis, Cornell University, USA.

Mendler, N. P. (1991). Inductive types and type constraints in the second-order lambda calculus. *Annals of pure and applied logic*, **51**(1-2), 159–172.

Middeldorp, A., Ohsaki, H., & Zantema, H. (1996). Transforming termination by self-labelling. *Proceedings of the 13th International Conference on Automated Deduction*, Lecture Notes in Computer Science 1104.

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

Milner, R. (1978). A theory of type polymorphism in programming. *Journal of computer and system sciences*, **17**(3), 348–375.

MiniAgda. (2014). http://www.cse.chalmers.se/ abela/miniagda/.

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

Monin, F., & Simonot, M. (2001). An ordinal measure based procedure for termination of functions. *Theoretical computer science*, **254**(1-2), 63–94.

Moser, G. (2014). KBOs, ordinals, subrecursive hierarchies and all that. *Journal of logic and computation*, 1–27. Published online on December 3.

Newman, M. (1942). On theories with a combinatorial definition of "equivalence". *Annals of mathematics*, **43**(2), 223–243.

Nipkow, T. (1991). Higher-order critical pairs. *Proceedings of the 6th IEEE Symposium on Logic in Computer Science*.

OCaml. (2017). http://ocaml.org/.

Okada, M. (1989). Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent term rev *Proceedings of the International Symposium on Symbolic and Algebraic Computation*.

Pareto, L. (2000). *Types for crash prevention*. Ph.D. thesis, Chalmers University of Technology, Göteborg, Sweden.

Paulson, L. (1986). Proving termination of normalization functions for conditional expressions. *Journal of automated reasoning*, **2**(1), 63–74.

Peano, G. (1889). *Arithmetices principia, nova methodo exposita*. Fratres Bocca. Partial English translation in (v. Heijenoort, 1977).

Plotkin, G. D. (1977). LCF considered as a programming language. *Theoretical computer science*, **5**(3), 223–255.

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

Pratt, V. (1977). *Two easy theories whose combination is hard*. Unpublished note.

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

Rathjen, M. (2006). The art of ordinal analysis. *Pages 45–69 of: Proceedings of the International Congress of Mathematicians*, vol. 2.

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

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

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

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

Rubin, H., & Rubin, J. E. (1963). *Equivalents of the axiom of choice*. North-Holland.

Sacchini, J. L. (2011). *On Type-Based Termination and Dependent Pattern Matching in the Calculus of Inductive Constructions*. Ph.D. thesis, ParisTech, France.

Sakai, M., Watanabe, Y., & Sakabe, T. (2001). 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.

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

Scott, D. S. (1972). Continuous lattices. *Pages 97–136 of:* Lawvere, E. (ed), *Toposes, algebraic geometry and logic*. Lecture Notes in Mathematics, no. 274. Springer.

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

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

Sternagel, C., & Middeldorp, A. (2008). Root labeling. *Proceedings of the 19th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 5117. This paper contains errors described in (Sternagel & Thiemann, 2010).

Sternagel, C., & Thiemann, R. (2010). Signature extensions preserve termination - an alternative proof via dependency pairs. *Proceedings of the 24th International Conference on Computer Science Logic*, Lecture Notes in Computer Science 6247.

Sulzmann, M. (2000). *A general framework for Hindley/Milner type systems with constraints*. Ph.D. thesis, Yale University, USA.

Sulzmann, M. (2001). A general type inference framework for Hindley/Milner style systems. *Proceedings of the 5th Fuji International Symposium on Functional and Logic Programming*, Lecture Notes in Computer Science 2024.

Szabo, M. E. (ed). (1969). *Collected papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North-Holland.

Tait, W. W. (1967). Intensional interpretations of functionals of finite type I. *Journal of symbolic logic*, **32**(2), 198–212.

Tarski, A. (1948). *A decision method for elementary algebra and geometry*. Tech. rept. R-109. RAND Corporation, USA.

Tarski, A. (1955). A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of mathematics*, **5**, 285–309.

Telford, A., & Turner, D. (2000). Ensuring termination in ESFP. *Proceedings of the 15th British Colloquium for Theoretical Computer Science*, Journal of Universal Computer Science 6(4).

TeReSe. (2003). *Term rewriting systems*. Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press.

Termination competition. (2017). http://termination-portal.org/wiki/Termination_Competition.

Thiemann, R., & Giesl, J. (2005). The size-change principle and dependency pairs for termination of term rewriting. *Applicable algebra in engineering communication and computing*, **16**(4), 229–270.

THOR. (2014). http://www.lsi.upc.es/ albert/.

Turing, A. M. (1942). *Some theorems about Church's system.* Unpublished typescript reproduced in (Gandy, 1980a).

v. Heijenoort, J. (ed). (1977). *From Frege to Gödel, a source book in mathematical logic, 1879-1931*. Harvard University Press.

van de Pol, J. (1993). Termination proofs for higher-order rewrite systems. *Proceedings of the 1st International Workshop on Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science 816.

van de Pol, J. (1995). Two different strong normalization proofs? Computability versus functionals of finite type. *Proceedings of the 2nd International Workshop on Higher-Order Algebra, Logic and Term Rewriting*, Lecture Notes in Computer Science 1074.

van de Pol, J. (1996). *Termination of higher-order rewrite systems*. Ph.D. thesis, Utrecht Universiteit, NL.

van Oostrom, V. (1994). *Confluence for abstract and higher-order rewriting*. Ph.D. thesis, Vrije Universiteit Amsterdam, NL.

Wahlstedt, D. (2007). *Dependent type theory with first-order parameterized data types and well-founded recursion*. Ph.D. thesis, Chalmers University of Technology, Sweden.

Walther, C. (1988). Argument-bounded algorithms as a basis for automated termination proofs. *Proceedings of the 9th International Conference on Automated Deduction*, Lecture Notes in Computer Science 310.

Wanda. (2015). http://wandahot.sourceforge.net/.

Weiermann, A. (1998). How is It That Infinitary Methods can be Applied to Finitary Mathematics? Gödel's T: A Case Study. *Journal of symbolic logic*, **63**(4), 1348–1370.

Werner, B. (1994). *Une théorie des constructions inductives*. Ph.D. thesis, Université Paris 7, France.

Wilken, G., & Weiermann, A. (2012). Derivation Lengths Classification of Gödel's T Extending Howard's Assignment. *Logical methods in computer science*, **8**(1), 1–44.

Xi, H. (2002). Dependent types for program termination verification. *Journal of higher-order and symbolic computation*, **15**(1), 91–131.

Xi, H. (2003). Applied Type System (extended abstract). *Proceedings of the International Workshop on Types for Proofs and Programs*, Lecture Notes in Computer Science 3085.

Xi, H., Chen, C., & Chen, G. (2003). Guarded recursive datatype constructors. *Proceedings of the 30th ACM Symposium on Principles of Programming Languages*.

Zantema, H. (1995). Termination of term rewriting by semantic labelling. *Fundamenta informaticae*, **24**, 89–105.

Zenger, C. (1997). Indexed types. *Theoretical computer science*, **187**(1-2), 147–165.