Department of Philosophy - Utrecht University

Proving Termination of Higher-order Rewrite Systems

J.C. van de Pol

Logic Group
Preprint Series
No. 93
June 1993

Department of Philosophy
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
The Netherlands

ISBN 90-393-0257-x
ISSN 0929-0710
Dr. A. Visser, Editor

Proving Termination of Higher-order Rewrite Systems

Jaco van de Pol
Department of Philosophy, Utrecht University
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
email: jaco@phil.ruu.nl

June 1993

Abstract

This paper deals with termination proofs for Higher-Order Rewrite Systems (HRSs), introduced in [Nip91, Nip93]. This formalism combines the computational aspects of term rewriting and simply typed lambda calculus. Our result is a proof technique for the termination of a HRS, similar to the proof technique "Termination by interpretation in a well-founded monotone algebra" described in [Zan93]. The resulting technique is as follows: Choose a higher-order algebra with operations for each function symbol in the HRS, equipped with some well-founded partial ordering. The operations must be strictly monotonic in this ordering. This choice generates a model for the HRS. If the choice can be made in such a way that for each rule and for each valuation of the free variables in that rule the value of the left hand side is greater than the value of the right hand side, then the HRS is terminating. At the end of the paper two applications of this technique are given, which show that this technique is natural and can easily be applied.

1 Introduction

In the field of automated proof verification one sees a development towards higher-order concepts. In the generic theorem prover Isabelle [Pau90], typed lambda calculus is used as the syntax for the formulae. In other systems, as Coq [PM93], typed lambda calculus is even used for the logic, using the Curry-Howard isomorphism which links formulae to types and proofs to terms.
This development is mirrored in the research on Term Rewriting Systems (TRS). There are different formalisms dealing with the combination of term rewriting and an abstraction mechanism. In [Klo80] the concept of Combinatory Reduction Systems (CRS) was introduced. These systems essentially are TRSs with bound variables. In [Nip91, Nip93] the formalism of Higher-order Rewrite Systems (HRS) is described, which is very similar to CRSs in essence, but rather different in presentation. A precise comparison is given in [vOvR93]. A more general setting is given in [Wol93]. Quite another approach can be found in [Bre88].
Two important questions about rewrite systems are termination and confluence. For results about local confluence of HRSs and confluence of orthogonal HRSs the reader is referred to [Nip91] and [Nip93] respectively. In [Klo80, C. II.3] the confluence of regular CRSs is proved. However, the question of termination of the higher-order frameworks seems hardly to have been explored. As far as we know, only in [Klo80, C. II.6.2] a sufficient condition for termination of regular CRSs is given. With this condition, stated in terms of redexes and descendants, a termination proof for CRSs remains a syntactical matter.

Termination of first-order Term Rewriting is already an undecidable problem. But as the termination of TRSs is an interesting question, many semi-algorithms and characterisations of termination are proposed in the literature.

A nice characterisation of termination is given in [Zan93]. The function symbols of a TRS $\mathcal{R}$ have to be interpreted as strictly monotonic operations in some well-founded algebra. This interpretation is extended to closed terms as a usual algebraic homomorphism. Now the associated rewrite relation is terminating if every left hand side is greater (under the chosen interpretation) than the belonging right hand side, for each possible interpretation of the variables in that rule.
The strength of this characterisation is that one can concentrate on the "intuitive reason" for termination. This intuition can be translated in suited operations on well-founded orderings, thus using semantical arguments. The real termination proof consists of testing a simple condition on the rules only instead of on all possible rewrite steps or all possible redexes. This semantical approach is more convenient than a syntactical technique.

The aim of this paper is to generalise this semantical characterisation of termination for TRSs to one for HRSs. We use an extension of the definition of an HRS in [Nip93] because we do not need the restrictions of the formalism (for instance, that the rules should be of base type and the left hand sides should be patterns). Our result is also applicable to the HRSs of the definition in [Nip91, Nip93].
The main result is that such a generalisation is possible. The interpretation of terms can be extended to the interpretation of higher-order terms. The orderings and the notion of strictness can also be generalised. The techniques to achieve this are similar to those used in [Gan80, dV87]. Moreover, the result that termination proofs can be given with a well-founded monotone algebra in [Zan93] carries over to HRSs with simple conditions on the well-founded ordering. With this technique some natural HRSs are proved to be terminating (see Section 7.)

I like to thank Jan Friso Groote, who supplied some crucial ideas. I am also grateful to Vincent van Oostrom, Jan Bergstra, Tonny Hurkens and Jan Springintveld for marking previous versions of this document. Finally Marc Bezem and Alex Sellink helped me with various discussions and comments.

2 Term Rewriting and Simply Typed Lambda Calculus

2.1 Types and Terms of Simply Typed Lambda Calculus

In this section the sets of types and terms of simply typed lambda calculus are defined. The types are constructed from a set of base types. The terms are constructed from typed constants and variables. Let $\mathcal{B}$ be the set of *base types*. Then the set $\mathbb{T}(\mathcal{B})$ of *simple types* over these base types is defined as:

Definition 2.1. *$\mathbb{T}(\mathcal{B})$ is the smallest set satisfying*

- $\mathcal{B} \subseteq \mathbb{T}(\mathcal{B})$
- *If $\sigma, \tau \in \mathbb{T}(\mathcal{B})$ then also $\sigma \to \tau \in \mathbb{T}(\mathcal{B})$.*

Let $\mathcal{C}_\tau$ be the set of *function symbols* (or constants) of type τ. The union $\bigcup \mathcal{C}_\tau$ is written as $\mathcal{C}$. Similarly, $\mathcal{V}_\tau$ and $\mathcal{V}$ are sets of *typed variables*. To collect this information we define the notion of a *signature*:

Definition 2.2. *A signature ($\mathcal{F}$) is a triple $(\mathcal{B}, \mathcal{C}, \mathcal{V})$, where $\mathcal{C}$ is a family of typed constants and $\mathcal{V}$ is a family of typed variables.*

Given a signature $\mathcal{F}$ we can define the set of *simply typed lambda terms*, denoted by $\Lambda^{\to}(\mathcal{F})$:

Definition 2.3. *Let $\mathcal{F}$ be the signature $(\mathcal{B}, \mathcal{C}, \mathcal{V})$. The sets $\Lambda^{\to}_\tau(\mathcal{F})$ (with $\tau \in \mathbb{T}(\mathcal{B})$) are defined inductively by:*

- $\mathcal{V}_\tau \subseteq \Lambda^{\to}_\tau(\mathcal{F})$,

- $\mathcal{C}_\tau \subseteq \Lambda^{\rightarrow}_\tau(\mathcal{F})$,
- *If* $m \in \Lambda^{\rightarrow}_{\sigma\rightarrow\tau}(\mathcal{F})$ *and* $n \in \Lambda^{\rightarrow}_\sigma(\mathcal{F})$, *then* $mn \in \Lambda^{\rightarrow}_\tau(\mathcal{F})$ *(application),*
- *If* $x \in \mathcal{V}_\sigma$ *and* $m \in \Lambda^{\rightarrow}_\tau(\mathcal{F})$, *then* $\lambda x.m \in \Lambda^{\rightarrow}_{\sigma\rightarrow\tau}(\mathcal{F})$ *(abstraction).*

The set of well-typed terms over the signature $\mathcal{F}$ is $\bigcup_{\tau\in\mathbb{T}(\mathcal{B})}\Lambda^{\rightarrow}_\tau(\mathcal{F})$ and is denoted by $\Lambda^{\rightarrow}(\mathcal{F})$.

Definition 2.4. *The set of free variables of a well-typed term* t *(*$FV(t)$*)is defined inductively as:*

- $FV(x) = \{x\}$ *for* $x \in \mathcal{V}$
- $FV(c) = \varnothing$ *for* $c \in \mathcal{C}$
- $FV(mn) = FV(m) \cup FV(n)$
- $FV(\lambda x.m) = FV(m) \setminus \{x\}$

In the sequel we often abbreviate $\mathbb{T}(\mathcal{B})$ and $\Lambda^{\rightarrow}(\mathcal{F})$ to $\mathbb{T}$ and $\Lambda^{\rightarrow}$. A variable x is called *bound* if it occurs in a subterm of the form $\lambda x.s$. Terms that only differ in the renaming of bound variables (known as α-conversion) are identified. This permits us to stick to the convention that variables never occur free and bound as well in any mathematical context. See [Bar84, 26] for details about the *variable convention.*
A *substitution* is a mapping from variables to terms of the same type. More precisely:

Definition 2.5. *A substitution* θ *is a finite function* $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, *with* $t_i \in \Lambda^{\rightarrow}_\tau$ *if* $x_i \in \mathcal{V}_\tau$. *The set* $\{x_1, \ldots, x_n\}$ *is called the domain of this substitution, denoted by* $DOM(\theta)$.

Substitutions are extended to homomorphisms on terms in the following standard way:

Definition 2.6.

- $\theta(x) = x$, *if* $x \in \mathcal{V}$ *and* $x \notin DOM(\theta)$.
- $\theta(c) = c$, *if* $c \in \mathcal{C}$.
- $\theta(mn) = (\theta(m))(\theta(n))$
- $\theta(\lambda x.m) = \lambda x.\theta(m)$.

Note that in the last case no variable can be bound due to the variable convention. Other notations of $\theta(t)$ are t^θ or $t[x_1, \ldots, x_n := t_1, \ldots, t_n]$. This last notation is justified because in fact we defined simultaneous substitution in Definition 2.6.
To identify a particular subterm of a term the notion of *positions* is used. This notion is not formalised here. A *context* is just a $\Lambda^{\rightarrow}$-term, which we like to see as a context. A context is denoted by C. The subterm of C at position p is denoted by $C|_p$. $C[s]_p$ denotes term C with the subterm at position p replaced by s. In this replacement free variables of s can be bound by λ-binders in C. This is an exception of the variable convention.
We recall the β-reduction scheme for the lambda calculus, denoted by $\rightarrow_\beta$:

Definition 2.7. *The relation* $\rightarrow_\beta$ *is the compatible closure of the relation* $\{((\lambda x.m)n, m[x := n]) | m \in \Lambda^{\rightarrow}$ *and if* $x \in \mathcal{V}_\sigma$ *then* $n \in \Lambda^{\rightarrow}_\sigma\}$

Standard theory tells us that every $\Lambda^{\rightarrow}_\tau$-term has a unique *normal form* with respect to β-conversion. This β-normal form is denoted by $t{\downarrow_\beta}$. Normal forms are always of the form $\lambda x_1.\cdots\lambda x_n.(at_1\cdots t_m)$, where $a \in \mathcal{V} \cup \mathcal{C}$. The reflexive, symmetric and transitive closure of $\rightarrow_\beta$ is denoted by $\equiv_\beta$. ($s \equiv_\beta t$ is equivalent to $s{\downarrow_\beta} = t{\downarrow_\beta}$).

2.2 Higher-order Rewrite Systems

There are various definitions of higher-order rewrite mechanisms in recent literature [Bre88, Klo80, Nip91, Nip93, Wol93]. The definition in this subsection is not meant to add a new formalism to the existing ones. The conditions on the rules are dropped, because they are not necessary in the proof. The rewrite relation is as liberal as possible. Of course, our result applies to formalisms admitting fewer rules and fewer rewrite steps. The chosen formalism is much like the formalism in [Wol93, Ch. 4.1], but it is presented in a style resembling [Nip93]. The main difference is that we do not consider η-conversion for expository reasons. It is not difficult to extend the proof for a formalism using $\beta\eta$-long normal forms instead of β-normal forms.
A *Higher-order Rewrite System* is given by a signature and a set of rules:

Definition 2.8. *A* Higher-order Rewrite System $\mathcal{R}$ *is a tuple* $(\mathcal{F}, R)$*, where* $\mathcal{F}$ *is a signature and* R *is a set of rules in this signature. A* rule *is a pair* (l, r)*, with* $l, r \in \Lambda^{\rightarrow}_{\tau}(\mathcal{F})$ *in* β*-normal form.*

Rules (l, r) are denoted by $l \rightarrow r$. In this rule, l is called the left hand side and r the right hand side. Note that the definition of HRS in [Nip93, p. 308] is a special case of Definition 2.8, because we do not require that the rules are of base type and that the left hand side is a pattern. Nor do we require that the left hand side is not a variable. These restrictions are not necessary to obtain the main theorem.

It is not enough to define rules. We also have to specify how to use the rules. In the first order case term rewriting is just replacing subterms that are instances of left hand sides (redexes) by the corresponding right hand sides. In the higher-order case it is not immediately clear what is meant by a redex and it is not clear when a subterm occurs in another term. We first concentrate on occurrences. It is desirable that the rewrite relation is compatible with β-equivalence. One could try to define that a redex occurs in s if it syntactically occurs in some t with $t \equiv_\beta s$. But this is not satisfactory:

Example 2.9. *Let* $l \rightarrow r$ *be some rule of type* σ*. The redex* l *occurs in every term* t*, because* $t \equiv_\beta (\lambda x.t)l$*, for some fresh variable* x*. This rewrites to* $(\lambda x.t)r$*. If we look at rewriting modulo* β*-congruence we have the step* $t \rightarrow t$*, which is difficult to prove terminating.*

A second approach is to define that a redex occurs in a term t if it *syntactically* occurs (in other words: if it is a subterm). But this delivers a rewrite relation which is not compatible with β-conversion, as can be seen in example 2.9: $(\lambda x.t)l$ reduces to $(\lambda x.t)r$ if we ask for syntactical occurrences, but the step $t \rightarrow t$ is impossible in general. In fact what we want are the occurrences that cannot disappear by β- reduction, forbidding both $(\lambda x.t)l$ and t to rewrite.
In [Nip93, 308] this problem is tackled by defining that a subterm occurs in a term only if that term is in normal form. But this is too restrictive if the rules are not of base type:

Example 2.10. *Let the only rule be* $\lambda x.cxd \rightarrow r$ *for some constants* c *and* d*. The term* $(\lambda x.cxd)a$ *can not be rewritten, because the normal form of this term is* cad*, which does not contain the left hand side* $\lambda x.cxd$*.*

We can weaken this restriction by only requiring the context in normal form and not the whole term. A context can be seen as a term with a special variable $\Box$. Then in the previous example the rewrite step can take place, because the context is $\Box a$, which is in normal form. Both $(\lambda x.cxd)a$ and cad can be rewritten. This approach is used in Definition 2.11.

The other open question is about redexes. In the first order case a redex is the result of a substitution on the left hand side of a rule. The higher-order formalisms of [Klo80] and [Nip93] diverge in defining a redex. In [Nip93, p. 308] a redex is defined to be the β-normal form of the result of a substitution. In [Klo80, p. 126] a redex is just the result of a substitution. The difference between them is that in HRSs the β-normalisation of a redex is carried out implicitly by the definition of rewriting, while

in CRSs this β-normalisation leads to explicit rewrite steps. In [vOvR93] a precise statement of the difference between CRSs and HRSs is given.
We like to see simply typed lambda calculus as our meta–language and the properties of it (as β-reduction) should not be confused with the rewrite relation. So β-normalisation should take place implicitly. In the rewrite relation only steps derived from the rewrite rules appear. However, as new β-redexes can be generated by plugging a term into a context (see example 2.10) also these redexes are to be removed. (The other extreme would be to see the β-rule on the same level as the rewrite rules. This gives a sort of TRS $\oplus$ β-rule. See for such a formalism [Bre88]).
These considerations lead to the following definition of the rewrite relation $\rightarrow_{\mathcal{R}}$ on β-normal forms:

Definition 2.11. *Let $\mathcal{R} = (\mathcal{F}, R)$ be an HRS. The relation $\rightarrow_{\mathcal{R}}$ is the smallest relation satisfying: If*

- $(l \rightarrow r) \in \mathcal{R}$,
- *θ is a substitution,*
- *C is a term in β-normal form and*
- *p is a position in C*

then $(C[(l^\theta)]_p)\downarrow_\beta \rightarrow_{\mathcal{R}} (C[(r^\theta)]_p)\downarrow_\beta$.

Definition 2.12. *An* infinite rewrite sequence *is a sequence $(s_i)_{i\in\mathbb{N}}$, with $s_i \in \Lambda^{\rightarrow}_{\tau}$ for some $\tau \in \mathbb{T}$, such that $s_i \rightarrow_{\mathcal{R}} s_{i+1}$ for all $i \in \mathbb{N}$. An HRS is said to be* terminating *if there is no infinite rewrite sequence.*

3 An ordering which is closed under substitution

We try to apply the general idea of the proof technique "termination by interpretation" for TRSs in [Zan93] to HRSs. The outline of this technique is as follows: The function symbols are interpreted by operations of the same arity in an algebra, equipped with a well-founded partial order. This interpretation is extended to the terms of the TRS in an algebraic way. The interpretation is chosen in such a way that for all rules, the left hand side is interpreted by a greater value than the right hand side. If such an interpretation can be found, the TRS is terminating. To prove the correctness of this technique, first we have to show that the ordering on terms is closed under substitution. So, if $l > r$ for a rewrite rule $l \rightarrow r$, then we also have $l^\theta > r^\theta$ for substitutions θ. The other step is to show that the ordering is closed under placing terms into a context. This can be proved using the fact that the function symbols are interpreted by strictly monotonic functions, thus preserving the ordering. Now we have for any context C that $C[l^\theta] > C[r^\theta]$. This is the exact form of a rewrite step, thus showing that a rewrite step can be translated to a decrease in the well-founded ordering. In this way termination of rewriting is guaranteed.
In the higher-order case matters are more complicated. First of all, notions of interpretation, strictness and ordering have to be extended to higher-order terms. After these definitions the same idea can be used. It will turn out that we have to use two different orderings. The condition on the rules of an HRS is stated in terms of one ordering, so for every rule $(l \rightarrow r)$ it must be the case that $l >_1 r$. We show that this ordering is closed under substitution, so for any substitution $l^\theta >_1 r^\theta$. Now we use a second ordering to show that for any context C, $C[l^\theta] >_2 C[r^\theta]$. This second ordering will be well-founded, thus proving termination of the HRS.
In this chapter we will define an interpretation for the terms into the hereditarily monotonic functions. A similar idea occurs already in [Gan80] and [dV87]. One difference is that in this paper two β-equivalent terms have the same interpretation.

3.1 Interpretation of types by monotonic function spaces

Let us start with the type interpretation $\mathcal{I}$ of the set of base types $\mathcal{B}$:

Definition 3.1. *A type interpretation $\mathcal{I}$ is a set of strict partial orders $\{(\mathcal{D}_B, >_B) | B \in \mathcal{B}\}$. These sets generate a type structure in the following way: $\mathcal{D}_{\sigma\to\tau} = \{f : \mathcal{D}_\sigma \to \mathcal{D}_\tau\}$*

The sets $\mathcal{D}_\rho$ are function spaces with currying and usual application (denoted by $f(x)$) and extensional equality: $f = g$ if and only if for all x in the appropriate domain $f(x) = g(x)$.
We will see that only monotonic functionals will be used. This idea is also used in [Gan80, p. 457], which shows that terms of the typed λ-I calculus denote strictly monotonic functions. The sets of *hereditarily $\geq$-monotonic functions* (monotonic functions for short) are denoted by $\mathcal{M}_\rho$ and depend on an ordering, denoted by ${}_{\text{mon}}\geq$. This partial order is inherited from the partial order on the interpretation of the base types in the way defined below. This ordering itself depends on the notion of monotonicity, so we give a definition by simultaneous induction:

Definition 3.2. *The sets $\mathcal{M}_\rho$ of* monotonic *functions in $\mathcal{D}_\rho$ and the relation ${}_{\text{mon}}\geq_\rho$ on these sets are defined for each $\rho \in \mathbb{T}$, with induction on ρ:*

- *For $\rho \in \mathcal{B}$:*
  - $\mathcal{M}_\rho = \mathcal{D}_\rho$.
  - $a \;{}_{\text{mon}}\geq_\rho\; b$ *for $a, b \in \mathcal{M}_\rho$ if and only if $a >_\rho b$ or $a = b$.*
- *For $\rho = \sigma \to \tau$:*
  - *$f \in \mathcal{M}_\rho$ if and only if $f \in \mathcal{D}_\rho$ and for all $x \in \mathcal{M}_\sigma$, $f(x) \in \mathcal{M}_\tau$, and for all $x, y \in \mathcal{M}_\sigma$, if $x \;{}_{\text{mon}}\geq_\sigma\; y$ then $f(x) \;{}_{\text{mon}}\geq_\tau\; f(y)$.*
  - *$f \;{}_{\text{mon}}\geq_\rho\; g$ if and only if $f, g \in \mathcal{M}_\rho$ and for all x in $\mathcal{M}_\sigma$, $f(x) \;{}_{\text{mon}}\geq_\tau\; g(x)$.*

Instead of $a \in \mathcal{M}_\rho$ for some ρ we say: a is *monotonic*. Furthermore, a set $\mathcal{M}_\rho$ is called a *domain*. Now we can define the following strict partial order:

Definition 3.3. *The relation ${}_{\text{mon}}>_\rho$ on domains $\mathcal{M}_\rho$ with $\rho \in \mathbb{T}$ is defined with induction on ρ:*

- *If $\rho \in \mathcal{B}$ and $a, b \in \mathcal{M}_\rho$ then $a \;{}_{\text{mon}}>_\rho\; b$ if and only if $a >_\rho b$.*
- *If $\rho = \sigma \to \tau$ and $f, g \in \mathcal{M}_\rho$ then $f \;{}_{\text{mon}}>_\rho\; g$ if and only if for all $x \in \mathcal{M}_\sigma$, $f(x) \;{}_{\text{mon}}>_\tau\; g(x)$.*

The type subscripts in ${}_{\text{mon}}>_\rho$ and ${}_{\text{mon}}\geq_\rho$ are omitted when this is not confusing. Note that $f \;{}_{\text{mon}}> g$ means that f is pointwise greater than g, but only for monotonic points. The reader is warned not to confuse $f \;{}_{\text{mon}}\geq g$ with $f \;{}_{\text{mon}}> g \vee f = g$. We have the following fact about ${}_{\text{mon}}>$ and ${}_{\text{mon}}\geq$ only in one direction:

Proposition 3.4. *For all $\rho \in \mathbb{T}$ and $f, g \in \mathcal{M}_\rho$, if $f \;{}_{\text{mon}}> g$ or $f = g$, then $f \;{}_{\text{mon}}\geq g$.*

Proof: Let $f \;{}_{\text{mon}}> g$, or $f = g$. If $\rho \in \mathcal{B}$ it is trivial. If $\rho = \sigma \to \tau$, take an $x \in \mathcal{M}_\sigma$. Then $f(x) \;{}_{\text{mon}}> g(x)$ (in case $f \;{}_{\text{mon}}> g$) or $f(x) = g(x)$. But then, by induction hypothesis, $f(x) \;{}_{\text{mon}}\geq g(x)$. Now by Definition 3.2 $f \;{}_{\text{mon}}\geq g$. ■

The following proposition is handy:

Proposition 3.5. *Transitivity between ${}_{\text{mon}}>$ and ${}_{\text{mon}}\geq$:*

- *If $f \;{}_{\text{mon}}\geq g$ and $g \;{}_{\text{mon}}> h$ then $f \;{}_{\text{mon}}> h$.*
- *If $f \;{}_{\text{mon}}> g$ and $g \;{}_{\text{mon}}\geq h$ then $f \;{}_{\text{mon}}> g$.*

Proof: Simple induction on the type of f. ■

3.2 Interpretation of terms in domains

Now we can take our next step: A term in $\Lambda_\tau^\to$ has to be interpreted as a value in the domain $\mathcal{M}_\tau$. To interpret the terms we have to specify what the interpretation of the free variables and the constants will be. The free variables are dealt with by *valuations*: mappings from variables to values. The interpretation of constants is given by a *constant interpretation*.

Definition 3.6. *A valuation α is a family of mappings α_ρ, with $\alpha_\rho : \mathcal{V}_\rho \to \mathcal{D}_\rho$ for $\rho \in \mathbb{T}$.*

Definition 3.7. *A* monotonic *valuation α is a family of mappings α_ρ, with $\alpha_\rho : \mathcal{V}_\rho \to \mathcal{M}_\rho$ for $\rho \in \mathbb{T}$.*

The following operation on valuations changes the value of one variable:

Definition 3.8. *If α is a valuation, $x \in \mathcal{V}_\sigma$ and $a \in \mathcal{M}_\sigma$, then $\alpha[x := a]$ is also a valuation, with*

- $\alpha[x := a](x) = a$.
- $\alpha[x := a](y) = \alpha(y)$ *for* $y \neq x$.

To compare two monotonic valuations we define:

Definition 3.9. $\alpha_1 \geq \alpha_2$ *if and only if for each* $x \in \mathcal{V}$, $\alpha_1(x) \,{}_{\text{mon}}{\geq}\, \alpha_2(x)$.

The following facts about valuations are tacitly used in the sequel:

Proposition 3.10.

1. *If α is a monotonic valuation and a is monotonic then also $\alpha[x := a]$ is monotonic.*
2. *If $\alpha_1 \geq \alpha_2$ are monotonic valuations and a is monotonic, then $\alpha_1[x := a] \geq \alpha_2[x := a]$.*
3. *If α is a monotonic valuation and $a \,{}_{\text{mon}}{\geq}\, b$, then $\alpha[x := a] \geq \alpha[x := b]$.*

Proof: trivial ■

The constants have to be interpreted by functionals of the right domain. Therefore the following notion is introduced:

Definition 3.11. *A constant interpretation $\mathcal{J}$ is a family of functions $\mathcal{J}_\sigma : \mathcal{C}_\sigma \to \mathcal{M}_\sigma$ for $\sigma \in \mathbb{T}$.*

Now we are ready to define the interpretation of terms, depending on a particular choice for the constant interpretation $\mathcal{J}$.

Definition 3.12. *The interpretation of a term under the valuation α is defined inductively on the structure of the term:*

- $[\![x]\!]_\alpha = \alpha(x)$ *for* $x \in \mathcal{V}$
- $[\![c]\!]_\alpha = \mathcal{J}(c)$ *for* $c \in \mathcal{C}$.
- $[\![mn]\!]_\alpha = [\![m]\!]_\alpha [\![n]\!]_\alpha$.
- $[\![\lambda x.m]\!]_\alpha = \boldsymbol{\lambda} a \in \mathcal{D}_\sigma . [\![m]\!]_{\alpha[x:=a]}$ *for* $x \in \mathcal{V}_\sigma$.

It is easy to see that a term $s \in \Lambda_\sigma^\to$ is interpreted by a functional in $\mathcal{D}_\sigma$. However, we have to ensure that this functional is monotonic if the valuation is monotonic. This is proved by the following conjunct. The proof is by simultaneous induction and is similar to the proof in [dV87, p. 89]:

Theorem 3.13. *For each $s \in \Lambda^\to$ and monotonic valuations α, β:*

1. $[\![s]\!]_\alpha$ *is monotonic.*

2. *If* $\alpha \geq \beta$ *then* $[\![s]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![s]\!]_\beta$.

Proof: (induction on the structure of s)

- If $s = x \in \mathcal{V}$:

  1. $[\![s]\!]_\alpha = \alpha(x)$ is monotonic because α is so.
  2. $[\![s]\!]_\alpha = \alpha(x) \ {}_{\text{mon}}\!\geq \beta(x) = [\![s]\!]_\beta$. (From Definition 3.9).

- If $s = c \in \mathcal{C}$:

  1. $[\![s]\!]_\alpha = \mathcal{J}(c)$ is monotonic by definition.
  2. $[\![s]\!]_\alpha = \mathcal{J}(c) = [\![s]\!]_\beta$, so they are ${}_{\text{mon}}\!\geq$-related (Proposition 3.4).

- If $s = mn$:

  1. By induction hypothesis (1) both $[\![m]\!]_\alpha$ and $[\![n]\!]_\alpha$ are monotonic. Then by Definition 3.2, also $[\![s]\!]_\alpha = [\![m]\!]_\alpha[\![n]\!]_\alpha$ is monotonic.
  2. By induction hypothesis (2) $[\![m]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![m]\!]_\beta$. By induction hypothesis (1) $[\![n]\!]_\alpha$ is monotonic, so we have, with Definition 3.2, that $[\![m]\!]_\alpha[\![n]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![m]\!]_\beta[\![n]\!]_\alpha$. We also get from the induction hypotheses (1,2) that $[\![n]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![n]\!]_\beta$ and $[\![m]\!]_\beta$ is monotonic. Therefore, again with Definition 3.2 $[\![m]\!]_\beta[\![n]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![m]\!]_\beta[\![n]\!]_\beta$. Now, using transitivity of ${}_{\text{mon}}\!\geq$ we have $[\![s]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![s]\!]_\beta$.

- If $s = \lambda x.t$: (Say $x \in \mathcal{V}_\sigma$)

  1. Firstly, choose $a \in \mathcal{M}_\sigma$, then $[\![\lambda x.t]\!]_\alpha(a) = [\![t]\!]_{\alpha[x:=a]}$. This is monotonic by induction hypothesis (1). Secondly, let $a \ {}_{\text{mon}}\!\geq_\sigma b$. Then $\alpha[x := a] \geq \alpha[x := b]$, so by induction hypothesis (2) $[\![t]\!]_{\alpha[x:=a]} \ {}_{\text{mon}}\!\geq [\![t]\!]_{\alpha[x:=b]}$. This is equivalent to $[\![\lambda x.t]\!]_\alpha(a) \ {}_{\text{mon}}\!\geq [\![\lambda x.t]\!]_\alpha(b)$, so $[\![s]\!]_\alpha$ *is monotonic.*
  2. Let $a \in \mathcal{M}_\sigma$. We have $\alpha[x := a] \ {}_{\text{mon}}\!\geq \beta[x := a]$. So using induction hypothesis (2) we can compute: $[\![\lambda x.t]\!]_\alpha(a) = [\![t]\!]_{\alpha[x:=a]} \ {}_{\text{mon}}\!\geq [\![t]\!]_{\beta[x:=a]} = [\![\lambda x.t]\!]_\beta(a)$. So indeed, $[\![s]\!]_\alpha \ {}_{\text{mon}}\!\geq [\![s]\!]_\beta$.

■

Corollary 3.14. $[\![\cdot]\!]_\alpha$ *is a family of functions from* $\Lambda^{\rightarrow}_\sigma$ *to* $\mathcal{M}_\sigma$ *for each* $\sigma \in \mathbb{T}$.

We have an ordering on the domains and an interpretation mapping terms into domains. Using this interpretation we can define an ordering on terms. The free variables are dealt with by valuations:

Definition 3.15. *For terms* s, t *in* $\Lambda^{\rightarrow}$, $s \ {}_{\text{mon}}\!> t$ *if for each monotonic valuation* α, $[\![s]\!]_\alpha \ {}_{\text{mon}}\!> [\![t]\!]_\alpha$.

3.3 The ordering ${}_{\text{mon}}\!>$ is closed under substitution

It should not come as a surprise that ${}_{\text{mon}}\!>$ is closed under substitution. If two terms s and t are ${}_{\text{mon}}\!>$-ordered then for all monotonic valuations this ordering holds. Now a substitution in s and t leads to a new valuation in the interpretation of s and t, giving the required ordering. This idea is worked out in the rest of this subsection. Lemma 3.16 establishes the link between substitutions and valuations. In Proposition 3.17 the desired conclusion about substitutions is drawn. Again no new technique is used here, the following proofs resemble those in [Gan80, dV87].

Lemma 3.16. *Let θ be a substitution and α a monotonic valuation. Define a new valuation $\gamma(\alpha,\theta)$ as: $\gamma(\alpha,\theta)(x) = [\![x^\theta]\!]_\alpha$. Then for each term $s \in \Lambda^\rightarrow$ we have $[\![s^\theta]\!]_\alpha = [\![s]\!]_{\gamma(\alpha,\theta)}$. The new valuation is monotonic.*

Proof: (induction on the structure of s)

- $s = x \in \mathcal{V}$: by definition of γ.
- $s = c \in \mathcal{C}$: $[\![c^\theta]\!]_\alpha = [\![c]\!]_\alpha = \mathcal{J}(c) = [\![c]\!]_{\gamma(\alpha,\theta)}$.
- $s = mn$:

$$\begin{array}{rcll} [\![s^\theta]\!]_\alpha & = & [\![m^\theta n^\theta]\!]_\alpha & = [\![m^\theta]\!]_\alpha [\![n^\theta]\!]_\alpha \\ [\![m]\!]_{\gamma(\alpha,\theta)}[\![n]\!]_{\gamma(\alpha,\theta)} \text{ (by i.h.)} & = & [\![s]\!]_{\gamma(\alpha,\theta)}. & \end{array}$$

- $s = \lambda x.m$: Say $x \in \mathcal{V}_\sigma$.

  Claim: *Let $a \in \mathcal{M}_\sigma$. In this case: $\gamma(\alpha[x := a], \theta) = (\gamma(\alpha,\theta))[x := a]$.*

  Proof: This depends on the variable convention: $x \notin DOM(\theta)$, so $x^\theta = x$, and for all $y \in DOM(\theta)$, $x \notin FV(y^\theta)$. Under this convention, $\gamma(\alpha[x := a], \theta)(x) = [\![x^\theta]\!]_{\alpha[x:=a]} = a = (\gamma(\alpha,\theta)[x := a])(x)$. And for $y \neq x$: $\gamma(\alpha[x := a], \theta)(y) = [\![y^\theta]\!]_{\alpha[x:=a]} = [\![y^\theta]\!]_\alpha = \gamma(\alpha,\theta)(y) = (\gamma(\alpha,\theta)[x := a])(y)$ ■

  Now this case can be proved by the following calculation:

$$\begin{array}{rcll} [\![s^\theta]\!]_\alpha & = & [\![\lambda x.(m^\theta)]\!]_\alpha & \\ & = & \boldsymbol{\lambda} a \in \mathcal{M}_\sigma.([\![m^\theta]\!]_{\alpha[x:=a]}) & \\ & = & \boldsymbol{\lambda} a \in \mathcal{M}_\sigma.([\![m]\!]_{\gamma(\alpha[x:=a],\theta)}) & \text{(by i.h.)} \\ & = & \boldsymbol{\lambda} a \in \mathcal{M}_\sigma.([\![m]\!]_{(\gamma(\alpha,\theta))[x:=a]}) & \text{(by claim)} \\ & = & [\![\lambda x.m]\!]_{\gamma(\alpha,\theta)} & \end{array}$$

■

Proposition 3.17. *If for terms s, t in $\Lambda^\rightarrow_\tau$, $s \ {}_{\text{mon}}\!> t$ then for each substitution θ, $s^\theta \ {}_{\text{mon}}\!> t^\theta$.*

Proof: Let $s \ {}_{\text{mon}}\!> t$. Then for arbitrary monotonic valuation α:

$$\begin{array}{rcll} [\![s^\theta]\!]_\alpha & = & [\![s]\!]_{\gamma(\alpha,\theta)} & \text{by Lemma 3.16} \\ & {}_{\text{mon}}\!> & [\![t]\!]_{\gamma(\alpha,\theta)} & \text{because } s \ {}_{\text{mon}}\!> t \\ & = & [\![t^\theta]\!]_\alpha & \text{by Lemma 3.16,} \end{array}$$

so indeed $s^\theta \ {}_{\text{mon}}\!> t^\theta$. ■

4 Hereditarily monotonic functionals serve as a model of $\lambda^\rightarrow$

We still have to show that the interpretation is well-behaved with respect to the β-rule. This is the same as showing that our interpretation is a model of $\lambda^\rightarrow$.

Proposition 4.1. *If $s \rightarrow_\beta t$ for two terms in $\Lambda^\rightarrow$, then $[\![s]\!]_\alpha = [\![t]\!]_\alpha$ for each valuation α.*

Proof: (induction on the structure of s) Let $s \rightarrow_\beta t$.

- $s \in \mathcal{V}$: impossible (contains no β-redex)
- $s \in \mathcal{C}$: idem

- $s = \lambda x.s_1$. Then the only possibility for a β-redex is in s_1. So $t = \lambda x.t_1$, with $s_1 \to_\beta t_1$. Now we can compute: $[\![s]\!]_\alpha = \boldsymbol{\lambda} a.[\![s_1]\!]_{\alpha[x:=a]} =$ (by induction hypothesis) $\boldsymbol{\lambda} a.[\![t_1]\!]_{\alpha[x:=a]} = [\![\lambda x.t_1]\!]_\alpha$.
- $s = s_1 s_2$. Now we have to consider three cases:
  - $t = t_1 s_2$ and $s_1 \to_\beta t_1$. Then $[\![s]\!]_\alpha = [\![s_1]\!]_\alpha [\![s_2]\!]_\alpha =$(by induction hypothesis) $[\![t_1]\!]_\alpha [\![s_2]\!]_\alpha = [\![t]\!]_\alpha$.
  - $t = s_1 t_2$ and $s_2 \to_\beta t_2$. Then $[\![s]\!]_\alpha = [\![s_1]\!]_\alpha [\![s_2]\!]_\alpha =$(by induction hypothesis) $[\![s_1]\!]_\alpha [\![t_2]\!]_\alpha = [\![t]\!]_\alpha$.
  - $s_1 = \lambda x.s_3$ and $t = s_3[x := s_2]$. The following fact is used:

    Claim: $[\![s_3]\!]_{\alpha[x:=[\![s_2]\!]_\alpha]} = [\![s_3[x := s_2]]\!]_\alpha$

    Proof: Define the substitution $\theta = \{x \mapsto s_2\}$. Using the definition of Lemma 3.16, $\gamma(\alpha, \theta) = \alpha[x := [\![s_2]\!]_\alpha]$. Furthermore, $s_3[x := s_2] = s_3^\theta$. Now, by Lemma 3.16 we have that $[\![s_3]\!]_{\gamma(\alpha,\theta)} = [\![s_3^\theta]\!]_\alpha$, which is what we wanted. ■

    Using this claim a simple calculation shows:

$$\begin{array}{rclcl} [\![s]\!]_\alpha & = & (\boldsymbol{\lambda} a.[\![s_3]\!]_{\alpha[x:=a]})[\![s_2]\!]_\alpha & = & [\![s_3]\!]_{\alpha[x:=[\![s_2]\!]_\alpha]} \\ & = & [\![s_3[x := s_2]]\!]_\alpha & = & [\![t]\!]_\alpha \end{array}$$

■

5 An ordering which is closed under context

5.1 On strictness

In section 3 we saw that the ordering $_{\text{mon}}>$ on terms is closed under substitution. We would like that this ordering is also closed under placing a term into a context. The first objective to this is the interpretation of the constants. We have to ensure that this interpretation is order preserving. The proof in [Zan93] also uses the condition that the constants have to be interpreted by *strictly monotonic* operations. Therefore we define the following notion:

Definition 5.1.[1] *The predicate "f is strict for $_{\text{mon}}>$" with $f \in \mathcal{M}_\rho$ is defined with induction on ρ:*

- *For $\rho \in \mathcal{B}$: $f \in \mathcal{M}_\rho$ is always strict for $_{\text{mon}}>$*
- *For $\rho = \sigma \to \tau$: $f \in \mathcal{M}_\rho$ is strict for $_{\text{mon}}>$ if and only if for all $x \in \mathcal{M}_\sigma$, $f(x)$ is strict for $_{\text{mon}}>$, and for all $x, y \in \mathcal{M}_\sigma$, if $x \,{}_{\text{mon}}>_\sigma y$ then $f(x) \,{}_{\text{mon}}>_\tau f(y)$.*

Unfortunately this is not strong enough, as the following example shows:

Example 5.2. *Take the following signature: There is one base type: o. The set of constants $\mathcal{C} = \{c, d : o; @ : ((o \to o) \to o) \to (o \to o) \to o\}$. The set of variables $\mathcal{V} = \{x, y : o; f : o \to o\}$. In this signature we define the HRS with one rule:*

$$@(\lambda f.fx)(\lambda y.c) \to @(\lambda g.g(@(\lambda f.fx)(\lambda y.c)))(\lambda y.d).$$

First of all, this HRS is not terminating, because the left hand side is a subterm of the right hand side. Now we choose as interpretation:

$$\begin{array}{rcl} \mathcal{I}(o) & = & \mathbb{N} \\ \mathcal{J}(c) & = & 2 \\ \mathcal{J}(d) & = & 1 \\ \mathcal{J}(@) & = & \boldsymbol{\lambda} G \in ((\mathbb{N} \to \mathbb{N}) \to \mathbb{N}).\boldsymbol{\lambda} g \in \mathbb{N} \to \mathbb{N}.G(g) + g(0) \end{array}$$

[1] This definition is only used for expository reasons. It is not really used in the sequel.

Under this interpretation we can compute the left hand side and the right hand side. It will turn out that the left hand side equals 4 and the right hand side equals 2. Yet, $\mathcal{J}(@)$ is strict for $_{\text{mon}}>$ in the sense of Definition 5.1. So this notion of strictness is too weak.

This lack is caused by a problem with the variables. This is shown in the following example:

Example 5.3. *Let $l \to r$ be the only rule of an HRS, with $l \,_{\text{mon}}> r$. Choose as context $C = x\Box$, with $x \in \mathcal{V}_{\sigma\to\tau}$ fresh. We wonder if $C[l] \,_{\text{mon}}> C[r]$. This is the case only if for all monotonic valuations α, $[\![xl]\!]_\alpha \,_{\text{mon}}> [\![xr]\!]_\alpha$. But if we take some constant $c \in \mathcal{C}_\tau$ and consider some monotonic valuation α with $\alpha(x) = \lambda y.c$ we observe that $[\![xl]\!]_\alpha = (\boldsymbol{\lambda} a.[\![c]\!]_\alpha)[\![l]\!]_\alpha = [\![c]\!]_\alpha = (\boldsymbol{\lambda} a.[\![c]\!]_\alpha)[\![r]\!]_\alpha = [\![xr]\!]_\alpha$. And because $_{\text{mon}}>$ is irreflexive, $C[l] \,_{\text{mon}}> C[r]$ is* not true.

Yet, this example does not show that the HRS of Example 5.3 is non-terminating (of cause it is terminating). It is too strong to demand that the relation $_{\text{mon}}>$ holds for every value of x. The variables of the context can not be instantiated during a rewrite step. So we have the freedom to restrict the condition and to look at some particular value of x. The idea, due to Jan Friso Groote, is to look at precisely those x that preserve the order, that is for the *strictly monotonic* x. This leads to a new ordering, $_{\text{str}}>$, which (intuitively) runs: $f \,_{\text{str}}> g$ if and only if for all strictly monotonic x, $f(x) \,_{\text{str}}> g(x)$. This new ordering is used to compare contexts.
The following example shows that the ordering $_{\text{mon}}>$ is necessary for the rules, so we can't use $_{\text{str}}>$ only:

Example 5.4. *Let $l, r \in \mathcal{C}_\sigma$ and $x \in \mathcal{V}_{\sigma\to\tau}$. The HRS $\mathcal{R}$ with the only rule $\lambda x.xl \to \lambda x.xr$ is non-terminating. Let $c \in \mathcal{C}_\tau$ and choose as context $\Box(\lambda y.c)$. This gives rise to the rewrite step: $((\lambda x.xl)(\lambda y.c))\downarrow_\beta \to_{\mathcal{R}} ((\lambda x.xl)(\lambda y.c))\downarrow_\beta$, which is equivalent to $c \to_{\mathcal{R}} c$. Now we have the infinite rewrite sequence $c \to_{\mathcal{R}} c \to_{\mathcal{R}} c \to_{\mathcal{R}} \cdots$.*
However, if we choose an interpretation with $\mathcal{J}(l) > \mathcal{J}(r)$, then $\lambda x.xl \,_{\text{str}}> \lambda x.xr$. Therefore, the $_{\text{str}}>$-relation is not strong enough for the rules.

These examples show that we need a definition of $_{\text{str}}>$ and a notion of strictness that switches from the $_{\text{str}}>$-ordering to the $_{\text{mon}}>$-ordering. These notions can be defined simultaneously and inductively on the structure of the types:

Definition 5.5. *The relation $_{\text{str}}>_\rho$ and the set $\mathcal{S}_\rho$ of hereditarily strict monotonic functions are defined with induction on $\rho \in \mathbb{T}$:*

- *For $\rho \in \mathcal{B}$:*
  - *$a \,_{\text{str}}>_\rho b$ for $a, b \in \mathcal{M}_\rho$ if and only if $a >_\rho b$.*
  - *$\mathcal{S}_\rho = \mathcal{M}_\rho$.*
- *For $\rho = \sigma \to \tau$: Let $f, g \in \mathcal{M}_\rho$, then*
  - *$f \,_{\text{str}}>_\rho g$ if and only if for all $x \in \mathcal{S}_\sigma$, $f(x) \,_{\text{str}}>_\tau g(x)$.*
  - *$f \in \mathcal{S}_\rho$ if and only if for all $x \in \mathcal{M}_\sigma$, $f(x) \in \mathcal{S}_\tau$, and for all $x, y \in \mathcal{M}_\sigma$, if $x \,_{\text{str}}>_\sigma y$ then $f(x) \,_{\text{mon}}>_\tau f(y)$.*

If $f \in \mathcal{S}_\rho$ for some $\rho \in \mathbb{T}$, we say that f is *strict* or strictly monotonic. Furthermore we will omit the type subscripts for $_{\text{str}}>_\rho$. Now we can prove the following relation between the different partial orders:

Proposition 5.6. *Let $\rho \in \mathbb{T}$ and $f, g \in \mathcal{M}_\rho$. If $f \,_{\text{mon}}>_\rho g$ then $f \,_{\text{str}}>_\rho g$.*

Proof: (induction on ρ):
If $\rho \in \mathcal{B}$ the two orderings coincide. Let $f \ {}_{\text{mon}}{>}_\rho \ g$ for $\rho = \sigma \to \tau$. Take an arbitrary strict $x \in \mathcal{S}_\sigma$. Then also $x \in \mathcal{M}_\sigma$, so $f(x) \ {}_{\text{mon}}{>}_\tau \ g(x)$. By induction hypothesis, $f(x) \ {}_{\text{str}}{>}_\tau \ g(x)$, giving $f \ {}_{\text{str}}{>} \ g$ (Definition 5.5). ■

Corollary 5.7. *Let f be strict, then both f is strict for ${}_{\text{mon}}{>}$ and f is strict for ${}_{\text{str}}{>}$.*

Proof: If $x \ {}_{\text{mon}}{>} \ y$ then $x \ {}_{\text{str}}{>} \ y$ (Proposition 5.6) and $f(x) \ {}_{\text{mon}}{>} \ f(y)$, by strictness of f. If $x \ {}_{\text{str}}{>} \ y$ then by strictness of f: $f(x) \ {}_{\text{mon}}{>} \ f(y)$ and by Proposition 5.6 $f(x) \ {}_{\text{str}}{>} \ f(y)$. ■

The notion of strictness is extended to valuations:

Definition 5.8. *A valuation α is strictly monotonic (*strict*) if for each $x \in \mathcal{V}$, $\alpha(x)$ is strictly monotonic.*

The following fact is used without referring to this proposition:

Proposition 5.9. *If α is a strict valuation and a is strict, then $\alpha[x := a]$ is also strict.*

Proof: trivial. ■

We can lift ${}_{\text{str}}{>}$ from the domain level into the term level in the same way as we lifted ${}_{\text{mon}}{>}$. To keep similarity between bound and free variables we now use strict valuations only.

Definition 5.10. *For terms s, t of $\Lambda^{\to}$, $s \ {}_{\text{str}}{>} \ t$ if for each* strict *valuation α, $[\![s]\!]_\alpha \ {}_{\text{str}}{>} \ [\![t]\!]_\alpha$.*

Proposition 5.11. *For all $s, t \in \Lambda^{\to}$, if $s \ {}_{\text{mon}}{>} \ t$ then $s \ {}_{\text{str}}{>} \ t$.*

Proof: If $s \ {}_{\text{mon}}{>} \ t$ for $s, t \in \Lambda^{\to}_\tau$ then for all valuations α, $[\![s]\!]_\alpha \ {}_{\text{mon}}{>}_\tau \ [\![t]\!]_\alpha$. This implies (using Proposition 5.6) that $[\![s]\!]_\alpha \ {}_{\text{str}}{>}_\tau \ [\![t]\!]_\alpha$. This holds for all α, so certainly for strict α this relation holds. Therefore $s \ {}_{\text{str}}{>} \ t$. ■

5.2 The ordering ${}_{\text{str}}{>}$ is well-founded

Definition 5.12. *A type interpretation $\mathcal{I} = \{(\mathcal{D}_B, >_B) | B \in \mathcal{B}\}$ is called* a well-founded type interpretation *if the following conditions on the interpretation of base types are satisfied:*

- *For each $b \in \mathcal{B}$, $\mathcal{D}_b$ is non-empty.*
- *For each $b \in \mathcal{B}$, $>_b$ is well-founded.*
- *For each $b, c, d \in \mathcal{B}$, the set $\mathcal{S}_{b \to c \to d}$ has at least one element, which is called $S_{b \to c \to d}$.*

This suggestive definition is justified by Proposition 5.14 which states that under a well-founded type interpretation, the domains are well-founded. But first, we need a lemma, saying that strict inhabitants exist for all domains, if we have a well-founded type interpretation:

Lemma 5.13. *If the underlying type interpretation is well-founded, then for every $\rho \in \mathbb{T}$ there is at least one inhabitant of $\mathcal{S}_\rho$, which we will call S_ρ.*

Proof: (with induction on the structure of ρ)
If ρ is of base type, we can choose any $S_\rho \in \mathcal{D}_\rho$, because the domains $\mathcal{D}_b$ are non-empty, and elements from $\mathcal{D}_b$ are always strict.

Now let $\rho = \sigma \to \tau$. Then σ and τ can be decomposed in $\sigma_1 \to \cdots \sigma_m \to c_0$ and $\tau_1 \to \cdots \tau_n \to c_1$, with $c_0, c_1 \in \mathcal{B}$. By induction hypothesis there exist strict inhabitants S_{σ_j} and S_τ. Because the definition of domains, we also have the strict element $S_{c_0 \to c_1 \to c_1}$. With this material we can define[2]:

$$S_\rho = \boldsymbol{\lambda} y \in \mathcal{D}_\sigma . \boldsymbol{\lambda} x_1 \in \mathcal{D}_{\tau_1} \cdots \boldsymbol{\lambda} x_n \in \mathcal{D}_{\tau_n} . S_{c_0 \to c_1 \to c_1}(y S_{\sigma_1} \cdots S_{\sigma_m})(S_\tau x_1 \cdots x_n)$$

Clearly, this functional is of the appropriate type and strict in y and $x_1, \ldots, x_n$. Try for example:

$$\begin{array}{ll} & y_1 \ {}_{\text{str}}{>}_\sigma \ y_2 \\ \Rightarrow & y_1 S_{\sigma_1} \cdots S_{\sigma_m} \ {}_{\text{str}}{>}_{c_0} \ y_2 S_{\sigma_1} \cdots S_{\sigma_m} \\ \Rightarrow & S_{c_0 \to c_1 \to c_1}(y_1 S_{\sigma_1} \cdots S_{\sigma_m}) \ {}_{\text{mon}}{>}_{c_1 \to c_1} \ S_{c_0 \to c_1 \to c_1}(y_2 S_{\sigma_1} \cdots S_{\sigma_m}) \\ \Rightarrow & S_\rho y_1 \ {}_{\text{mon}}{>}_\tau \ S_\rho y_2 \end{array}$$

A similar reasoning applies when some x_i is varied. ■

Proposition 5.14. *If the underlying type interpretation is well-founded, then for every $\rho \in \mathbb{T}$, the partial order $(\mathcal{M}_\rho, \ {}_{\text{str}}{>}_\rho)$ is well-founded.*

Proof: (induction on the structure of ρ)
For $\rho \in \mathcal{B}$ well-foundedness is given by the well-foundedness of $>_\rho$. Let $\rho = \sigma \to \tau$. Assume that there exist $\{f_i | i \in \mathbb{N}\}$ with $f_i \ {}_{\text{str}}{>}_{\sigma \to \tau} \ f_{i+1}$. But now, using $S_\sigma \in \mathcal{S}_\sigma$ we can construct the descending chain $\{f_i(S_\sigma) | i \in \mathbb{N}\}$, with $f_i(S_\sigma) \ {}_{\text{str}}{>}_\tau \ f_{i+1}(S_\sigma)$, which contradicts the induction hypothesis. ■

Proposition 5.15. *The relation ${}_{\text{str}}{>}$ is well-founded on terms of $\Lambda^{\to}$.*

Proof: Let $\{s_i | i \in \mathbb{N}\}$ of some type τ be given such that $s_i \ {}_{\text{str}}{>} \ s_{i+1}$. By Definition 5.10 this means that for every strict valuation α, $[\![s_i]\!]_\alpha \ {}_{\text{str}}{>}_\tau \ [\![s_{i+1}]\!]_\alpha$. Let α be the valuation $\{x \mapsto S_\sigma | x \in \mathcal{V}_\sigma, \sigma \in \mathbb{T}\}$ (see Proposition 5.13), then α is such a strict valuation, giving rise to a descending ${}_{\text{str}}{>}$ chain in $\mathcal{M}_\tau$, which contradicts Proposition 5.14. ■

5.3 Contexts are morphisms from ${}_{\text{mon}}{>}$ to ${}_{\text{str}}{>}$

In section 5.1 we showed that the interpretation of the constants should be strict. Therefore we define:

Definition 5.16. *A constant interpretation $\mathcal{J}$ is strict, if $\mathcal{J}(c)$ is strictly monotonic for all $c \in \mathcal{C}$.*

Now we can indeed prove, after some lemmas, that if two terms are ${}_{\text{mon}}{>}$-related then placing these terms into a context results in two ${}_{\text{str}}{>}$-related terms.

Lemma 5.17. *Let $a \in \mathcal{S}_{\sigma_1 \to \cdots \sigma_n \to \tau}$. Let elements $s_i \in \mathcal{M}_{\sigma_i}$ be given for $1 \leq i \leq n$. Let furthermore $s_j \ {}_{\text{str}}{>} \ t$, for some $1 \leq j \leq n$. Then $a s_1 \cdots s_n \ {}_{\text{mon}}{>} \ a s_1 \cdots s_{j-1} t s_{j+1} \cdots s_n$.*

Proof: By Definition 5.5 (and induction on j), $a s_1 \cdots s_{j-1}$ is strict. So by the same definition, $a s_1 \cdots s_{j-1} s_j \ {}_{\text{mon}}{>} \ a s_1 \cdots s_{j-1} t$. Now by Definition 3.3 (and induction on $n - j$) $a s_1 \cdots s_n \ {}_{\text{mon}}{>} \ a s_1 \cdots s_{j-1} t s_{j+1} \cdots s_n$. ■

Lemma 5.18. *If for two $\Lambda^{\to}_\tau$-terms $s \ {}_{\text{mon}}{>} \ t$ then $\lambda x.s \ {}_{\text{mon}}{>} \ \lambda x.t$.*

Proof: Let $s \ {}_{\text{mon}}{>} \ t$, say $x \in \mathcal{V}_\sigma$. Choose $k \in \mathcal{M}_\sigma$ arbitrarily. Let α be any monotonic valuation. Then

$$\begin{array}{rcll} [\![\lambda x.s]\!]_\alpha(k) & = & (\boldsymbol{\lambda} a.[\![s]\!]_{\alpha[x:=a]})(k) & \\ & = & [\![s]\!]_{\alpha[x:=k]} & \\ & {}_{mon}{>}_\tau & [\![t]\!]_{\alpha[x:=k]} & \text{Because } s \ {}_{\text{mon}}{>} \ t \\ & = & (\boldsymbol{\lambda} a.[\![t]\!]_{\alpha[x:=a]})(k) & \\ & = & [\![\lambda x.t]\!]_\alpha(k) & \end{array}$$

This is for any k, so $[\![\lambda x.s]\!]_\alpha \ {}_{\text{mon}}{>}_{\sigma \to \tau} \ [\![\lambda x.t]\!]_\alpha$. This holds for any α, so $\lambda x.s \ {}_{\text{mon}}{>} \ \lambda x.t$ ■

[2] See [Gan80, p. 461] and [dV87, p. 83] for comparable functionals

Lemma 5.19. *If for two $\Lambda^{\rightarrow}_{\tau}$ terms $s \ {}_{\text{str}}\!> t$ then $\lambda x.s \ {}_{\text{str}}\!> \lambda x.t$.*

Proof: Let $s \ {}_{\text{str}}\!> t$, let $x \in \mathcal{V}_\sigma$. Choose $k \in \mathcal{S}_\sigma$ arbitrarily. Let α be any strict valuation. Now

$$\begin{array}{rcll} [\![\lambda x.s]\!]_\alpha(k) & = & (\boldsymbol{\lambda} a.[\![s]\!]_{\alpha[x:=a]})(k) & \\ & = & [\![s]\!]_{\alpha[x:=k]} & \\ & {}_{\text{str}}\!>_\tau & [\![t]\!]_{\alpha[x:=k]} & \text{Because } s \ {}_{\text{str}}\!> t \text{ and } \alpha[x := k] \text{ strict.} \\ & = & (\boldsymbol{\lambda} a.[\![t]\!]_{\alpha[x:=a]})(k) & \\ & = & [\![\lambda x.t]\!]_\alpha(k) & \end{array}$$

This holds for any strict k and *alpha*, so $\lambda x.s \ {}_{\text{str}}\!> \lambda x.t$. ■

Proposition 5.20. *Let C be a term in normal form. Let p be a position within C. Let $\mathcal{J}$ be a strict constant interpretation. If $s \ {}_{\text{mon}}\!> t$ then $C[s]_p \ {}_{\text{str}}\!> C[t]_p$.*

Proof: C is of the form $\lambda x_1.\cdots\lambda x_n.(a s_1 \cdots s_m)$ with $a \in \mathcal{C} \cup \mathcal{V}$. Now there are three different cases for the position p within C:

1. $C|_p = \lambda x_i.\cdots\lambda x_n.(a s_1 \cdots s_m)$ with $1 \leq i \leq n$.
2. $C|_p = (a s_1 \cdots s_j)$ with $0 \leq j \leq m$
3. $C|_p = s_k|_q$ with $1 \leq k \leq m$ and q a postfix of p.

These cases are proved in the sequel:

1. $C[s]_p = \lambda x_1.\cdots\lambda x_{i-1}.s$ and $C[t]_p = \lambda x_1.\cdots.\lambda x_{i-1}.t$. We know that $s \ {}_{\text{mon}}\!> t$. Now apply Lemma 5.18 to obtain that $C[s]_p \ {}_{\text{mon}}\!> C[t]_p$. Then by Proposition 5.6 also $C[s]_p \ {}_{\text{str}}\!> C[t]_p$.
2. $C[s]_p = \lambda x_1.\cdots\lambda x_n.(s s_{j+1} \cdots s_m)$ and $C[t]_p = \lambda x_1.\cdots\lambda x_n.(t s_{j+1} \cdots s_m)$. Now $s \ {}_{\text{mon}}\!> t$, so $(s s_{j+1} \cdots s_m) \ {}_{\text{mon}}\!> (t s_{j+1} \cdots s_m)$ (Lemma 5.17). But now, by Lemma 5.18, $C[s]_p \ {}_{\text{mon}}\!> C[t]_p$. Then by Proposition 5.6 also $C[s]_p \ {}_{\text{str}}\!> C[t]_p$.
3. C is in normal form, so also s_k is. Now $C[s]_p = \lambda x_1.\cdots.\lambda x_n.(a s_1 \cdots (s_k[s]_q) \cdots s_m)$ and $C[t]_p = \lambda x_1.\cdots.\lambda x_n.(a s_1 \cdots (s_k[t]_q) \cdots s_m)$. By induction hypothesis $s_k[s]_q \ {}_{\text{str}}\!> s_k[t]_q$.

   Take a strict valuation α. We have that $a \in \mathcal{V} \cup \mathcal{C}$. If $a \in \mathcal{C}$, then $[\![a]\!]_\alpha = \mathcal{J}(a)$ is strict, because the constant interpretation is strict. Otherwise, if $a \in \mathcal{V}$ then $[\![a]\!]_\alpha = \alpha(a)$ is also strict, because α is a strict valuation. Now we can apply Lemma 5.17 to obtain $[\![a s_1 \cdots (s_k[s]_p) \cdots s_m]\!]_\alpha \ {}_{\text{mon}}\!> [\![a s_1 \cdots (s_k[t]_p) \cdots s_m]\!]_\alpha$.

   This being true for each strict valuation α gives us that

$$a s_1 \cdots (s_k[s]_p) \cdots s_m \ {}_{\text{str}}\!> a s_1 \cdots (s_k[t]_p) \cdots s_m$$

   Applying Lemma 5.19 we can put the lambda binders in front to get: $C[s]_p \ {}_{\text{str}}\!> C[s]_p$

■

6 Rewriting is Decreasing

Definition 6.1. *Let $\mathcal{R}$ be an HRS, with signature $\mathcal{F} = (\mathcal{B}, \mathcal{C}, \mathcal{V})$ and R a set of rules. We say that $\mathcal{R}$ has a* Reduction Interpretation *if there is a well-founded type interpretation $\mathcal{I}$ for $\mathcal{B}$ and a strict constant interpretation $\mathcal{J}$, such that for each rule $l \to r \in R$, $l \ {}_{\text{mon}}\!> r$.*

Now we are able to state the relation between a rewrite step and the ordering of the domains:

Proposition 6.2. *Let $\mathcal{R}$ have a Reduction Interpretation. If $s \to_{\mathcal{R}} t$ then $s \ {}_{\text{str}}\!> t$.*

Proof: Let $s \to_{\mathcal{R}} t$. Then $s = (C[l^\theta]_p)\downarrow_\beta$ and $t = (C[r^\theta]_p)\downarrow_\beta$, for some context in normal form C, position p in C, rule $(l \to r) \in R$ and substitution θ. Now $l \ {}_{\text{mon}}\!> r$, so by Lemma 3.17 $l^\theta \ {}_{\text{mon}}\!> r^\theta$. Now by Proposition 5.20, $C[l^\theta]_p \ {}_{\text{str}}\!> C[r^\theta]_p$. Applying Proposition 4.1 yields: $s \ {}_{\text{str}}\!> t$. ■

The main theorem of this paper uses the well-foundedness of the domains:

Theorem 6.3. *If a rule system $\mathcal{R}$ has a Reduction Interpretation, then $\mathcal{R}$ is terminating.*

Proof: If $\mathcal{R}$ is non-terminating, then there exists a sequence $(s_i)_{i\in\mathbb{N}}$, with $s_i \to_{\mathcal{R}} s_{i+1}$. By Proposition 6.2 we get an infinite descending chain $(s_i)_{i\in\mathbb{N}}$, with $s_i \ {}_{\text{str}}\!> s_{i+1}$. This is impossible because of the well-foundedness of ${}_{\text{str}}\!>$ (Proposition 5.15). Thus $\mathcal{R}$ must be terminating. ■

The following is a recollection of the conditions that appear in the previous sections. Theorem 6.3 suggests the following proof for the termination of an HRS:

- Find convenient domains $\mathcal{I}(B)$ for each base type B, satisfying:
  - $\mathcal{I}(B)$ is non-empty.
  - $\mathcal{I}(B)$ is well-founded.
  - There exist binary strictly monotonic functions with type $\mathcal{I}(a) \to \mathcal{I}(b) \to \mathcal{I}(c)$ for all combinations (a, b, c) of base types.
- Find a convenient strict interpretation $\mathcal{J}(c)$ for each constant symbol $c \in \mathcal{C}$.
- Prove that for each rule $(l \to r)$ in the system and for each monotonic valuation α of variables to values, the interpretation of the left hand side is greater than the interpretation of the right hand side, in symbols $[\![l]\!]_\alpha \ {}_{\text{mon}}\!> [\![r]\!]_\alpha$.

In the next section the applicability of this proof method is shown.

7 Applications

7.1 Process Algebra

The first application comes from Process Algebra, or better an extension of it: μCRL [GP90]. We only concentrate on the fragment of Process Algebra with choice ($+$), sequential composition ($\cdot$) and deadlock (δ) and the data dependent choice (Σ) from μCRL. The Process Algebra part can be formulated in a first order Term Rewriting System (see for instance [AB91]). The rules for the Sum-operator require higher-order rewrite rules to deal with the bound variables. This reformulation of μCRL can be found in [Sel93, p. 33].
There are two base types: $\{\mathsf{Proc}, \mathsf{Data}\}$. Furthermore, here is a list of function symbols with their types:

$$\begin{aligned} + &: \ \mathsf{Proc} \to \mathsf{Proc} \to \mathsf{Proc} \\ \cdot &: \ \mathsf{Proc} \to \mathsf{Proc} \to \mathsf{Proc} \\ \delta &: \ \mathsf{Proc} \\ \Sigma &: \ (\mathsf{Data} \to \mathsf{Proc}) \to \mathsf{Proc} \end{aligned}$$

The set of free variables is $\{X, Y, Z, P, Q, D\}$. Now we have the following set of rules, with the binary function symbols written infix:

$$\begin{array}{lrcl}
\text{A3:} & X + X & \to & X \\
\text{A4:} & (X + Y) \cdot Z & \to & (X \cdot Z) + (Y \cdot Z) \\
\text{A5:} & (X \cdot Y) \cdot Z & \to & X \cdot (Y \cdot Z) \\
\text{A6:} & X + \delta & \to & X \\
\text{A7:} & \delta \cdot X & \to & \delta \\
\text{Sum1:} & \Sigma(\lambda d : \mathsf{Data}.X) & \to & X \\
\text{Sum3:} & (\Sigma P) + (PD) & \to & (\Sigma P) \\
\text{Sum4:} & \Sigma(\lambda d : \mathsf{Data}.((Pd) + (Qd))) & \to & (\Sigma P) + (\Sigma Q) \\
\text{Sum5:} & (\Sigma P) \cdot X & \to & \Sigma(\lambda d : \mathsf{Data}.((Pd) \cdot X))
\end{array}$$

To prove termination of this system we interpret both base types Data and Proc by $\mathbb{N}_{\geq 1}$, with the usual ordering. This is a domain, because it is well-founded, non-empty and there exists a binary strict function, ordinary "+" for instance. The function symbols are interpreted in the following way:

$$\begin{array}{rcl}
[\![+]\!] & = & \boldsymbol{\lambda} a.\boldsymbol{\lambda} b.a + b + 1 \\
[\![\cdot]\!] & = & \boldsymbol{\lambda} a.\boldsymbol{\lambda} b.a \times b + a \\
[\![\delta]\!] & = & 1 \\
[\![\Sigma]\!] & = & \boldsymbol{\lambda} f.3 \times f(1) + 1
\end{array}$$

This is an extension of the interpretation in [AB91] for the Process Algebra part of the system. The first three functions are clearly strict. The last is also strict: Take $f \ {}_{\text{str}}\!> g$, then (because 1 is strict) $f(1) \ {}_{\text{mon}}\!> g(1)$. But then also $3 \times f(1) + 1 \ {}_{\text{mon}}\!> 3 \times g(1) + 1$. Now we compute the values of the left hand sides and right hand sides.

	interpretation of the left hand side	interpretation of the right hand side
A3	$2 \times [\![X]\!] + 1$	$[\![X]\!]$
A4	$([\![X]\!] + [\![Y]\!] + 1) \times [\![Z]\!] + [\![X]\!] + [\![Y]\!] + 1$	$[\![X]\!] \times [\![Z]\!] + [\![X]\!] + [\![Y]\!] \times [\![Z]\!] + [\![Y]\!] + 1$
A5	$[\![X]\!] \times [\![Y]\!] \times [\![Z]\!] + [\![X]\!] \times [\![Z]\!] + [\![X]\!] \times [\![Y]\!] + [\![X]\!]$	$[\![X]\!] \times [\![Y]\!] \times [\![Z]\!] + [\![X]\!] \times [\![Y]\!] + [\![X]\!]$
A6	$[\![X]\!] + 2$	$[\![X]\!]$
A7	$[\![X]\!] + 1$	1
Sum1	$3 \times [\![X]\!] + 1$	$[\![X]\!]$
Sum3	$3 \times [\![P]\!](1) + [\![PD]\!] + 2$	$3 \times [\![P]\!](1) + 1$
Sum4	$3 \times ([\![P]\!](1) + [\![Q]\!](1)]) + 4$	$3 \times ([\![P]\!](1) + [\![Q]\!](1)]) + 3$
Sum5	$3 \times [\![P]\!](1) \times [\![X]\!] + [\![X]\!] + 3 \times [\![P]\!](1) + 1$	$3 \times [\![P]\!](1) \times [\![X]\!] + 3 \times [\![P]\!](1) + 1$

The reader can verify that the interpretation of the left hand side is greater than the interpretation on the right hand side on each line in the table. So this system of Process Algebra- and Sum rules is terminating.

7.2 Quantifier reasoning

In [Nip91] some HRSs concerning first order predicate logic are presented as an example. One of them is called *mini scoping*, pushing quantifiers inwards. The base types are $\{\mathsf{Term}, \mathsf{Form}\}$. The function symbols are:

$$\begin{array}{rcl}
\vee, \wedge & : & \mathsf{Form} \to \mathsf{Form} \to \mathsf{Form} \\
\forall & : & (\mathsf{Term} \to \mathsf{Form}) \to \mathsf{Form}
\end{array}$$

The rules (with free variables $\{P, P', Q, Q'\}$) are:

$$\begin{array}{rcl}
\forall(\lambda x.P) & \to & P \\
\forall(\lambda x.((P'x) \wedge (Q'x))) & \to & (\forall P') \wedge (\forall Q') \\
\forall(\lambda x.((P'x) \vee Q)) & \to & (\forall P') \vee Q \\
\forall(\lambda x.(P \vee (Q'x))) & \to & P \vee (\forall Q')
\end{array}$$

Again we take as interpretation for both base types the positive natural numbers $\mathbb{N}_{\geq 1}$. The interpretation of the function symbols is as follows:

$$[\![\wedge]\!] = [\![\vee]\!] = \boldsymbol{\lambda} a.\boldsymbol{\lambda} b.a + b + 1$$
$$[\![\forall]\!] = \boldsymbol{\lambda} f.2 \times f(1)$$

It is easily verified that under this interpretation the left hand side of each rule is greater than the interpretation of the right hand side.

7.3 Surjective Disjoint Union

Another application is given by the following example. The signature is given by the only base type (Form), the function symbols:

$$\begin{array}{rcl} case & : & \mathsf{Form} \rightarrow (\mathsf{Form} \rightarrow \mathsf{Form}) \rightarrow (\mathsf{Form} \rightarrow \mathsf{Form}) \rightarrow \mathsf{Form} \\ inl, inr & : & \mathsf{Form} \rightarrow \mathsf{Form} \end{array}$$

and the free variables $\{X, F, G, U\}$. The rules are:

$$\begin{array}{rcl} case(inl(X), F, G) & \rightarrow & F(X) \\ case(inr(X), F, G) & \rightarrow & G(X) \\ case(U, \lambda x.F(inl(x)), \lambda x.F(inr(x))) & \rightarrow & F(U) \end{array}$$

Note that this example does not fit in the framework of [Nip91] (see page 347). Termination for this example is less trivial, because there is a real application in the interpretation of the function symbols. Furthermore it is not the case that the number of "case" occurrences decreases in every step: If X contains a "case" occurrence, then F can generate many copies of it in the right hand side of the first rule.
But the interpretation in a Termination Model is easy: Take $\mathcal{I}(\mathsf{Form}) = \mathbb{N}_{\geq 1}$. Furthermore, interpret:

$$[\![case]\!] = \boldsymbol{\lambda} a.\boldsymbol{\lambda} f.\boldsymbol{\lambda} g.f(a) + g(a) + a$$
$$[\![inl]\!] = [\![inr]\!] = \boldsymbol{\lambda} a.a + 1$$

These functions are strict: we only need to take into account monotonic f and g functionals. Take for example $a \ {}_{\mathsf{str}}{>} \ b$. On base types this is the same as $a \ {}_{\mathsf{mon}}{>} \ b$. So by monotonicity of f we have $f(a) \ {}_{\mathsf{mon}}{\geq} \ f(b)$ and the same holds for g. But then $f(a) + g(a) + a \ {}_{\mathsf{mon}}{>} \ f(b) + g(b) + b$. Furthermore, the interpretations of the left- and right hand sides can be computed:

Left hand side	Right hand side
$[\![F]\!]([\![X]\!] + 1) + [\![G]\!]([\![X]\!] + 1) + [\![X]\!] + 1$	$[\![F]\!]([\![X]\!])$
$[\![F]\!]([\![X]\!] + 1) + [\![G]\!]([\![X]\!] + 1) + [\![X]\!] + 1$	$[\![G]\!]([\![X]\!])$
$2 \times [\![F]\!]([\![U]\!] + 1) + [\![U]\!]$	$[\![F]\!]([\![U]\!])$

The left hand sides are all greater than the right hand sides, because we may restrict to monotonic functionals for F and G. So the system of "surjective disjoint union" is terminating.

References

[AB91] G.J. Akkerman and J.C.M. Baeten. Term rewriting analysis in process algebra. Technical Report CS-R9130, Centre for Mathematics and Computer Science, June 1991.

[Bar84] H.P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics.* North-Holland, Amsterdam, second, revised edition, 1984.

[Bre88] Val Breazu-Tannen. Combining algebra and higher-order types. In *Proceedings 3^{th} Annual Symposium on Logic in Computer Science,* Edinburgh, pages 82–90, July 1988.

[dV87] R. de Vrijer. *Surjective Pairing and Strong Normalization: Two Themes in Lambda Calculus.* PhD thesis, University of Amsterdam, 1987.

[Gan80] R. Gandy. Proofs of strong normalization. In J.R. Hindley and J.P. Seldin, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 457–477. Academic Press, 1980.

[GP90] J.F. Groote and A. Ponse. The syntax and semantics of μCRL. Report CS-R9076, CWI, Amsterdam, 1990.

[Klo80] J.W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts.* Mathematisch Centrum, Amsterdam, 1980.

[Nip91] T. Nipkow. Higher-order critical pairs. In *Proceedings 6^{th} Annual Symposium on Logic in Computer Science,* Amsterdam, The Netherlands, pages 342–349, 1991.

[Nip93] T. Nipkow. Orthogonal higher-order rewrite systems are confluent. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 306–317. Springer-Verlag, 1993.

[Pau90] L. C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press Limited, London, 1990. In: The APIC-series 31.

[PM93] C. Paulin-Mohring. Inductive definitions in the system Coq. Rules and properties. In M. Bezem and J.F. Groote, editors, *Proceedings of the 1^{st} International Conference on Typed Lambda Calculi and Applications, TLCA '93,* Utrecht, The Netherlands, volume 664 of *Lecture Notes in Computer Science*, pages 328–345. Springer-Verlag, 1993.

[Sel93] M.P.A. Sellink. Verifying process algebra proofs in type theory. Technical Report Logic Group Preprint Series No. 87, Utrecht University, 1993.

[vOvR93] Vincent van Oostrom and Femke van Raamsdonk. Comparing combinatory reduction systems and higher-order rewrite systems. To appear, 1993.

[Wol93] D.A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge tracts in theoretical computer science.* Cambridge University Press, Cambridge, 1993.

[Zan93] H. Zantema. Termination of term rewriting by interpretation. In M. Rusinowitch and J.L. Rémy, editors, *Conditional Term Rewriting Systems, proceedings third international workshop CTRS-92*, volume 656 of *Lecture Notes in Computer Science*, pages 155–167. Springer-Verlag, 1993. Full version appeared as report RUU-CS-92-14, Utrecht University.