Standardization and Evaluation in Combinatory Reduction Systems

(Working Paper)

J. B. Wells*
Heriot-Watt University
http://www.cee.hw.ac.uk/~jbw/

Robert Muller†
Boston College
http://www.cs.bc.edu/~muller/

August 30, 2000

Abstract

A rewrite system has *standardization* iff for any rewrite sequence there is an equivalent one which contracts the redexes in a *standard* order. Standardization is extremely useful for finding normalizing strategies and proving that a rewrite system for a programming language is sound with respect to the language's operational semantics.

Although for some rewrite systems the standard-order can be simple, e.g., left-to-right or outermost-first, many systems need a more delicate order. There are abstract notions of standard order which always apply, but proofs (often quite difficult) are required that the rewrite system satisfies a number of axioms and not much guidance is provided for finding a concrete order that satisfies the abstract definition.

This paper gives a framework based on combinatory reduction systems (CRS's) which is general enough to handle many programming languages. If the CRS is orthogonal and fully extended and a *good* redex ordering can be found, then a standard order is obtained together with the standardization theorem. If the CRS also satisfies further criteria, then a good redex ordering is mechanically obtained from the rewrite rules. If the CRS is a constructor system and satisfies an additional requirement, then definitions of *value* and *evaluation* providing an operational semantics are automatically obtained together with a Plotkin/Wadsworth/Felleisen-style standardization theorem.

1 Introduction

1.1 Motivation

The motivations for this paper are as follows.

1. In the design of a programming language, it is desired that the operational semantics (a.k.a. the evaluation relation) of the language is consistent with an associated rewriting system (calculus) that is used to reason about program equivalence [Plo75]. This situation may arise in either of two ways.
   (a) One may start with a rewriting system and devise an operational semantics with the intention that the operational semantics is maximally defined w.r.t. the possibilities allowed by the rewriting system, i.e., the operational semantics is as complete as possible.
   (b) One may start with an operational semantics and devise a rewriting system hoping that the rewrite system equates as many programs as possible without equating operationally distinct programs, i.e., the rewriting system is sound.

*This author's work was partially supported by NSF grant CCR–9417382 and EPSRC grant GR/L 36963. Part of this author's work was done while visiting Boston University and part while employed at the University of Glasgow.

†Corresponding author. Voice: +1 617 552 3964. Fax: +1 617 552 2097. E-mail: muller@cs.bc.edu. This author's work was partially supported by NSF grant EIA–9806746 and by a Faculty Fellowship from the Wallace E. Carroll School of Management.

In either case, one is faced with the task of verifying the consistency between the operational semantics and the rewriting system. A proof of standardization is one way to do this.

2. The existing theoretical tools for the above task have proven difficult to use. In practice, standardization proofs have been carefully hand-crafted for equational calculi intended for program reasoning [FF89, FH92, Mul92, AF97]. When proving standardization for a new language, it is a laborious task to adapt an existing proof to the new language. There are some general-purpose tools from the rewriting community, but the programming language theorist has problems using them for various reasons:

   (a) Some methods are too weak, e.g., the method of Klop only handles left-normal rule sets [Klo80]. Any method which only supports the traditional "left-to-right" order will usually be too weak.

   (b) Other methods are too abstract. There are very general syntax-free frameworks [GLM92, KG96], but they do not reduce the burden significantly in comparison with the hand-crafted approach because the programming language theorist still has to prove their system satisfies the various axioms, which is very tedious to do correctly.

   What seems to be needed is a framework that is abstract enough to handle a variety of languages, but is concrete enough that the programming language theorist can embed their language in it without too much difficulty.

1.2 Contributions of this Paper

This paper contributes a framework for proving standardization for programming language calculi based on higher-order rewriting techniques, specifically, combinatory reduction systems (CRS's). The work is significant in the following ways:

1. The framework applies to any calculus where the left-hand sides of rewrite rules are such that when partial matches for more than one left-hand side exist there is always a position to check for the rest of the match which is in common among all of the partial matches. Programming language calculi generally seem to have this property. The framework avoids any "left-to-right" bias.

2. For a calculus to which the framework applies, we prove the standardization property. We carry out all of the necessary underlying work for this, proving all of the necessary properties of CRS's.

3. For CRS's that typically arise in programming language semantics, the *constructor systems*, we show how to automatically derive the sets of *values* and *evaluation contexts* directly from the rules of the CRS. These in turn determine an evaluation relation for the CRS.

4. We demonstrate that the methods developed in this paper are convenient to use by proving standardization theorems for two call-by-value λ-calculi, thereby showing for each the consistency of its operational semantics with its rewriting system. A particular contribution here is demonstrating the technical details of encoding a programming language calculus as a CRS.

1.3 Related Work

Standardization of a rewrite system is the property that for any rewrite sequence, there is a permutation-equivalent one which contracts the redexes in a nice order which is called "standard". One of the important properties of standard rewriting is that it provides a normalizing rewrite strategy. Standardization was first shown by Curry and Feys [CF58].

There are a variety of methods of proving standardization. Two important methods were devised by Klop, (1) identifying the "leftmost" (most needed) redex contracted in a rewrite sequence and performing it first and (2) replacing anti-standard pairs with standard complete developments [Klo80]. We use Klop's second method. Huet and Lévy define standard reductions for orthogonal TRS's as outside-in reductions, using a leftmost choice function to determine a unique standard reduction for a permutation equivalence class [HL91a]. Their proof of termination of the standardization algorithm depends on the disjointness of residuals through arbitrary rewrite sequences, a property of OTRS's but not of HORS's. Gonthier, Lévy,

and Melliès proved a standardization theorem for abstract rewriting [GLM92]. Melliès has done further work on abstract standardization [Mel98].

Other research into standardization includes the following. Jim and Meyer use a combination of Klop's first and second method to prove standardization for a variant of PCF and then use that result to prove the context lemma [JM96]. Suzuki used Klop's second method to prove standardization for conditional term rewriting systems [Suz96]. There is some more discussion of Klop's second method in [vO96]. Khasidashvili and Glauert proved something they call abstract standardization [KG96] and what they call relative standardization [GK]. Standardization has been used extensively for validating the consistency of an operational semantics with a calculus by Plotkin, Felleisen, Ariola, Friedman, Hieb, Muller, and others not listed [Plo75, FF89, FH92, Mul92, AF97]. The method of Ariola and Felleisen depends on disjoint redexes having disjoint residuals.

Higher-order term rewriting has been presented in a number of different formalisms, including several variations on the format of CRS's [Klo80, Nip91, KvO95, Ken89, vR96, vO94, KvOvR93, Kha90, Tak93, Wol93].

Because one of the aims of standardization is finding normalizing rewriting strategies, much work on normalization is related. Huet and Lévy devised the idea of needed redexes, those which must be contracted in any rewrite sequence to normal form [HL91a, HL91b]. To aid in finding needed redexes, they devised the notions of sequentiality and strong sequentiality. Klop and Middeldorp provide a quite readable discussion of strong sequentiality [KM91].

Barendregt, Kennaway, Klop, and Sleep raised the idea of needed redexes to the λ-calculus [BKKS87]. Glauert, Khasidashvili, Nöcker, and Middeldorp have all written about "normalization" to sets of terms that are not exactly normal forms [GK, Nöc94, Mid97]. Van Raamsdonk showed that the outermost-fair (multistep) strategy is normalizing for some HORS's [vR96]. Kennaway, Antoy, and Middeldorp have devised 1-step normalizing rewrite strategies for "non-sequential" systems [Ken89, AM96]. Sekar and Ramakrishnan have another approach to normalization [SR93].

1.4 Overview

Section 3 defines combinatory reduction systems (CRSs). Section 4 defines labelling for CRSs, from which we obtain descendants, residuals, developments, and Hyland/Wadsworth labelling for ensuring termination. Section 5 introduces good redex ordering functions which define standard reduction and yield normalizing reduction strategies. Section 6 shows how to obtain good redex ordering functions from top-down, no-lookahead, redex-directed subterm ordering functions. Section 7 shows how to automatically obtain sets of *values* and *evaluation contexts* directly from the reduction rules for the special case ofconstructor system CRSs. It also presents a specialized form of the standardization theorem that is more relevant to programming language semantics than is the classical theorem of Curry and Feys. In a companion paper [MW00], the results developed in this paper are applied to two programming calculi.

2 Mathematical Preliminaries

Most of the notation presented in this subsection is quite standard and is given here only to avoid ambiguities over minor differences in usage. However, some of the notation here is new.

Abbreviations for Sequences The notation $\vec{a}$ abbreviates the notation $a_1, a_2, \cdots$ where the number of items is unspecified or clear from the context. The notation $\vec{a}^n$ abbreviates the notation $a_1, \ldots, a_n$.

Binary Relations A *binary relation* $\Re$ is any set of pairs. Let $\Re$ range over binary relations. The statements $\Re(a, b)$ and $a \ \Re \ b$ mean the same as $(x, y) \in \Re$ and the expression $\Re^{-1}$ denotes the relation $\{ (b, a) \mid (a, b) \in \Re \}$. A relation $\Re$ is *transitive* iff $\Re(a, b)$ and $\Re(b, c)$ implies $\Re(a, c)$. A relation $\Re$ is *antisymmetric* iff $\Re(a, b)$ and $\Re(b, a)$ implies $a = b$. A relation $\Re$ is *reflexive* on some set S iff $\Re(a, a)$ for every $a \in S$. A relation $\Re$ is *irreflexive* iff $\Re(a, b)$ implies $a \neq b$. A relation $\Re$ is *well founded* iff there is no infinite

sequence a_1, a_2, a_3, ..., such that $\Re(a_i, a_{i+1})$ for all $i \geq 1$.[1] A relation $\Re$ is *finitely branching* iff $\{ b \mid \Re(a, b) \}$ is finite for all a.

Functions A *function* f is a binary relation such that if $(a, b) \in f$ and $(a, c) \in f$ then $b = c$. In this case, we write the pairs in f in the form $(a \mapsto b)$. Given a function f and a value a, if a value b exists such that $(a \mapsto b) \in f$, then $f(a)$ denotes the value b, else $f(a)$ is undefined. The *domain of definition* of a function f is $\mathrm{DomDef}(f) = \{ a \mid (a \mapsto b) \in f \}$ and the *range* of f is $\mathrm{Ran}(f) = \{ b \mid (a \mapsto b) \in f \}$. The assertion $f : S_1 \to S_2$ holds whenever $\mathrm{DomDef}(f) \subseteq S_1$ and $\mathrm{Ran}(f) \subseteq S_2$, in which case we can call S_1 and S_2 respectively a *domain* and a *codomain* of f. A function f is *total* w.r.t. a domain S iff $\mathrm{DomDef}(f) = S$. Given set S, if $S \subseteq \mathrm{DomDef}(f)$, then $f(S) = \{ f(a) \mid a \in S \}$, otherwise $f(S)$ is undefined. The *restriction* of a function f to a set S is the new function $f \downarrow S = \{ (a \mapsto b) \mid (a \mapsto b) \in f, a \in S \}$. The *composition* of functions f and g, notation $f \circ g$, is the function such that $(f \circ g)(x) = f(g(x))$. The *n-fold composition* of a function f is the function f^n such that $f^n(a) = f(f^{n-1}(a))$ if $n > 0$ and $f^0(a) = a$. The expression $f[a \mapsto b]$ denotes the function $(f \setminus \{ (a \mapsto c) \mid f(a) = c \}) \cup \{(a \mapsto b)\}$.

Orders An *order* O is a transitive and anti-symmetric binary relation. An order O is a *partial order* on set S iff O is reflexive on S. An order O is a *total order* on set S iff O is a partial order on S and $O(a, b)$ or $O(b, a)$ for every $a, b \in S$. When we use a symbol for a partial order like $\leq$ or $\preceq$ or $\trianglelefteq$, possibly superscripted or subscripted, the removal of the bottom line, e.g., $<$, flipping the symbol, e.g., $\geq$, and slashing the symbol, e.g., $\not\leq$, have the usual meaning. The set of minimal elements of a set S w.r.t. an order O is $\min_O S = \{ a \mid a \in S, \nexists b \in S.\, O(b, a) \text{ and } b \neq a \}$. If the context guarantees that $\min_O S = \{a\}$ will be a singleton set, we may freely use $\min_O S$ as the value a.

Strict Orders An order O is *strict* iff O is irreflexive. A *strict partial order* is any strict order. An order O is a *strict total order* on set S iff O is strict and $O(a, b)$, $O(b, a)$, or $a = b$ for every $a, b \in S$. (Following an unfortunate tradition, a strict partial order is not a partial order and a strict total order is not a total order.)

Sequences The expression $\langle a_1, a_2, \ldots \rangle$ is the sequence of a_1, a_2, ... treated as a single object. The expression $\langle\rangle$ denotes the 0-length sequence. Given a set S, the expression S^n denotes the set of sequences $\{ \langle \vec{a}^n \rangle \mid \{\vec{a}^n\} \subseteq S \}$. Given sequences $\chi_1 = \langle \vec{a}^n \rangle$ and $\chi_2 = \langle b_1, b_2, \ldots \rangle$ where the first sequence is finite, the expression $\chi_1 \cdot \chi_2$ (the concatenation of χ_1 and χ_2) denotes the sequence $\langle \vec{a}^n, b_1, b_2, \ldots \rangle$. Given a sequence χ, the expression $|\chi|$ evaluates to the number of elements in χ if χ is finite and evaluates to ω if χ is infinite. We consider only countable sequences. If a sequence $\langle a_1, a_2, \ldots \rangle$ is used in a context requiring a set, we treat it as the set $\{a_1, a_2, \ldots\}$ (where duplicates have the same effect as a single occurrence). We write $[a_1, a_2, \ldots]$ for a sequence that has no duplicates, i.e., where $a_i \neq a_j$ if $i \neq j$.

Relations and Orders from Sequences Given a sequence $\chi = \langle a_1, a_2, \ldots \rangle$, the statement $a_i \prec_\chi a_j$ holds iff $i < j$. If χ is a sequence with no duplicates, i.e., it can be written as $\chi = [a_1, a_2, \ldots]$, then $\prec_\chi$ is a strict total order on the set of elements of χ. In this case, we may refer to the sequence χ as a strict total order.

Lexicographic Order If O is a strict order, then its *lexicographic extension* O_{lex} is the strict order on sequences such that $O_{\mathrm{lex}}(\chi_1, \chi_2)$ iff there exists some common prefix χ such that $\chi_1 = \chi \cdot \chi_1'$, $\chi_2 = \chi \cdot \chi_2'$, and either $\chi_1' = \langle\rangle \neq \chi_2'$ or $\chi_1' = \langle a, \ldots \rangle$, $\chi_2' = \langle b, \ldots \rangle$, and $O(a, b)$. Observe that if O is a strict total order, then O_{lex} is a strict total order. Let $\min_{\mathrm{lex}}$ stand for $\min_{<_{\mathrm{lex}}}$.

Naming Conventions When using symbols to range over various kinds of entities, we follow the following conventions:

[1] Ideally, this notion would say that $\Re$ has no infinite *descending* chains. But which direction is descending, to the left or to the right? This differs between relations, e.g., $<$ descends to the left and $>$ descends to the right. Because associating a direction for "descent" with each relation seems too painful, we follow Baader and Nipkow [BN98, p. 14] in stating that descent is to the right. Some others have chosen that descent is to the left, e.g., Taylor [Tay99, p. 97].

a, b, c, X	arbitrary entity	c	label combining function
d	label decrementing function	f	function
i, j, k, l, m, n	natural number	p, q	path
r	reduction rule	s, t	metaterm
u, v	term	w	preterm
x, y, z	term variable	C	context
E	evaluation context	F, G, H, I	function symbol
F	piece of evaluation context	L, M	label set
M, N	term of λ_{v} or λ^{CIL}	O	order
P	label predicate	P	set of paths
R	set of reduction rules	R	renaming of metavariables
S	set	V	value (restricted term of λ_{v} or λ^{CIL})
Z	CRS metavariable	$\mathcal{L}$	labelling scheme
$\mathcal{R}$	redex ordering function	$\Re$	binary relation
α, β	label	γ	subterm ordering
δ	subterm ordering or "wrong"	χ	sequence
θ	labelling	ν	valuation
ρ	redex ordering	σ	reduction sequence
Γ	subterm ordering function	Δ	redex occurrence
Σ	combinatory reduction system		

3 Combinatory Reduction Systems

This section defines *combinatory reduction systems* (CRSs). We use the *functional* presentation of combinatory reduction systems [KvOvR93] rather than the original *applicative* presentation [Klo80]. Both ways of presenting CRSs have the same expressiveness. They differ in minor ways such as the number of "garbage terms" that must be ignored and how to determine the head symbol of a redex. We find it much easier to rigorously prove theorems using the functional CRS presentation.

For those familiar with CRSs, the non-standard bits are (1) the definition of the tree of a term, (2) the definition of subterm occurrence, (3) the definition of valuation, (4) the restriction requiring reduction rules to be fully extended, and (5) some notation for reduction sequences.

3.1 Basic CRS Definitions

Alphabet The alphabet used in constructing the preterms of a CRS consists of the following:

1. The countably infinite set Fun of *function symbols*. Let F, G range over Fun.
2. The countably infinite set Var of (ordinary) *variables*. Let x, y, z range over Var.
3. The countably infinite set MVar of *metavariables*. Let Z range over MVar.
4. The symbols "(", ")", "[", "]", "$\Box$" and ",".

Each function symbol F or metavariable Z has a fixed arity, written $\mathrm{Arity}(F)$ or $\mathrm{Arity}(Z)$. There are an infinite number of functions symbols and metavariables of each arity. The arity will often be indicated by writing $F^{(n)}$ or $Z^{(n)}$ in a statement, which is equivalent to writing merely F or Z and adding the side condition that $\mathrm{Arity}(F) = n$ or $\mathrm{Arity}(Z) = n$. The arity will usually be obvious. Ordinary variables are considered to have arity 0. Let $<^{\mathrm{mv}}$ be some fixed strict total order on MVar such that $>^{\mathrm{mv}}$ is well founded.

Preterms The set PTer of *preterms* is the smallest set satisfying the following conditions. Let w range over preterms.

1. If $x \in \mathrm{Var}$, then $x \in \mathrm{PTer}$.
2. If $x \in \mathrm{Var}$ and $w \in \mathrm{PTer}$, then $[x]w \in \mathrm{PTer}$.

3. If $F^{(n)} \in \text{Fun}$ and $\{\vec{w}^n\} \subset \text{PTer}$, then $F(\vec{w}^n) \in \text{PTer}$.

4. If $Z^{(n)} \in \text{MVar}$ and $\{\vec{w}^n\} \subset \text{PTer}$, then $Z(\vec{w}^n) \in \text{PTer}$.

5. $\Box \in \text{PTer}$.

All free occurrences of x in w are bound in $[x]w$. If $X^{(0)} \in \text{Fun} \cup \text{MVar}$, then the preterm $X()$ may (and usually will) be written as just X. Below, the sets of contexts, metaterms, and terms will be defined as subsets of PTer.

Paths The set $\mathcal{P}$ of *paths* is the set of sequences over $\mathbb{N}$ (the natural numbers). We generally write a path as $i_0 \cdot \ldots \cdot i_n$ rather than $\langle i_0, \ldots, i_n \rangle$. In a context requiring a path, the number i is implicitly coerced to the 1-length path $\langle i \rangle$. Let p, q range over paths, let P range over sets of paths, and let ϵ denote the 0-length path. The *prefix* partial order on paths is the order $\leq$ such that $p \leq q$ ("p is a prefix of q") iff there exists a path p' such that $q = p \cdot p'$. The statement $p \mid q$ ("p is incomparable with q") means $p \not\leq q$ and $q \not\leq p$.

Tree of a Preterm The *tree* of a preterm w is an alternate representation of the essential information in w. The function Tree is the least-defined partial function from PTer to $\mathcal{P}$ to $\text{Fun} \cup \text{Var} \cup \text{MVar} \cup \{\, [X] \mid X \in \text{Var} \cup \{\bullet\} \,\} \cup \mathcal{P} \cup \{\Box\}$ such that:

1. $\text{Tree}(x) = \{\epsilon \mapsto x\}$.

2. $$\text{Tree}([x]w)(p) = \begin{cases} [x] & \text{if } p = \epsilon \text{ and } \exists q.\ (\text{Tree}(w)(q) = \Box \text{ and } \nexists q' \leq q.\ \text{Tree}(w)(q') = [x]), \\ [\bullet] & \text{if } p = \epsilon \text{ and } \nexists q.\ (\text{Tree}(w)(q) = \Box \text{ and } \nexists q' \leq q.\ \text{Tree}(w)(q') = [x]), \\ \text{Tree}(w)(q) & \text{if } p = 1 \cdot q \text{ and } \text{Tree}(w)(q) \neq x, \\ p & \text{if } p = 1 \cdot q \text{ and } \text{Tree}(w)(q) = x. \end{cases}$$

3. For $X^{(n)} \in \text{Fun} \cup \text{MVar}$,
$$\text{Tree}\left(X(\vec{w}^n)\right)(p) = \begin{cases} X & \text{if } p = \epsilon, \\ \text{Tree}(w_i)(q) & \text{if } p = i \cdot q \text{ and } 1 \leq i \leq n. \end{cases}$$

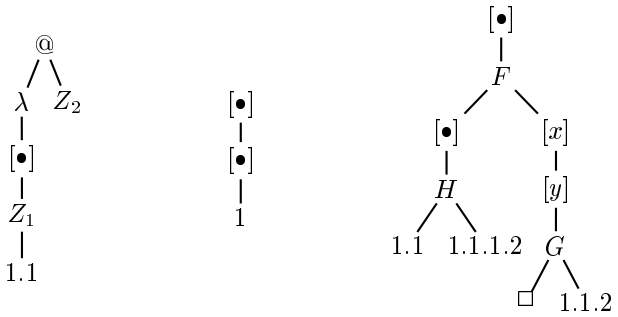
4. $\text{Tree}(\Box) = \{\epsilon \mapsto \Box\}$.

The *skeleton* of a preterm w is $\text{Skel}(w) = \text{DomDef}(\text{Tree}(w))$. A position p is *bound at q in w*, written $\text{BindPos}(w, p) = q$, iff $p = q \cdot p'$ and $\text{Tree}(w)(q) = [X]$ for some $X \in \text{Var} \cup \{\bullet\}$ and $\text{Tree}(w)(p) = p'$.

Here are three examples of the trees of preterms:

Preterm w: $\quad @(\lambda([x]Z_1(x)), Z_2) \qquad [x][y]y \qquad [x]F([z]H(z,x), [x][y]G(\Box, x))$

Tree(w):



A binding is recorded in the tree as "[•]" to ignore the name of the bound variable when its name is irrelevant. A binding is recorded as "[x]" only when there is at least one hole in the scope of the binding such that filling the hole by a preterm w with free variable x should result in the capture of the free variable by the binding. Bound variables are represented by the path from the internal node of the binder to the leaf node of the bound variable. De Bruijn indices or some similar scheme could have been used instead, but it turned out to be quite convenient to record the actual path from the binder to the variable.

Quotienting by α-Conversion The statement $w \equiv w'$ ("w and w' are *tree-equivalent*") means $\mathrm{Tree}(w) = \mathrm{Tree}(w')$. For preterms without holes, tree-equivalence corresponds exactly to the standard notion of α-conversion. For preterms with holes, for which α-conversion is not usually defined, tree-equivalence gives the best possible definition of α-conversion.

Convention 3.1. Throughout the rest of this article, tree-equivalent preterms are considered equal. □

In interpreting this convention, the reader can think of the set PTer as really being a set of trees of the form given above rather than as a set of syntactic entities. The "$\equiv$" symbol is used instead of "$=$" to avoid confusion with equality on the syntactic entities used to write preterms and equality w.r.t. an equational theory.

Metaterms, Terms, and Contexts The set MTer of *metaterms* is the subset of PTer containing all of the preterms which do not mention $\Box$. Let s and t range over metaterms.

The set Ter of all *terms* is the subset of MTer containing all of the metaterms which do not mention metavariables. Let u and v range over terms.

The set Ctxt of *contexts* is the subset of PTer containing all of the preterms which mention "$\Box$" (the hole). Let C range over contexts. Unless otherwise specified, a context is assumed to have exactly one hole. The arity of a context is the number of holes in the context. Given a context C, its arity n (respectively the position p of its unique hole if it has only one) may be indicated by writing $C^{(n)}$ (respectively C^p) in a statement, which is equivalent to writing merely C and adding the side condition that $\mathrm{Arity}(C) = n$ (respectively $\mathrm{Arity}(C) = 1$ and $\mathrm{Tree}(C)(p) = \Box$). If $\mathrm{Arity}(C) \neq n$, then $C[\vec{w}^n]$ is undefined, otherwise $C[\vec{w}^n]$ denotes the result of replacing all of the occurrences of $\Box$ in C by $w_1, \ldots, w_n$ in order from left-to-right, possibly capturing free variables of w.

Variables The set of metavariables occurring in a preterm w is $\mathrm{MV}(w) = \mathrm{Ran}(\mathrm{Tree}(w)) \cap \mathrm{MVar}$. The statement $\mathrm{DMV}(w, w')$ ("w and w' have disjoint metavariables") holds iff $\mathrm{MV}(w) \cap \mathrm{MV}(w') = \varnothing$. For a set of preterms S, the statement $\mathrm{DMV}(S)$ holds iff $\mathrm{DMV}(w, w')$ holds for every $w, w' \in S$ such that $w \not\equiv w'$. The set of (ordinary) variables occurring free in a preterm w is $\mathrm{FV}(w) = \mathrm{Ran}(\mathrm{Tree}(w)) \cap \mathrm{Var}$.

Subterms and Subterm Occurrences A preterm w' is a *subterm* of a preterm w, written $w' \trianglelefteq w$, iff there is a context C such that $w \equiv C[w']$. Quite different from a subterm, a *subterm occurrence* is specified by its position. If $p \in \mathrm{Skel}(w)$, then $w.p$ denotes the subterm occurrence in w at position p. We write $w_1.p_1 \equiv w_2.p_2$ to mean that $w_1 \equiv w_2$ and $p_1 = p_2$. When the context of discussion makes the preterm w obvious, p will sometimes stand for $w.p$.

REMARK 3.2. If $w \equiv C^p[w']$, some researchers will use w' to stand for subterm occurrence $w.p$. We avoid this because it can be ambiguous in two ways: (1) there may be another position $q \neq p$ and context C_1^q such that $w \equiv C_1^q[w']$ and (2) there may be another context C_2^p and metaterm w'' such that $w \equiv C_2^p[w'']$, where w' and w'' differ only in the names of free variables which are bound by C^p and C_2^p. □

Valuation A *valuation* is a function ν from $\mathrm{MVar} \cup \mathrm{Var}$ to MTer such that for each $X \in \mathrm{DomDef}(\nu)$ of arity n, the metaterm $\nu(X)$ mentions only metavariables in the distinguished set $\{\hat{Z}_1^{(0)}, \ldots, \hat{Z}_n^{(0)}\}$. (These distinguished metavariables play the part of the parameters of *substitutes* as used in other formulations of CRSs [Kah94].) Given valuation ν, let $\mathrm{FV}(\nu) = \mathrm{DomDef}(\nu) \cup \bigcup_{X \in \mathrm{DomDef}(\nu)} \mathrm{FV}(\nu(X))$. A *valuation for metaterm* s is a valuation ν such that $\mathrm{DomDef}(\nu) \supseteq \mathrm{MV}(s)$. The application $\nu(s)$ of a valuation to a metaterm is the term $\nu'(s)$ where ν' is the function from MTer to Ter defined as follows.

1. $\nu'(x) \equiv \begin{cases} x & \text{if } \nu(x) \text{ is undefined,} \\ \nu(x) & \text{otherwise.} \end{cases}$
2. $\nu'([x]t) \equiv [x']\nu'(t')$ where $[x]t \equiv [x']t'$ and $x' \notin \mathrm{FV}(\nu)$. We can always find such an x' and t' by α-conversion and it does not matter which ones we use.
3. $\nu'(F(\vec{s}^n)) \equiv F(\nu'(s_1), \ldots, \nu'(s_n))$.

4. $\nu'(Z(\vec{s}^n)) \equiv \nu''(\nu(Z))$ where $\nu'' = \{\, \hat{Z}_i^{(0)} \mapsto \nu'(s_i) \mid 1 \leq i \leq n \,\}$.

For compatibility with traditional substitution notation, let $t[X_1{:=}s_1, \ldots, X_n{:=}s_n]$ stand for $\nu(t)$ where $\nu = \{X_1 \mapsto s_1, \ldots, X_n \mapsto s_n\}$.

REMARK 3.3. Our definition of valuation differs from earlier definitions (e.g., [Klo80] and [KvOvR93]) in that (1) it uses a different (but equivalent) mechanism to handle the replacement of metavariables of arity greater than 0 and (2) it also allows replacing ordinary variables because it is needed in more situations. □

Reduction Rule A *pattern* is a metaterm s such that any metavariable $Z^{(n)}$ occurs in s only in the form $Z(\vec{x}^n)$ where $x_1, \ldots, x_n$ are n distinct (ordinary) variables. A *reduction rule* r is a pair $s \to t$ of metaterms satisfying the following conditions.

1. The metaterm s is a pattern.
2. The metaterm s is of the form $F(\vec{s}^n)$ for some function symbol F.
3. Both s and t are closed, i.e., in s and t each (ordinary) variable x occurs in the scope of a matching binder $[x]$.
4. Any metavariable which occurs in t also occurs in s.

For $r = s \to t$, let $\mathrm{LHS}(r) \equiv s$ (the *left-hand side*) and $\mathrm{RHS}(r) \equiv t$ (the *right-hand side*).

A position p is *internal* in pattern s iff p is the position in s of a function symbol, a binder, or an ordinary variable which is not a child of a metavariable. Formally, this is written $p \in \mathrm{Int}(s)$ and defined as follows.

$$\begin{aligned}\mathrm{Int}(s) = \quad & \{\, p \mid \mathrm{Tree}(s)(p) \in \mathrm{Fun} \cup \{[\bullet]\} \,\} \\ \cup\ & \{\, p \mid \mathrm{Tree}(s)(p) \in \mathcal{P} \cup \mathrm{Var}, (p = \epsilon \text{ or } (p = q \cdot i \text{ and } \mathrm{Tree}(s)(q) \notin \mathrm{MVar})) \,\}\end{aligned}$$

This notion is extended to reduction rules so that for rule $r = s \to t$ it holds that $\mathrm{Int}(r) = \mathrm{Int}(s)$ and position p is *r-internal* iff p is internal in s.

Rules Differing Only by Metavariable Names Two metaterms s_1 and s_2 are the *same up to metavariable renaming*, written $s_1 \simeq s_2$, iff s_2 is the result of renaming the metavariables in s_1 in a one-to-one manner. For metaterms that are the same up to metavariable renaming, we will define a strict total order to support the purpose of arbitrarily choosing one of them. Let $s_1 \blacktriangleleft s_2$ hold iff $s_1 \simeq s_2$ and $\langle \vec{Z} \rangle <_{\mathrm{lex}}^{\mathrm{mv}} \langle \vec{Z}' \rangle$ where $\langle \vec{Z} \rangle$ and $\langle \vec{Z}' \rangle$ are the sequences of metavariables occurring respectively in s_1 and s_2, in the order of occurrence from left to right.

We extend these notions to reduction rules as follows. Let $F^{(2)}$ be some fixed function symbol. Let $r = s \to t$ and $r' = s' \to t'$ be reduction rules. Then $r \simeq r'$ iff $F(s,t) \simeq F(s',t')$ and $r \blacktriangleleft r'$ iff $F(s,t) \blacktriangleleft F(s',t')$. Let $\mathrm{LeastUpToRenaming}(r) = \min_{\blacktriangleleft}\{\, r' \mid r \simeq r' \,\}$.

Reduction Relation Let r be a reduction rule $s \to t$, ν a valuation for s, C^p a term context, and u and v terms. If $u \equiv C^p[\nu(s)]$ and $v \equiv C^p[\nu(t)]$, then define the following.

1. The term $\nu(s)$ is an *r-redex term* and $\nu(t)$ is its *r-contractum term*.
2. The subterm/rule pair $\Delta = (u.p, r)$ is a *redex (occurrence)*.
3. The term u *reduces* to its *reduct* v by *contracting* Δ, written $u \xrightarrow{\Delta} v$.
4. The triple $u \xrightarrow{\Delta} v$, also written $\langle u, \Delta, v \rangle$, is a *reduction step*.
5. Equivalent variations indicating the rule r are $u \xrightarrow{\Delta}_r v$ and $u \xrightarrow{p}_r v$.

If R is a set of reduction rules, $u \xrightarrow{\Delta}_R v$ means $u \xrightarrow{\Delta}_r v$ for some $r \in R$. For X which is either a rule r or a rule set R, we write $u \longrightarrow_X v$ to mean $u \xrightarrow{\Delta}_X v$ for some unspecified Δ. The transitive, reflexive closure of $\longrightarrow_X$ is $\twoheadrightarrow_X$. A term u is in *normal form* w.r.t. rule set R, written $R\text{-nf}(u)$, iff there is no term v such that $u \longrightarrow_R v$. A term u has R-normal form v, written $u \xrightarrow{\mathrm{nf}}_R v$, iff $u \twoheadrightarrow_R v$ and $R\text{-nf}(v)$. If $\Delta = (u.p, r)$, $\Delta' = (u.q, r')$, and $\Re$ is a relation on paths, then $\Delta \,\Re\, \Delta'$ iff $p \,\Re\, q$. The equational theory of R, written $=_R$, is the least equivalence relation on Ter containing $\longrightarrow_R$.

Reduction Sequence A *reduction sequence* σ is an alternating sequence of terms and redex occurrences beginning with a term such that (1) every 3-element subsequence of the form $\langle u, \Delta, v\rangle$ is a valid reduction step and (2) σ either ends with a term or is infinite. We write $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow v$ to indicate both the initial and final term and $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow \cdots$ to indicate only the initial term. The length of a reduction sequence, written $|\sigma|$, is the number of reduction steps it contains (ω for infinite sequences). The expression $\sigma[i..j]$ for $0 \leq i \leq j \leq |\sigma|$ denotes the subsequence of σ starting with the ith term (where terms and steps are both numbered starting with 0) and ending with the jth term and $\sigma[i..]$ denotes the suffix starting with the ith term. Let $\sigma[i]$ be the ith term in σ. We write $\sigma \sim \sigma'$ iff σ and σ' are *coinitial* and *cofinal*, i.e., $\sigma[0] \equiv \sigma'[0]$ and $\sigma[|\sigma|] \equiv \sigma'[|\sigma'|]$. Given $\sigma = \langle u_0, \Delta_0, u_1, \Delta_1, u_2, \ldots\rangle$ where $\Delta_i = (u_i.p_i, r_i)$, we will sometimes write this as $\sigma = u_0 \xrightarrow{p_0}_{r_0} u_1 \xrightarrow{p_1}_{r_1} u_2 \cdots$. Given $\sigma = \langle \ldots, u\rangle$ and $\sigma' = \langle u, \ldots\rangle$, their concatenation $\sigma \star \sigma'$ is $\langle \ldots, u, \ldots\rangle$. Given rule set R, a reduction sequence σ is an R-reduction sequence iff the rule of every redex occurrence in σ belongs to R.

Combinatory Reduction System A *combinatory reduction system* (CRS) Σ is specified by a set of *terms* $\mathrm{Ter}(\Sigma)$ and a set of *reduction rules* $\mathrm{Red}(\Sigma)$ (sometimes called *rewrite rules*). The set of terms $\mathrm{Ter}(\Sigma)$ is some subset of Ter which must be closed under the reduction relation $\longrightarrow_{\mathrm{Red}(\Sigma)}$, i.e., if $u \longrightarrow_{\mathrm{Red}(\Sigma)} v$ and $u \in \mathrm{Ter}(\Sigma)$ then $v \in \mathrm{Ter}(\Sigma)$.[2] In contexts in which a set of rules is required, we may use Σ to stand for $\mathrm{Red}(\Sigma)$, but when forming a reduction relation we restrict it to terms in $\mathrm{Ter}(\Sigma)$, e.g., $\longrightarrow_\Sigma = \{\, (u, v) \mid u \longrightarrow_{\mathrm{Red}(\Sigma)} v \text{ and } u, v \in \mathrm{Ter}(\Sigma) \,\}$. The equational theory $=_\Sigma$ is similarly restricted to $\mathrm{Ter}(\Sigma)$. This is the *only* way the restriction to terms in $\mathrm{Ter}(\Sigma)$ affects the reduction relation.

Strong Normalization A term u is *strongly normalizing* (SN) w.r.t. a rule set R, written R-SN(u), iff there is no infinite R-reduction sequence σ with initial term u. A metaterm s such that $\mathrm{MV}(s) = \{Z_1^{(i_1)}, \ldots, Z_n^{(i_n)}\}$ is strongly R-normalizing, written R-SN(s), iff the statement R-SN($\nu(s)$) holds where $\nu = \{\, Z_j^{(i_j)} \mapsto F_j^{(i_j)}(\hat{Z}_1, \ldots, \hat{Z}_{i_j}) \mid 1 \leq j \leq n \,\}$ and $\vec{F}^n$ are fresh function symbols of appropriate arities. For a valuation ν it holds that Σ-SN(ν) iff R-SN(s) for every metaterm $s \in \mathrm{Ran}(\nu)$. A CRS Σ is strongly normalizing, written SN(Σ), iff Σ-SN(u) for every $u \in \mathrm{Ter}(\Sigma)$.

3.2 Restrictions on CRSs

Fully Extended A pattern s is *fully extended* [HP99, vR96, Def. 4.2.51] iff for any metavariable occurrence $Z(\vec{x})$ in s, the sequence $\vec{x}$ includes as many variables as possible, i.e., for each binding of some variable x whose scope includes the metavariable occurrence, x is one of $\vec{x}$. A reduction rule $s \to t$ is fully extended iff s is fully extended. A CRS Σ is *fully extended* iff each reduction rule $r \in \mathrm{Red}(\Sigma)$ is fully extended.

Convention 3.4. Throughout the rest of this article, reduction rules (and therefore also CRSs) are restricted to be fully extended. □

REMARK 3.5. Requiring reduction rules to be fully extended is non-standard. Without this requirement, contracting an erasing redex occurrence at position p can create a redex occurrence for rule r at position q even when q is above p and $p \neq q \cdot q'$ for any $q' \in \mathrm{Int}(r)$. For example, applying the rule $F(Z) \to G$ to the term $H([x]F(x))$ creates a redex of the non-fully-extended rule $H([x]Z) \to I$. Our proof for theorem 6.4, which extends a proof of Klop [Klo80], fails in the presence of non-fully-extended rules. This problem seems related to the fact that the outermost-fair (multistep) reduction strategy is not normalizing for higher-order rewriting systems with non-fully-extended rules [vR96, Chap. 6.2]. □

Constructor Systems A CRS Σ is a *constructor CRS* iff there exists a set $\mathrm{FCon}(\Sigma)$ of *constructors* and a set $\mathrm{FDes}(\Sigma)$ of *destructors*[3] such that both of the following conditions hold.

[2] Previous definitions of a CRS required giving for a CRS Σ some fixed set S of function symbols and using all the terms that can be generated with those function symbols, i.e., setting $\mathrm{Ter}(\Sigma) = \{\, u \mid u \in \mathrm{Ter}, \mathrm{Fun}(u) \subseteq S \,\}$ where $\mathrm{Fun}(u) = \mathrm{Ran}(\mathrm{Tree}(u)) \cap \mathrm{Fun}$. Further restricting the set of terms was called a *substructure* CRS by Klop [Klo80] and [KvOvR93]. Our definition makes this the default case, because in practice it is nearly always necessary to do so.

[3] These are sometimes called *functions*, but we avoid that terminology due to the ambiguity with "function symbol."

1. The constructors and destructors partition the function symbols, i.e., $\mathrm{FCon}(\Sigma) \cup \mathrm{FDes}(\Sigma) = \mathrm{Fun}$ and $\mathrm{FCon}(\Sigma) \cap \mathrm{FDes}(\Sigma) = \varnothing$.

2. For any rule $r \in \mathrm{Red}(\Sigma)$, the root function symbol of $\mathrm{LHS}(r)$ is a destructor and any other function symbol in $\mathrm{LHS}(r)$ is a constructor. Formally, if $r \in \mathrm{Red}(\Sigma)$ and $\mathrm{Tree}(\mathrm{LHS}(r))(p) = F$, then $p = \epsilon \Rightarrow F \in \mathrm{FDes}(\Sigma)$ and $p \neq \epsilon \Rightarrow F \in \mathrm{FCon}(\Sigma)$.

Left-Linear Rule A metaterm s is *linear* iff every metavariable in s occurs in s exactly once. A reduction rule $s \to t$ is *left-linear* iff s is linear. A set of reduction rules R is left-linear iff every reduction rule $r \in R$ is left-linear.

Ambiguous Rules Given metaterms s and t and position $p \in \mathrm{Int}(t)$, it is said that s *interferes with* t *at* p iff there exist valuations ν and ν' for respectively s and t and a term context C^p such that $C[\nu(s)] \equiv \nu'(t)$.[4] This interference is *at the root* iff $p = \epsilon$. Distinct reduction rules $s \to t$ and $s' \to t'$ are *ambiguous* (sometimes called *overlapping*) whenever s and s' interfere. A reduction rule $s \to t$ is ambiguous with itself whenever s interferes with itself provided the interference is not at the root. A set of reduction rules R is ambiguous iff there exists an ambiguous pair of rules $r, r' \in R$ (where r and r' may be the same rule).

Orthogonal CRS A CRS Σ is *orthogonal* (also called *regular*) iff the set of reduction rules $\mathrm{Red}(\Sigma)$ is left-linear and non-ambiguous.[5]

Theorem 3.6 (Confluence of Every Orthogonal CRS). *If Σ is orthogonal, $u \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v_1$, and $u \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v_2$, then there exists u' such that $v_1 \longrightarrow\!\!\!\!\!\rightarrow_\Sigma u'$ and $v_2 \longrightarrow\!\!\!\!\!\rightarrow_\Sigma u'$.* □

Proof. See [Klo80] or [KvOvR93]. □

4 Labelling

This section defines a generic notion of labelling which is instantiated to yield definitions of descendants, residuals, developments, and Hyland/Wadsworth labelling. The specific instantiations are quite similar to systems in [Klo80]. Labelling in this article differs from some earlier labelling as described in remark 4.1. The labelling of a CRS Σ is a CRS $\Sigma^{\mathcal{L}}$, and labelling preserves orthogonality, thus allowing certain proofs to be chained, e.g., the proof of lemma 5.6 implicitly uses three levels of labelling.

4.1 Generic Labelling Framework

Labelling Scheme A *labelling scheme* is given by a quadruple $\mathcal{L} = \langle L, c, d, P \rangle$ which contains:

1. A set of *labels* L. Let α and β range over L.

2. An associative label combination operator $c : (L \times L) \to L$.

3. A label decrementing operator $d : L \to L$.

4. A predicate on labels $P : L \to \{true, false\}$.

[4] This definition is equivalent to the definitions of [Klo80] and [KvOvR93] for metaterms which can be LHS's of reduction rules, but differs in how it is stated. Note that the definition of interference would be the same even without our restriction to fully extended reduction rules.

[5] The definition given here considers valuations producing terms that are outside of $\mathrm{Ter}(\Sigma)$ when determining the ambiguity of $\mathrm{Red}(\Sigma)$. The definition could be changed to avoid considering such garbage terms, but doing this without breaking something would make the definition very messy.

Preterm Labelling We take the label set L to be a subset of Fun containing 1-ary function symbols. Given a label sequence $\vec{\alpha}^n$ such that $\{\vec{\alpha}^n\} \subseteq L$, let $w^{\langle\vec{\alpha}\rangle} = \alpha_n(\cdots(\alpha_2(\alpha_1(w)))\cdots)$, sometimes written as $(\cdots(w^{\alpha_1})\cdots)^{\alpha_n}$. A preterm w is *unlabelled* w.r.t. L iff w does not mention any symbols in L. Given an unlabelled preterm w, an *L-labelling* (a.k.a. *$\mathcal{L}$-labelling*) θ for w is a total function from $\mathrm{Skel}(w)$ to $\bigcup_{k\in\mathbb{N}} L^k$. A labelling θ is *complete* iff $\langle\rangle \notin \mathrm{Ran}(\theta)$. The expression $\theta.q$ denotes the function such that $(\theta.q)(p) = \theta(q\cdot p)$. If θ is a labelling for w, then w^θ is defined as follows.

1. $X^\theta \equiv X^{\theta(\epsilon)}$ for $X \in \mathrm{Var} \cup \{\Box\}$.
2. $([x]w)^\theta \equiv ([x](w^{\theta.1}))^{\theta(\epsilon)}$.
3. $(X(\vec{w}^n))^\theta \equiv (X({w_1}^{\theta.1}, \ldots, {w_n}^{\theta.n}))^{\theta(\epsilon)}$ for $X \in (\mathrm{Fun} \cup \mathrm{MVar}) - L$.

Observe that labelling fails for already-labelled preterms. Observe also that labelling of metaterms, terms, and contexts produces respectively metaterms, terms, and contexts, i.e., the sets MTer, Ter, and Ctxt are closed under the operation of labelling.

Rule Labelling Let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme. A *LHS labelling* for a rule $r = s \to t$ or a pattern s is a labelling θ for s where $\theta(p) \neq \langle\rangle$ iff $p \in \mathrm{Int}(s)$ and $\theta(\epsilon) = \langle\alpha\rangle$ for some α (i.e., $\theta(\epsilon)$ is a sequence of length 1).

Given a LHS labelling θ for r, we construct the corresponding *RHS labelling* $\overline{\theta}$ as follows. Extend c to any arity one or greater by setting $c(\alpha) = \alpha$ and $c(\alpha, \alpha', \beta, \ldots) = c(c(\alpha, \alpha'), \beta, \ldots)$ and to sequences by $c(\langle\vec{\alpha}\rangle) = c(\vec{\alpha})$. Let the *degree* of θ be $\mathrm{Deg}(\theta) = c(c(\theta(p_1)), \ldots, c(\theta(p_n)))$ where $\{\vec{p}^n\} = \{\, p \in \mathrm{DomDef}(\theta) \mid |\theta(p)| \geq 1 \,\}$ and $p_1 <_{\mathrm{lex}} \cdots <_{\mathrm{lex}} p_n$. Finally, let $\overline{\theta} = \{\, p \mapsto \langle d(\mathrm{Deg}(\theta))\rangle \mid p \in \mathrm{Skel}(t) \,\}$. Observe that if α occurs in a RHS labelling, then $\alpha \in \mathrm{Ran}(d)$.

Given a LHS labelling θ for r and the corresponding RHS labelling $\overline{\theta}$, the labelling of the rule r is given as $r^\theta = s^\theta \to t^{\overline{\theta}}$. Observe that the result of rule labelling obeys the requirements for a rule, i.e., no free variables, function symbol at root of LHS, metavariables in RHS occur in LHS, and metavariables in LHS have only distinct variables as arguments. A rule $r = s \to t$ is *unlabelled* iff both s and t are unlabelled. Define the degree of r^θ to be $\mathrm{Deg}(r^\theta) = \mathrm{Deg}(\theta)$. For a redex occurrence $\Delta = (u.p, r^\theta)$ let $\mathrm{Deg}(\Delta) = \mathrm{Deg}(\theta)$. For redex terms let $\mathrm{Deg}(\nu(\mathrm{LHS}(r^\theta))) = \mathrm{Deg}(\theta)$ (assuming the rule can be unambiguously determined).

CRS Labelling The labelling of a CRS goes as follows. Let Σ be a CRS and let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme. Assume that the members of $\mathrm{Ter}(\Sigma)$ and $\mathrm{Red}(\Sigma)$ are unlabelled w.r.t. L, if necessary by renaming all labels (members of L) to fresh names throughout $\mathcal{L}$. Assume if necessary that there are an infinite number of function symbols unused by Σ. Then $\Sigma^{\mathcal{L}}$ is the entity where:

1. $\mathrm{Ter}(\Sigma^{\mathcal{L}}) = \{\, u^\theta \mid u \in \mathrm{Ter}(\Sigma)$ and θ is a complete labelling for $u \,\}$.
2. $\mathrm{Red}(\Sigma^{\mathcal{L}}) = \{\, r^\theta \mid r \in \mathrm{Red}(\Sigma)$, θ is a LHS labelling for r, and $P(\mathrm{Deg}(\theta)) \,\}$.

It will be shown in theorem 4.8 that $\Sigma^{\mathcal{L}}$ is in fact a CRS, i.e., its set of terms is closed under reduction.

Remark 4.1 (Comparison with other Approaches to Labelling). The approach to labelling used in this paper generalizes methods of Klop [Klo80]. Klop's various presentations of labelled CRS's were not unified in this manner, although all were in a style resembling the result of our method instantiated with particular labelling schemes.

Any method which labels positions by inserting additional function symbols must somehow deal with the accumulation of labels through reduction. In his specific labelled CRS's, Klop made the error of identifying $(u^\alpha)^\beta$ with $u^{c(\alpha,\beta)}$. This is an error because it either (1) contradicts the claim that $\Sigma^{\mathcal{L}}$ is a CRS or (2) requires extra reduction rules (which were not supplied) to reduce $(u^\alpha)^\beta$ to $u^{c(\alpha,\beta)}$. To avoid this error, we do not identify $(u^\alpha)^\beta$ with $u^{c(\alpha,\beta)}$. Some researchers (e.g., van Oostrom [vO97]) have avoided this error by adding the extra rules. Although this approach can work, we did not choose it because it prevents the one-to-one correspondence between reduction steps in Σ and $\Sigma^{\mathcal{L}}$ which is obtained below as theorem 4.6. Although our labelled systems treat $u^{\langle\vec{\alpha}\rangle}$ and $u^{c(\vec{\alpha})}$ in essentially the same way, the underlying term could be different.

A difference from earlier labelling approaches is that our approach works for both singleton-redex rules and collapsing rules. A *singleton-redex* rule is a rule $s \to t$ where s has exactly one function symbol and no bindings, e.g., $F(Z) \to Z$ or $G \to H$. If labels are fresh function symbols inserted *between* other function symbols, then the obvious simple labelling approach will not associate any label with a singleton redex, because it does not have two distinct symbols or binders to put a label between. The key point of our labelling method that handles singleton-redex rules is the requirement that every labelling of a rule r matches at least one label symbol above the root symbol of the LHS of r.

A rule $s \to t$ is *collapsing* iff t contains no function symbols or binders. An earlier labelling scheme by Kenneway [?] failed to work for collapsing rules. If a new redex occurrence Δ' is created by the contraction of a redex Δ of a collapsing rule, then it is essential that at least one label associated with Δ' is derived from the labels associated with Δ. In our approach, if a redex is created by a collapsing rule, then at least one of these two cases must hold:

1. Part of the redex must come from the surrounding context and part must come from the substitution for a metavariable Z in t.
2. Part of the redex must come from an occurrence of a metavariable Z_1 in t and part must come from a distinct occurrence of a metavariable Z_2 in t. (The two metavariables may be the same, but the occurrences must be different.)

In both cases it can be checked that our labelling scheme puts a label in a place such that any labelled redex created by the labelled rule must contain at least one copy of the label. □

4.2 Basic Properties of Labelling

Term/Labelling Decomposition For label set L and preterm w, if $w \equiv {w'}^{\theta}$ for L-labelling θ and unlabelled w', then (w', θ) is a *term/labelling decomposition* of w w.r.t. L. The following definitions support obtaining term/labelling decompositions. Let $q \cdot \theta$ denote the least defined function such that $(q \cdot \theta)(q \cdot p) = \theta(p)$. Let TLD be the least-defined function satisfying the following.

1. $\mathrm{TLD}_L(\alpha_n(\cdots(\alpha_1(X))\cdots)) = (X, \{\epsilon \mapsto \langle \vec{\alpha}^n \rangle\})$ for $X \in \mathrm{Var} \cup \{\Box\}$.
2. $\mathrm{TLD}_L(\alpha_n(\cdots(\alpha_1([x]w))\cdots)) = ([x]w', \{\epsilon \mapsto \langle \vec{\alpha}^n \rangle\} \cup 1 \cdot \theta)$ where $\mathrm{TLD}_L(w) = (w', \theta)$.
3. $\mathrm{TLD}_L(\alpha_n(\cdots(\alpha_1(X(\vec{w}^n)))\cdots)) = (X(\vec{w'}^n), \{\epsilon \mapsto \langle \vec{\alpha}^n \rangle\} \cup (\bigcup_{1 \le i \le n} i \cdot \theta_i))$ where $X \notin L$ and $\mathrm{TLD}_L(w_i) = (w_i', \theta_i)$ for $1 \le i \le n$.

Lemma 4.2 (Unique Term/Labelling Decomposition). *For any preterm w and label set L it holds that $\mathrm{TLD}_L(w) = (w', \theta)$ is the unique term/labelling decomposition of w w.r.t. L.* □

Proof. Induction on the size of w shows that $w \equiv {w'}^{\theta}$ where θ is an L-labelling iff $\mathrm{TLD}_L(w) = (w', \theta)$. This is sufficient to see that $\mathrm{TLD}_L(w)$ is defined, is a term/labelling decomposition of w w.r.t. L, and is the unique such decomposition. □

Label Erasure The *label erasure* of a preterm w w.r.t. the label set L of a labelling scheme $\mathcal{L}$, is $\|w\|_L \equiv \|w\|_{\mathcal{L}} \equiv w'$ (or $\|w\|$ if L is clear) where $\mathrm{TLD}_L(w) = (w', \theta)$. For a subterm occurrence $s.p$ where $s \equiv C^p[t]$, let $\|s.p\| \equiv \|s\|.q$ where $\|C^p\| \equiv C'^q$. For a valuation ν, define $\|\nu\|$ so that $\|\nu\|(X) \equiv \|\nu(X)\|$ for $X \in \mathrm{Var} \cup \mathrm{MVar}$. Label erasure is extended to other entities (e.g., reduction rules, redex occurrences, reduction sequences, etc.) componentwise.

Lemma 4.3 (Distribution of Label Erasure). *Given any label set L, metaterm s, context C, and valuation ν for s, both of the following statements hold:*

1. $\|C[s]\|_L \equiv \|C\|_L[\|s\|_L]$.
2. $\|\nu(s)\|_L \equiv \|\nu\|_L(\|s\|_L)$. □

Proof.

1. By induction on the size of C.

2. Let $\mathrm{MaxMVArity}(t) = \max\{0\} \cup \mathrm{Arity}(\mathrm{MV}(t))$. By induction on s in the order O such that $O(t_1, t_2)$ iff $\mathrm{MaxMVArity}(t_1) < \mathrm{MaxMVArity}(t_2)$ or $\mathrm{MaxMVArity}(t_1) = \mathrm{MaxMVArity}(t_2)$ and t_1 is smaller than t_2. By cases on the shape of s. Let $\nu' = \|\nu\|_L$.

   (a) Case $s \equiv x$. Then $\|s\|_L \equiv s$. By cases on whether $\nu(x)$ is defined.

      i. Case $\nu(x)$ is undefined. Then $\nu'(x)$ is also undefined. It can be checked that $\|\nu(s)\|_L \equiv \nu(s) \equiv s \equiv x \equiv \|s\|_L \equiv \nu'(\|s\|_L)$.

      ii. Case $\nu(x) \equiv u$. Then $\|\nu(s)\|_L \equiv \|u\|_L$. Also, $\|s\|_L \equiv x$. Thus, $\nu'(\|s\|_L) \equiv \nu'(x) \equiv \|u\|_L$. Case done.

   (b) Case $s \equiv [x]t$. By α-conversion assume that $x \notin \mathrm{FV}(\mathrm{Ran}(\nu)) \cup \mathrm{DomDef}(\nu)$. So $\nu(s) \equiv [x]\nu(t)$. So $\|\nu(s)\|_L \equiv [x]\|\nu(t)\|_L$. Also, $\|s\|_L \equiv [x]\|t\|_L$. So $\nu'(\|s\|_L) \equiv [x]\nu'(\|t\|_L)$. By induction hypothesis, $\|\nu(t)\|_L \equiv \nu'(\|t\|_L)$. This is sufficient to finish this case.

   (c) Case $s \equiv F(\vec{t})$ where $F \notin L$. By reasoning like the previous case, only simpler, because there is no need to consider α-conversion.

   (d) Case $s \equiv \alpha(t)$ where $\alpha \in L$. Then $\nu(s) \equiv \alpha(\nu(t))$. Then $\|\nu(s)\|_L \equiv \|\nu(t)\|_L$. Also, $\|s\|_L \equiv \|t\|_L$. So $\nu'(\|s\|_L) \equiv \nu'(\|t\|_L)$. By induction hypothesis, $\|\nu(t)\|_L \equiv \nu'(\|t\|_L)$. This is enough to finish this case.

   (e) Case $s \equiv Z(\vec{t}^n)$. By cases on $\mathrm{Arity}(Z)$.

      i. Case $n = 0$. Then $\nu(s) \equiv \nu(Z)$. Also, $\|s\|_L \equiv Z()$. Then $\|\nu(s)\|_L \equiv \|\nu(Z)\|_L \equiv \nu'(Z) \equiv \nu'(Z()) \equiv \nu'(\|s\|_L)$. Case done.

      ii. Case $n \geq 1$. Let $s' \equiv \nu(Z)$. Observe that $\mathrm{MaxMVArity}(s') = 0 < n \leq \mathrm{MaxMVArity}(s)$. By definition, $\nu(s) \equiv \nu''(s')$ where $\nu'' = \{\, \hat{Z}_i \mapsto \nu(t_i) \mid 1 \leq i \leq n \,\}$. Let $\nu''' = \|\nu''\|_L$. By induction hypothesis, $\|\nu''(s')\|_L \equiv \nu'''(\|s'\|_L)$. So $\|\nu(s)\|_L \equiv \nu'''(\|s'\|_L)$. It holds that $\|s\|_L \equiv Z(\|t_1\|_L, \ldots, \|t_n\|_L)$. Also, $\nu'(Z) \equiv \|s'\|_L$. So $\nu'(\|s\|_L) \equiv \nu''''(\|s'\|_L)$ where $\nu'''' = \{\, \hat{Z}_i \mapsto \nu'(\|t_i\|_L) \mid 1 \leq i \leq n \,\}$. By induction hypothesis, $\|\nu(t_i)\|_L \equiv \nu'(\|t_i\|_L)$ for $1 \leq i \leq n$. Thus, $\nu''' = \nu''''$. This is sufficient to finish the case. □

Subterm Occurrence Labelling For a subterm occurrence $u.p$ where θ is an $\mathcal{L}$-labelling for u and $\mathcal{L} = \langle L, c, d, P \rangle$, let $(u.p)^\theta \equiv u^\theta.q$ where $\|u^\theta.q\| \equiv u.p$ and $\mathrm{Tree}(u^\theta)(q) \notin L$.

Context/Subterm Labelling Decomposition Given a labelled preterm $C^p[w]^\theta$, it is desirable to be able to decompose the labelling θ into a portion applied to C and a separate portion applied to w. Given a labelling θ, a path p, and a natural number i, let $\theta_{\mathrm{c},p,i}$ and $\theta_{\mathrm{s},p,i}$ be the least-defined functions such that

$$\theta_{\mathrm{c},p,i}(q) = \begin{cases} \theta(q) & \text{if } p \not\leq q, \\ \langle \alpha_{i+1}, \ldots, \alpha_n \rangle & \text{if } p = q,\ \theta(p) = \langle \vec{\alpha}^n \rangle, \text{ and } 0 \leq i. \end{cases}$$

$$\theta_{\mathrm{s},p,i}(q) = \begin{cases} \theta(p \cdot q) & \text{if } q \neq \epsilon, \\ \langle \alpha_1, \ldots, \alpha_i \rangle & \text{if } q = \epsilon,\ \theta(p) = \langle \vec{\alpha}^n \rangle, \text{ and } 0 \leq i. \end{cases}$$

Lemma 4.4. *Let $w \equiv C^p[w']$ and let θ be an L-labelling for w. Then for any context $\hat{C}^q$ and preterm $\hat{w}$ where $\|\hat{C}\| \equiv C$ and $\|\hat{w}\| \equiv w$, the following equivalence holds:*

$$w^\theta \equiv \hat{C}[\hat{w}].$$

$$\Updownarrow$$

There exists an i such that $\hat{C} \equiv C^{\theta_{\mathrm{c},p,i}}$ and $\hat{w} \equiv w'^{\theta_{\mathrm{s},p,i}}$. □

Proof. By induction on the size of C. By cases on the shape of C.

1. Suppose $C \equiv \Box$. Then $p = \epsilon$ and $w \equiv w'$. Let $\theta(\epsilon) = \langle \vec{\alpha}^n \rangle$. Let $w'' = w^{\theta[\epsilon \mapsto \langle\rangle]}$. The two directions are proved separately.

($\Downarrow$) Suppose $w^\theta \equiv w'^\theta \equiv \hat{C}[\hat{w}]$. It holds that $w^\theta \equiv \alpha_n(\cdots(\alpha_1(w''))\cdots)$. Because $\|\hat{C}\| \equiv C \equiv \Box$, it must be the case that $\hat{C}$ is of the form $\beta_m(\cdots(\beta_1(\Box))\cdots)$ for some labels $\vec{\beta} \in L$. It can be shown that $\langle\vec{\beta}\rangle = \langle\alpha_{i+1}, \ldots, \alpha_n\rangle$ where $0 \leq i \leq n$. Thus, $\hat{C} \equiv C^{\theta_{\mathrm{c},\epsilon,i}} \equiv C^{\theta_{\mathrm{c},p,i}}$. It can be checked that $\hat{w} \equiv \alpha_i(\cdots(\alpha_1(w''))\cdots)$. Thus, $\hat{w} \equiv w^{\theta_{\mathrm{s},\epsilon,i}} \equiv w^{\theta_{\mathrm{s},p,i}}$.

($\Uparrow$) Suppose there is an i such that $\hat{C} \equiv C^{\theta_{\mathrm{c},p,i}} \equiv C^{\theta_{\mathrm{c},\epsilon,i}}$ and $\hat{w} \equiv w^{\theta_{\mathrm{s},p,i}} \equiv w^{\theta_{\mathrm{s},\epsilon,i}}$. Then $\hat{C} \equiv \alpha_n(\cdots(\alpha_{i+1}(\Box))\cdots)$ and $\hat{w} \equiv \alpha_i(\cdots(\alpha_1(w''))\cdots)$. Then $\hat{C}[\hat{w}] \equiv w''^{\langle\vec{\alpha}\rangle} \equiv w^\theta$.

2. Suppose $C \equiv X(\vec{s}^n, \bar{C}, \vec{t}^m)$ for some $X \in \mathrm{Fun} \cup \mathrm{MVar}$, some context $\bar{C}$, and some metaterms $\vec{s}$ and $\vec{t}$. Let $\breve{C} \equiv X(\vec{s}, \Box, \vec{t})$. Let $\bar{w} \equiv \bar{C}[w']$. It holds that $w \equiv \breve{C}[\bar{w}]$.

   Let $k = n+1$ and let $\bar{\theta} = \theta.k$. Observe that $\bar{\theta}$ is an L-labelling for $\bar{w}$. Let $\breve{\theta} = \theta[k \mapsto \langle\rangle]$. Observe that $w^\theta \equiv \breve{C}^{\breve{\theta}}[\bar{w}^{\bar{\theta}}]$.

   The two directions are proved separately.

   ($\Downarrow$) Suppose $w^\theta \equiv \hat{C}[\hat{w}]$.

   ($\Uparrow$)

3. Suppose $C \equiv [x]\bar{C}$ for some variable x and context $\bar{C}$. This is proved similarly to the previous case, except that the proof is simpler. □

Pattern/Valuation Labelling Decomposition Given an instantiated and then labelled linear pattern $\nu(s)^\theta$, it is desirable to be able to decompose the labelling θ into a portion applied to the pattern s and portions applied to each metaterm in the range of the valuation ν. Given a linear pattern s, a valuation ν for s, and a labelling θ for $\nu(s)$, define the labelling θ_s and the labelled valuation $\nu_{s,\theta}$ as follows.

$$\theta_s = \{\, p \mapsto \theta(p) \mid p \in \mathrm{Int}(s) \,\} \cup \{\, p \mapsto \langle\rangle \mid p \in \mathrm{Skel}(s) - \mathrm{Int}(s) \,\}$$
$$\nu_{s,\theta} = \{\, Z \mapsto (\nu(Z))^{\theta.q} \mid Z \in \mathrm{MV}(s), \mathrm{Tree}(s)(q) = Z \,\}$$

Observe that $\nu_{s,\theta}$ is a function because s is required to be linear.

Lemma 4.5. *Let s be a linear pattern, let ν be a valuation for s, and let θ be a labelling for $\nu(s)$. Let $\hat{\theta}$ be any LHS labelling for s and let $\hat{\nu}$ be any valuation for $s^{\hat{\theta}}$. Then the following equivalence holds.*

$$(\nu(s))^\theta \equiv \hat{\nu}(s^{\hat{\theta}}) \iff (\hat{\nu} = \nu_{s,\theta} \text{ and } \hat{\theta} = \theta_s)$$

□

Proof. By induction on s, with the base case being where s is of the form $Z(\vec{x})$. □

Theorem 4.6 (Labelled/Unlabelled Reduction Step Correspondence). *With respect to labelling scheme $\mathcal{L} = \langle L, c, d, P\rangle$, the following statements hold:*

1. *If θ is a LHS labelling for unlabelled rule r and $u \xrightarrow{\Delta}_{r^\theta} v$, then $\|u\| \xrightarrow{\|\Delta\|}_r \|v\|$.*
2. *Let r be an unlabelled rule, let θ be a complete $\mathcal{L}$-labelling for u, and let $u \xrightarrow{\Delta}_r v$. Then there are a unique LHS labelling θ' for r, a unique term v', and a unique redex Δ' such that $u^\theta \xrightarrow{\Delta'}_{r^{\theta'}} v'$ where $\|\Delta'\| = \Delta$. Furthermore, $v' \equiv v^{\hat{\theta}}$ for some complete $\mathcal{L}$-labelling $\hat{\theta}$.* □

Proof.

1. Let $\Delta = (u.p, r^\theta)$ and let $r = s \to t$. There exists some context C^p such that $u \equiv C^p[\nu(s^\theta)] \xrightarrow{p}_{r^\theta} C^p[\nu(t^{\overline{\theta}})] \equiv v$. Let C' be the context such that $\|C^p\| \equiv C'^q$. By lemma 4.3, it holds that $\|u\| \equiv \|C\|[\|\nu\|(\|s^\theta\|)] \equiv C'[\|\nu\|(s)] \xrightarrow{q}_r C'[\|\nu\|(t)] \equiv \|C\|[\|\nu\|(\|t^{\overline{\theta}}\|)] \equiv \|v\|$. Observe that $\|\Delta\| = (\|u.p\|, \|r^\theta\|) = (\|u\|.q, r)$ is therefore the contracted redex.

2. First we will make some definitions. Let $\Delta = (u.p, r)$ and $r = s \to t$. There exists some C and ν such that $u \equiv C^p[\nu(s)] \xrightarrow{\Delta}_r C^p[\nu(t)] \equiv v$. Let $\theta' = (\theta_{s,p,1})_s$ and $\nu' = \nu_{s,(\theta_{s,p,1})}$. Let $C'^{p'} \equiv C^{\theta_{c,p,1}}$ and $\Delta' = (u^\theta.p', r^{\theta'})$. Recall the definition that $r^{\theta'} = s^{\theta'} \to t^{\overline{\theta'}}$.

   Now that we have established some definitions, we will first show that the LHS labelling θ' and redex Δ' together with the term v' that results from contracting Δ' satisfy the desired properties. Then we will show their uniqueness.

   By lemma 4.4, we know that $u^\theta \equiv C^{\theta_{c,p,1}}[(\nu(s))^{\theta_{s,p,1}}]$. Observe that θ' is a valid LHS labelling for s. By lemma 4.5, we know that $(\nu(s))^{\theta_{s,p,1}} \equiv \nu'(s^{\theta'})$. Thus, $u^\theta \equiv C'^{p'}[\nu'(s^{\theta'})]$. Thus, we know for appropriate v' that $u^\theta \equiv C'^{p'}[\nu'(s^{\theta'})] \xrightarrow{\Delta'}_{r^{\theta'}} C'^{p'}[\nu'(t^{\overline{\theta'}})] \equiv v'$. It holds that $\|r^{\theta'}\| = r$. By definition, $\|u^\theta.p'\| \equiv u.p$. Thus, $\|\Delta'\| = \Delta$. By theorem 4.6 part 1, it holds that $u \xrightarrow{\Delta}_r \|v'\|$. Thus, $\|v'\| \equiv v$. We know that $v' \equiv C'^{p'}[\nu'(t^{\overline{\theta'}})] \equiv C^{\theta_{c,p,1}}[\nu_{s,(\theta_{s,p,1})}(t^{\overline{(\theta_{s,p,1})_s}})]$. From this it is possible (details omitted) to construct a $\mathcal{L}$-labelling $\hat{\theta}$ such that $v' \equiv v^{\hat{\theta}}$ and to show that $\hat{\theta}$ is complete.

   We have now shown the existence of the appropriate LHS labelling θ', redex Δ', and term v'. What remains to be shown is their uniqueness.

   Suppose $u^\theta \xrightarrow{\Delta''}_{r^{\theta''}} v''$ where $\|\Delta''\| = \Delta$. Let $\Delta'' = (u^\theta.p'', r^{\theta''})$. We know that there exists some C'' and ν'' such that $u^\theta \equiv C''^{p''}[\nu''(s^{\theta''})]$. We know that $\|C''\| \equiv C$ because $\|u^\theta.p''\| \equiv u.p$. By lemma 4.4, we know that $C'' \equiv C^{\theta_{c,p,i}}$ and $\nu''(s^{\theta''}) \equiv (\nu(s))^{\theta_{s,p,i}}$ for some i. Because θ'' is a LHS labelling, $\nu''(s^{\theta''})$ must have exactly one label in outermost position. Thus, $i = 1$ and therefore $C''^{p''} \equiv C'^{p'}$ and $\nu'(s^{\theta'}) \equiv \nu''(s^{\theta''})$. By lemma 4.5, we know that $\theta'' = \theta'$ and $\nu'' = \nu'$. Thus, $\Delta'' = \Delta'$ and $v'' \equiv v'$. □

Reduction Sequence Labelling A *reduction sequence labelling* for a reduction sequence σ is a pair $(\mathcal{L}, \theta)$ of a labelling scheme $\mathcal{L}$ and a complete $\mathcal{L}$-labelling θ for $\sigma[0]$ where σ is unlabelled w.r.t. $\mathcal{L}$. The labelling scheme is omitted when it is obvious. Alternatively, we will refer to θ as a $\mathcal{L}$-labelling for σ. The labelling of σ by θ, written σ^θ, is the unique (by theorem 4.6) reduction sequence such that $(\sigma^\theta)[0] \equiv (\sigma[0])^\theta$ and $\|\sigma^\theta\| = \sigma$. Let R be the set of reduction rules used in σ^θ. Observe that if σ is a Σ-reduction sequence, then σ^θ will be a valid $\Sigma^{\mathcal{L}}$-reduction sequence iff $P(\mathrm{Deg}(r))$ holds for all $r \in R$.

Lemma 4.7 (Preservation of Internal Positions under Erasure). *Let s be a pattern and let ν be a valuation for s. Let $p \in \mathrm{Int}(s)$ be such that $\|\nu(s).p\| \equiv \|\nu(s)\|.q$. Let $\breve{p}$ be a path such that $p \leq \breve{p}$, $\breve{p} \in \mathrm{Int}(s)$, and $\mathrm{Tree}(s)(\breve{p}) \notin L$. Then $q \in \mathrm{Int}(\|s\|)$.* □

Proof. By induction on the size of s. □

Theorem 4.8 (Properties of Labelled CRS's). *Let Σ be a CRS and $\mathcal{L} = \langle L, c, d, P\rangle$ a labelling scheme.*

1. *$\Sigma^{\mathcal{L}}$ is a CRS.*
2. *$\Sigma^{\mathcal{L}}$ is orthogonal if Σ is orthogonal.*

□

Proof.

1. We need to check that $\mathrm{Ter}(\Sigma^{\mathcal{L}})$ is closed under $\longrightarrow_{\mathrm{Red}(\Sigma^{\mathcal{L}})}$. Suppose $u \xrightarrow{\Delta}_r v$ where $u \in \mathrm{Ter}(\Sigma^{\mathcal{L}})$, $\Delta = (u.p, r)$, and $r \in \mathrm{Red}(\Sigma^{\mathcal{L}})$. Thus, $u \equiv \hat{u}^\theta$ for some term $\hat{u} \in \mathrm{Ter}(\Sigma)$ and some complete $\mathcal{L}$-labelling θ for $\hat{u}$. Also, $r = \hat{r}^{\hat{\theta}}$ where $\hat{r} \in \mathrm{Red}(\Sigma)$ and $\hat{\theta}$ is a LHS $\mathcal{L}$-labelling for $\hat{r}$. By theorem 4.6 part 1, we know that $\hat{u} \xrightarrow{\hat{\Delta}}_{\hat{r}} \|v\|$ where $\hat{\Delta} = (\hat{u}.\hat{p}, \hat{r})$ for some $\hat{p}$. Let $\hat{v} \equiv \|v\|$. Because $\hat{r} \in \mathrm{Red}(\Sigma)$ and Σ is a CRS, $\hat{v} \in \mathrm{Ter}(\Sigma)$. By theorem 4.6 part 2, it holds that r, Δ, and v are unique and that $v \equiv \hat{v}^{\breve{\theta}}$ for some complete $\mathcal{L}$-labelling $\breve{\theta}$. Thus $v \in \mathrm{Ter}(\Sigma^{\mathcal{L}})$.
2. Assume that Σ is an orthogonal CRS. We must show that $\mathrm{Red}(\Sigma^{\mathcal{L}})$ is left-linear and non-ambiguous. It is easy to see that if θ is a LHS labelling for r, then r^θ is left-linear iff r is left-linear. Thus, $\mathrm{Red}(\Sigma^{\mathcal{L}})$ is left-linear.

Now we will show that $\mathrm{Red}(\Sigma^{\mathcal{L}})$ is non-ambiguous. We will suppose that $\mathrm{Red}(\Sigma^{\mathcal{L}})$ is ambiguous and derive a contradiction. Suppose for $r, r' \in \mathrm{Red}(\Sigma^{\mathcal{L}})$ that r and r' are ambiguous. Let $\mathrm{LHS}(r) = s$ and $\mathrm{LHS}(r') = s'$. W/o.l.o.g., let s' interfere with s at p. (We assume w/o.l.o.g. that s' has the lower position.) Thus, we know that $p \in \mathrm{Int}(s)$ and there exist valuations ν and ν' and context C^p such that $\nu(s) \equiv C^p[\nu'(s')]$. If $r = r'$, then it must hold that $p \neq \epsilon$.

It holds that $\|\nu(s)\| \equiv \|C^p[\nu'(s')]\|$. Thus by lemma 4.3, $\|\nu\|(\|s\|) \equiv \|C\|[\|\nu'\|(\|s'\|)]$. Let the hole in $\|C\|$ be at q. By definition, $\|\nu(s).p\| \equiv \|\nu(s)\|.q$. By lemma 4.7, $q \in \mathrm{Int}(\|s\|)$. Therefore, $\|s\|$ and $\|s'\|$ interfere at q. Thus $\|r\|$ and $\|r'\|$ interfere at q.

If $q \neq \epsilon$ or $\|r\| \neq \|r'\|$, then this means that Σ is ambiguous, a contradiction. Suppose $q = \epsilon$ and $\|r\| = \|r'\|$. Thus, $\|C\| = \Box$. Thus, C mentions only labels from L and $\Box$. If $C \not\equiv \Box$, then s and s' have a different number of outermost labels, contradicting the way rules are labelled. Thus, $C \equiv \Box$. Thus, $p = \epsilon$. Let $\|r\| = \|r'\| = \breve{r} = \breve{s} \to \breve{t}$. It holds that $r = \breve{r}^{\theta}$ and $r = \breve{r}^{\theta'}$ for some LHS labellings θ and θ'. Observe that $\nu(s) \equiv u^{\bar{\theta}}$ for some term u and labelling $\tilde{\theta}$. It holds that $\nu(\breve{s}^{\theta}) \equiv \nu(s) \equiv u^{\bar{\theta}} \equiv \nu'(s') \equiv \nu'(\breve{s}^{\theta'})$. By lemma 4.5, $\theta = \tilde{\theta}_{\breve{s}} = \theta'$. Thus, $r = r'$, a contradiction. $\Box$

4.3 Labels for Tracing Subterms and Redexes

Tracing A labelling scheme $\mathcal{L} = \langle L, c, d, P\rangle$ is a *tracing labelling scheme* iff $c(\alpha, \beta) = \alpha$ for $\alpha, \beta \in L$. Given tracing labelling scheme $\mathcal{L} = \langle L, c, d, P\rangle$ and $M \subseteq L$, let $\mathrm{PosLabelIn}_c(\theta, M) = \{\, p \mid c(\theta(p)) \in M \,\}$ (i.e., the set of all positions labelled by θ with labels in the label set M).

The following lemma will establish that tracing labelling schemes are sufficiently well-behaved to be used for defining the notion of descendant.

Lemma 4.9 (Tracing Sanity). *Let $\mathcal{L}_i = \langle L_i, c_i, d_i, P_i\rangle$ be a tracing labelling scheme for $i \in \{1,2\}$. Let σ be a finite reduction sequence. For $i \in \{1,2\}$, let θ_i be an $\mathcal{L}_i$-labelling for σ. For $i \in \{1,2\}$ and $j \in \{0, \ldots, |\sigma|\}$, define $\theta_{i,j}$ as the labelling such that $\sigma^{\theta_i}[j] = \sigma[j]^{\theta_{i,j}}$. Let $M_i \subseteq L_i \setminus \mathrm{Ran}(d_i)$ for $i \in \{1,2\}$. Let $\mathrm{PosLabelIn}_{c_1}(\theta_1, M_1) \subseteq \mathrm{PosLabelIn}_{c_2}(\theta_2, M_2)$. Then for $0 \leq j \leq |\sigma|$ it holds that $\mathrm{PosLabelIn}_{c_1}(\theta_{1,j}, M_1) \subseteq \mathrm{PosLabelIn}_{c_2}(\theta_{2,j}, M_2)$.* $\Box$

Proof. The proof is by induction on j, the index into the reduction sequence σ. The base case where $j = 0$ is trivial because $\theta_i = \theta_{i,0}$ for $i \in \{1,2\}$.

Otherwise, consider the case where $j \geq 1$. Let $k = j - 1$. By induction hypothesis, we know that $\mathrm{PosLabelIn}_{c_1}(\theta_{1,k}, M_1) \subseteq \mathrm{PosLabelIn}_{c_2}(\theta_{2,k}, M_2)$. Let the jth reduction step in σ and σ^{θ_i} for $i \in \{1,2\}$ be as follows where $r = s \to t$:

$$\begin{array}{lll} \sigma[k] & \equiv C^p[\nu(s)] & \xrightarrow{p}_r \quad C^p[\nu(t)] \equiv \sigma[j] \\ \sigma[k]^{\theta_{i,k}} & \equiv C_i^{p_i}[\nu_i(s^{\theta'_{i,j}})] & \xrightarrow[r^{\theta'_{i,j}}]{p_i} C_i^{p_i}[\nu_i(\overline{t^{\theta'_{i,j}}})] \equiv \sigma[j]^{\theta_{i,j}} \end{array}$$

We now want to show that $\mathrm{PosLabelIn}_{c_1}(\theta_{1,j}, M_1) \subseteq \mathrm{PosLabelIn}_{c_2}(\theta_{2,j}, M_2)$.

Define the predicate $\mathrm{LabelPattern}_L(u, p, \theta, q, \alpha)$ to hold iff θ is an L-labelling for u, $\|u^{\theta}.q\| \equiv \|u^{\theta}.(q \cdot 1)\| \equiv u.p$, $\mathrm{Tree}(u^{\theta})(q) = \alpha$, and $\mathrm{Tree}(u^{\theta})(q \cdot 1) \notin L$.

Suppose $q \in \mathrm{PosLabelIn}_{c_1}(\theta_{1,j}, M_1)$. Thus, $c_1(\theta_{1,j}(q)) \in M_1$. Thus, $\theta_{1,j}(q) = \langle \alpha, \ldots\rangle$ for some $\alpha \in M_1$. Thus, there is some q_1 such that $\mathrm{LabelPattern}_{L_1}(\sigma[j], q, \theta_{1,j}, q_1, \alpha)$. We now show that $q \in \mathrm{PosLabelIn}_{c_2}(\theta_{2,j}, M_2)$ by separately considering the possible cases.

1. Suppose $q_1 \cdot 1 \not\geq p_1$. It must hold that $q \not\geq p$ and that $\mathrm{LabelPattern}_{L_1}(\sigma[k], q, \theta_{1,k}, q_1, \alpha)$, because nothing involved in making $\mathrm{LabelPattern}_{L_1}(\sigma[j], q, \theta_{1,j}, q_1, \alpha)$ true changed from $\sigma[k]$ to $\sigma[j]$. Thus, $q \in \mathrm{PosLabelIn}_{c_1}(\theta_{1,k}, M_1)$. By induction hypothesis, $q \in \mathrm{PosLabelIn}_{c_2}(\theta_{2,k}, M_2)$. Thus, there exists some q_2 and some $\beta \in M_2$ such that $\mathrm{LabelPattern}_{L_2}(\sigma[k], q, \theta_{2,k}, q_2, \beta)$. It must hold that $q_2 \cdot 1 \not\geq p_2$. Thus, $\mathrm{LabelPattern}_{L_2}(\sigma[j], q, \theta_{2,j}, q_2, \beta)$. Thus, $q \in \mathrm{PosLabelIn}_{c_2}(\theta_{2,j}, M_2)$.

2. Suppose $q_1 \cdot 1 = p_1$ or $q_1 = p_1$. By the definition of rule labelling, we know that $\mathrm{Tree}(\overline{t^{\theta'_{1,j}}})(\epsilon) \in \mathrm{Ran}(d_1)$. Thus, $\mathrm{Tree}(\sigma[j]^{\theta_{1,j}})(p_1) \in \mathrm{Ran}(d_1)$. Observe that $\alpha \notin \mathrm{Ran}(d_1)$ because $\alpha \in M_1$. Because $\alpha \notin \mathrm{Ran}(d)$, we know that $q_1 \neq p_1$. Because $\mathrm{Tree}(\sigma[j]^{\theta_{1,j}})(q_1 \cdot 1) \notin L_1$ and $\mathrm{Ran}(d_1) \subseteq L_1$, we can deduce that $q_1 \cdot 1 \neq p_1$. Thus, this case is impossible.

3. Suppose $q_1 > p_1$. First, let the functions f and g, which depend on ν_i and $t^{\overline{\theta'_{i,j}}}$ for $i \in \{1,2\}$, be the least defined functions such that

$$f(i,p',l) = \begin{cases} \{l\} & \text{if } \mathrm{Tree}(t^{\overline{\theta'_{i,j}}})(p') \notin \mathrm{MVar}, \\ \{\, p'' \mid \mathrm{Tree}(\nu_i(Z))(p'') = Z_l \,\} & \text{if } \mathrm{Tree}(t^{\overline{\theta'_{i,j}}})(p') = Z \end{cases}$$
$$g(i, l_1 \cdot \ldots \cdot l_n) = \{\, p'_1 \cdot \ldots \cdot p'_n \mid p'_m \in f(i, l_1 \cdot \ldots \cdot l_{m-1}, l_m) \text{ for } 1 \le m \le n \,\}$$

Because $\alpha \in M_1 \subseteq L_1 \setminus \mathrm{Ran}(d_1)$, by the definition of rule labelling we know that α does not occur in $t^{\overline{\theta'_{1,j}}}$. Thus, there exist $\hat{q}_1$, $\bar{q}_1$, and $\acute{q}_1$ such that $q_1 = p_1 \cdot \acute{q}_1 \cdot \bar{q}_1$, $\acute{q}_1 \in g(1, \hat{q}_1)$, $\mathrm{Tree}(t^{\overline{\theta'_{1,j}}})(\hat{q}_1) = Z$, $\mathrm{Tree}(\nu_1(Z))(\bar{q}_1) = \alpha$, and $\mathrm{Tree}(\nu_1(Z))(\bar{q}_1 \cdot 1) \notin L_1$. Let $\breve{q}_1$ be the path such that $\mathrm{Tree}(s^{\theta'_{1,j}})(\breve{q}_1) = Z$. Let $\tilde{q}_1 = p_1 \cdot \breve{q}_1 \cdot \bar{q}_1$. There exists a $\tilde{q}$ such that $\mathrm{LabelPattern}_{L_1}(\sigma[k], \tilde{q}, \theta_{1,k}, \tilde{q}_1, \alpha)$. Let $\breve{q}$ be the path such that $\|s^{\theta'_{1,j}}.\breve{q}_1\| \equiv s.\breve{q}$. Let $\bar{q}$ be the path such that $\|\nu_1(Z).\bar{q}_1\| \equiv \nu(Z).\bar{q}$. It holds that $\tilde{q} = p \cdot \breve{q} \cdot \bar{q}$. It holds that $\tilde{q} \in \mathrm{PosLabelIn}_{c_1}(\theta_{1,k}, M_1)$. By induction hypothesis, $\tilde{q} \in \mathrm{PosLabelIn}_{c_2}(\theta_{2,k}, M_2)$. Thus, there is a $\tilde{q}_2$ and $\beta \in M_2$ such that $\mathrm{LabelPattern}_{L_2}(\sigma[k], \tilde{q}, \theta_{2,k}, \tilde{q}_2, \beta)$. There must exist $\breve{q}_2$ and $\bar{q}_2$ such that $\tilde{q}_2 = p_2 \cdot \breve{q}_2 \cdot \bar{q}_2$, $\|s^{\theta'_{2,j}}.\breve{q}_2\| \equiv s.\breve{q}$, and $\|\nu_2(Z).\bar{q}_2\| \equiv \nu(Z).\bar{q}$. Let $\hat{q}_2$ be the path such that $\mathrm{Tree}(t^{\overline{\theta'_{2,j}}})(\hat{q}_2) = Z$ and $\|t^{\overline{\theta'_{1,j}}}.\hat{q}_1\| \equiv \|t^{\overline{\theta'_{2,j}}}.\hat{q}_2\|$. There must exist $\acute{q}_2 \in g(2, \hat{q}_2)$ such that $\|\nu_1(t^{\overline{\theta'_{1,j}}}).\acute{q}_1\| \equiv \|\nu_2(t^{\overline{\theta'_{2,j}}}).\acute{q}_2\|$. Let $q_2 = p_2 \cdot \acute{q}_2 \cdot \bar{q}_2$. It must hold that $\mathrm{LabelPattern}_{L_2}(\sigma[j], q, \theta_{2,j}, q_2, \alpha)$. Thus, $q \in \mathrm{PosLabelIn}_{c_2}(\theta_{2,j}, M_2)$. □

Descendant, Ancestor Given finite reduction sequence σ and labelling $(\mathcal{L}, \theta)$ for σ where $\mathcal{L} = \langle L, c, d, P \rangle$ is a tracing labelling scheme, the *tracing relation* $[\![\sigma]\!]^\theta$ is the relation on subterm occurrences such that $(\sigma[0].p)\ [\![\sigma]\!]^\theta\ (\sigma[|\sigma|].q)$ holds iff $c(\theta(p)) = c(\theta'(q)) \notin \mathrm{Ran}(d)$ where $(\sigma^\theta)[|\sigma|] \equiv (\sigma[|\sigma|])^{\theta'}$. This can be abbreviated as $p\ [\![\sigma]\!]^\theta\ q$.

A tracing labelling scheme $\mathcal{L} = \langle L, c, d, P \rangle$ is also a *descendant labelling scheme* iff $L - \mathrm{Ran}(d)$ is infinite. A *descendant labelling* for reduction sequence σ is a complete $\mathcal{L}$-labelling θ for σ where $\mathcal{L} = \langle L, c, d, P \rangle$ is a descendant labelling scheme, $c(\theta(p)) \notin \mathrm{Ran}(d)$ for all p, and if $c_{\mathrm{desc}}(\theta(p)) = c_{\mathrm{desc}}(\theta(q))$ then $p = q$.

Let ϕ be any choice function such that $\phi(\sigma)$ is a descendant labelling for σ. Then the *descendant relation* of a finite reduction sequence σ is $[\![\sigma]\!] = [\![\sigma]\!]^{\phi(\sigma)}$. (Lemma 4.10 shows that $[\![\sigma]\!]$ does not depend on the choice of ϕ or the labelling scheme used by ϕ.) With respect to reduction sequence σ, we say that $\sigma[k].q$ is a *descendant* of $\sigma[j].p$ (and $\sigma[j].p$ is an *ancestor* of $\sigma[k].q$) where $j \le k$ iff $(\sigma[j].p)\ [\![\sigma[j..k]]\!]\ (\sigma[k].q)$.

$\mathcal{L}_{\mathrm{desc}}$ and $\Sigma^{\mathcal{A}}$ An example descendant labelling scheme is as follows. Let $\mathcal{A}$ be a countably infinite set of labels. Let $\circ \notin \mathcal{A}$ be a distinguished label (meaning "no label"). Let $L_{\mathrm{desc}} = \mathcal{A} \cup \{\circ\}$, $c_{\mathrm{desc}}(\alpha, \beta) = \alpha$, $d_{\mathrm{desc}}(\alpha) = \circ$, and $P_{\mathrm{desc}}(\alpha) = \mathit{true}$. Let $\mathcal{L}_{\mathrm{desc}} = \langle L_{\mathrm{desc}}, c_{\mathrm{desc}}, d_{\mathrm{desc}}, P_{\mathrm{desc}} \rangle$ and let $\Sigma^{\mathcal{A}} = \Sigma^{\mathcal{L}_{\mathrm{desc}}}$.

Lemma 4.10 (Properties of $[\![\sigma]\!]$). *The descendant relation has the following nice properties:*

1. *Let θ and θ' be descendant labellings for a finite reduction sequence σ. Then $[\![\sigma]\!]^\theta = [\![\sigma]\!]^{\theta'}$.*
2. *$u.p\ [\![\sigma_1 \star \sigma_2]\!]\ v.q$ iff there exists path p' such that $u.p\ [\![\sigma_1]\!]\ u'.p'$ and $u'.p'\ [\![\sigma_2]\!]\ v.q$ where $u' \equiv \sigma_2[0]$.* □

Proof.

1. Suppose $u.p\ [\![\sigma]\!]^\theta\ v.q$. Let θ be an $\mathcal{L}$-labelling for $\mathcal{L} = \langle L, c, d, P \rangle$ and let θ be an $\mathcal{L}'$-labelling for $\mathcal{L}' = \langle L', c', d', P' \rangle$. Let $\sigma^\theta[|\sigma|] = \sigma[|\sigma|]^{\tilde{\theta}}$ and let $\sigma^{\theta'}[|\sigma|] = \sigma[|\sigma|]^{\tilde{\theta}'}$. By the definition of $[\![\sigma]\!]^\theta$, we know that $c(\theta(p)) = c(\tilde{\theta}(q)) = \alpha$ for some $\alpha \in L$. Let $\beta = c'(\theta'(p))$. Observe that $\mathrm{PosLabelIn}_c(\theta, \{\alpha\}) = \mathrm{PosLabelIn}_{c'}(\theta', \{\beta\})$. By two applications of lemma 4.9, $\mathrm{PosLabelIn}_c(\tilde{\theta}, \{\alpha\}) = \mathrm{PosLabelIn}_{c'}(\tilde{\theta}', \{\beta\})$. Because $q \in \mathrm{PosLabelIn}_c(\tilde{\theta}, \{\alpha\})$, it also holds that $q \in \mathrm{PosLabelIn}_{c'}(\tilde{\theta}', \{\beta\})$. Thus, $c'(\tilde{\theta}'(q)) = c'(\theta'(p))$. Thus, $u.p\ [\![\sigma]\!]^{\theta'}\ v.q$. Repeating this reasoning in the other direction allows us to conclude that $u.p\ [\![\sigma]\!]^\theta\ v.q$ iff $u.p\ [\![\sigma]\!]^{\theta'}\ v.q$, which proves that $[\![\sigma]\!]^\theta = [\![\sigma]\!]^{\theta'}$.

2. Let $\sigma_0 = \sigma_1 \star \sigma_2$. Let $(\mathcal{L}_i, \theta_i) = \phi(\sigma_i)$ where $\mathcal{L}_i = \langle L_i, c_i, d_i, P_i \rangle$ for $i \in \{0,1,2\}$. For $i \in \{0,1,2\}$, for $0 \le j \le |\sigma_i|$, let $\sigma_i{}^{\theta_i}[j] \equiv \sigma_i[j]^{\theta_i^j}$. Let $M_i = \{c_i(\theta_i(p))\}$ and let $Q_i = \mathrm{PosLabelIn}_{c_i}(\theta_i^{|\sigma_1|}, M_i)$ for $i \in \{0,1\}$. By lemma 4.9, it holds that $Q_0 = Q_1$. Let $M_2 = \{\, \alpha \mid \hat{p} \in Q_1, c_2(\theta_2(\hat{p})) = \alpha \,\}$. Let

$Q_2 = \mathrm{PosLabelIn}_{c_2}(\theta_2, M_2)$. It holds that $Q_1 = Q_2$ because θ_2 is a descendant labelling. Let $Q_i' = \mathrm{PosLabelIn}_{c_i}(\theta_i^{|\sigma_i|}, M_i)$ for $i \in \{0,2\}$. By lemma 4.9, it holds that $Q_0' = Q_2'$.

Suppose $u.p\ [\![\sigma_0]\!]\ v.q$. Thus, $q \in Q_0' = Q_2'$. Let $\beta = \theta_2^{|\sigma_2|}(q)$. We know that $\beta \in M_2$. Thus, there is some p' such that $c_2(\theta_2(p')) = \beta$ and $p' \in Q_2$. Thus, $u.p\ [\![\sigma_1]\!]\ u'.p'$ and $u'.p'\ [\![\sigma_2]\!]\ v.q$.

Suppose there is some p' such that $u.p\ [\![\sigma_1]\!]\ u'.p'$ and $u'.p'\ [\![\sigma_2]\!]\ v.q$. We know immediately that $p' \in Q_1 = Q_2$. Let $c_2(\theta_2(p')) = \beta \in M_2$. It holds that $q \in \mathrm{PosLabelIn}_{c_2}(\theta_2^{|\sigma_2|}, \{\beta\}) \subseteq \mathrm{PosLabelIn}_{c_2}(\theta_2^{|\sigma_2|}, M_2) = Q_2' = Q_0'$. Thus, $u.p\ [\![\sigma_0]\!]\ v.q$. □

Residual, Created Redex Given $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow v$, the descendant relation $[\![\sigma]\!]$ is extended into a *residual relation* on redex occurrences as follows. We say that $(u.p, r)\ [\![\sigma]\!]\ (v.q, r)$ iff $u.(p \cdot q')\ [\![\sigma]\!]\ v.(q \cdot q')$ for all $q' \in \mathrm{Int}(r)$.[6] With respect to a reduction sequence σ, a redex occurrence $(\sigma[k].q, r)$ is a *residual* of redex occurrence $(\sigma[j].p, r)$ where $j \leq k$ iff $(\sigma[k].q, r)\ [\![\sigma[j..k]]\!]\ (\sigma[j].p, r)$. If σ contains the reduction step $u \xrightarrow{\Delta} v$ and Δ' is a redex occurrence in v, then the (contraction of the) redex occurrence Δ *created* the redex occurrence Δ' iff Δ' is not a residual of any redex occurrence in u.

Lemma 4.11. *$(u.p, r)\ [\![\sigma_1 \star \sigma_2]\!]\ (v.q, r)$ iff $(u.p, r)\ [\![\sigma_1]\!]\ (u'.p', r)$ and $(u'.p', r)\ [\![\sigma_2]\!]\ (v.q, r)$ for some path p' where $u' \equiv \sigma_2[0]$.* □

Lemma 4.12. *Let $\sigma = u_0 \longrightarrow\!\!\!\!\!\rightarrow u_n$ be a reduction sequence, let $n = |\sigma|$, and let θ_0 be an $\mathcal{L}$-labelling for σ. Let θ_n be the labelling such that $(\sigma^{\theta_0})[n] \equiv {u_n}^{\theta_n}$. Let q_i and q_i' be the paths such that $u_i.q_i^{\theta_i} \equiv {u_i}^{\theta_i}.q_i'$ for $i \in \{0, n\}$. Then it holds that:*

1. *$u_0.q_0\ [\![\sigma]\!]\ u_n.q_n$ iff $({u_0}^{\theta_0}.q_0')\ [\![\sigma^{\theta_0}]\!]\ ({u_n}^{\theta_n}.q_n')$.*
2. *$(u_0.q_0, r)\ [\![\sigma]\!]\ (u_n.q_n, r)$ iff $({u_0}^{\theta_0}.q_0'', r^{\theta'})\ [\![\sigma^{\theta_0}]\!]\ ({u_n}^{\theta_n}.q_n'', r^{\theta'})$ for some unique LHS labelling θ' where $q_i' = q_i'' \cdot 1$ for $i \in \{0, n\}$.* □

Proof.

1. Let $\mathcal{L} = \langle L, c, d, P\rangle$. The proof also uses $\mathcal{L}_{\mathrm{desc}}$. Assume w/o.l.o.g. that $L \cap L_{\mathrm{desc}} = \varnothing$.

   We proceed by induction on n (the length of σ). We consider the following cases.

   (a) If $n = 0$, the result is immediate.

   (b) Consider the case where $n \geq 1$. Let $\sigma = u_0 \xrightarrow{p}_r u_1$ where $r = s \to t$. There exist various contexts, paths, valuations, and labellings such that $\|C'\| \equiv \|\breve{C}\| \equiv \|\hat{C}\| \equiv \|C\|$, θ' and $\hat{\theta}$ are LHS labellings for s, $\breve{\theta}$ is a LHS labelling for $s^{\theta'}$, $\hat{\theta}_i$ is a complete labelling for u_i for $i \in \{0,1\}$, $\breve{\theta}_i$ is a complete labelling for ${u_i}^{\theta_i}$ for $i \in \{0,1\}$, $\breve{\theta}$, $\breve{\theta}_0$, $\breve{\theta}_1$, $\hat{\theta}$, $\hat{\theta}_0$, and $\hat{\theta}_1$ are $\mathcal{L}_{\mathrm{desc}}$-labellings, $\breve{\theta}_0$ and $\hat{\theta}_0$ are descendant labellings, $\|C'^{p'}\| \equiv C^p \equiv \|\hat{C}^{\hat{p}}\|$, $\|\breve{C}^{\breve{p}}\|_{L_{\mathrm{desc}}} \equiv C'^{p'}$, and

$$\begin{array}{llll}
u_0 & \equiv C^p[\nu(s)] & \xrightarrow{p}_r & C^p[\nu(t)] \equiv u_1 \\
{u_0}^{\theta_0} & \equiv C'^{p'}[\nu'(s^{\theta'})] & \xrightarrow{p'}_{r^{\theta'}} & C'^{p'}[\nu'(t^{\overline{\theta'}})] \equiv {u_1}^{\theta_1} \\
({u_0}^{\theta_0})^{\breve{\theta}_0} & \equiv \breve{C}^{\breve{p}}[\breve{\nu}((s^{\theta'})^{\breve{\theta}})] & \xrightarrow{\breve{p}}_{(r^{\theta'})^{\breve{\theta}}} & \breve{C}^{\breve{p}}[\breve{\nu}((t^{\overline{\theta'}})^{\overline{\breve{\theta}}})] \equiv ({u_1}^{\theta_1})^{\breve{\theta}_1} \\
{u_0}^{\hat{\theta}_0} & \equiv \hat{C}^{\hat{p}}[\hat{\nu}(s^{\hat{\theta}})] & \xrightarrow{\hat{p}}_{r^{\hat{\theta}}} & \hat{C}^{\hat{p}}[\hat{\nu}(t^{\overline{\hat{\theta}}})] \equiv {u_1}^{\hat{\theta}_1}
\end{array}$$

   The claim can now be rephrased as

$$c_{\mathrm{desc}}(\hat{\theta}_0(q_0)) = c_{\mathrm{desc}}(\hat{\theta}_1(q_1)) \Leftrightarrow c_{\mathrm{desc}}(\breve{\theta}_0(q_0')) = c_{\mathrm{desc}}(\hat{\theta}_1(q_1'))$$

[6] These conditions are sufficient to ensure that all essential aspects of the identity of the redex occurrence persist through the intermediate steps. For orthogonal CRS's, the definition may be simplified to read as follows: For redex occurrences $(u.p, r)$ and $(v.q, r)$, $(u.p, r)\ [\![\sigma]\!]\ (v.q, r)$ iff $u.p\ [\![\sigma]\!]\ v.q$.

By lemma 4.3, the following hold:

$$\|\breve{C}^{\breve{p}}[\breve{\nu}((s^{\theta'})^{\breve{\theta}})]\|_L \equiv \|\breve{C}^{\breve{p}}\|_L[\|\breve{\nu}\|_L(\|(s^{\theta'})^{\breve{\theta}}\|_L)]$$
$$\|\breve{C}^{\breve{p}}[\breve{\nu}((t^{\overline{\theta'}})^{\overline{\breve{\theta}}})]\|_L \equiv \|\breve{C}^{\breve{p}}\|_L[\|\breve{\nu}\|_L(\|(t^{\overline{\theta'}})^{\overline{\breve{\theta}}}\|_L)]$$

Let $\|\breve{C}^{\breve{p}}\|_L \equiv \grave{C}^{\grave{p}}$, $\|\breve{\nu}\|_L = \grave{\nu}$, $\|(s^{\theta'})^{\breve{\theta}}\|_L \equiv \grave{s}$, and $\|(t^{\overline{\theta'}})^{\overline{\breve{\theta}}}\|_L \equiv \grave{t}$. It is clear that $\|\grave{C}^{\grave{p}}\|_{L_{\text{desc}}} \equiv C^p$, $\|\grave{\nu}\|_{L_{\text{desc}}} = \nu$, and $\|\grave{s} \to \grave{t}\|_{L_{\text{desc}}} = s \to t$. Thus, there are $\grave{\theta}$, $\grave{\theta}_0$, and $\grave{\theta}_1$ such that

$$u_0{}^{\grave{\theta}_0} \equiv \grave{C}^{\grave{p}}[\grave{\nu}(s^{\grave{\theta}})] \xrightarrow[r^{\grave{\theta}}]{\grave{p}} \grave{C}^{\grave{p}}[\grave{\nu}(s^{\overline{\grave{\theta}}})] \equiv u_1{}^{\grave{\theta}_1}$$

It holds that $\grave{\theta}_0$ is a descendant labelling because $\breve{\theta}_0$ is. Specifically, for any $\bar{q}$, for $i \in \{0,1\}$, $\grave{\theta}_i(\bar{q}) = \breve{\theta}_i(\bar{q}_m) \cdot \ldots \cdot \breve{\theta}_i(\bar{q}_1)$ where $\{\vec{\bar{q}}\} = \{\, \acute{q} \mid \|u_i{}^{\theta_i}.\acute{q}\|_L \equiv u_i.\bar{q} \,\}$ and $\vec{\bar{q}}$ are in lexicographic order. Thus,

$$c_{\text{desc}}(\hat{\theta}_0(q_0)) = c_{\text{desc}}(\hat{\theta}_1(q_1)) \Leftrightarrow c_{\text{desc}}(\grave{\theta}_0(q_0)) = c_{\text{desc}}(\grave{\theta}_1(q_1)) \Leftrightarrow c_{\text{desc}}(\breve{\theta}_0(q_0)) = c_{\text{desc}}(\breve{\theta}_1(q_1))$$

By induction hypothesis, the result holds for $\sigma[1..n]$. The result for the first step and the rest of σ can be concatenated.

2. Using previous part. □

Lemma 4.13. *Let $\mathcal{L}$ be a labelling scheme, let σ be a $\Sigma^{\mathcal{L}}$-reduction sequence, and let Δ and Δ' be $\Sigma^{\mathcal{L}}$-redex occurrences in σ such that Δ' is a residual of Δ. Then* $\text{Deg}(\Delta) = \text{Deg}(\Delta')$. □

Proof. Let $\Delta = (\sigma[i].p, r)$ and $\Delta' = (\sigma[j].q, r)$. By definition, $\text{Deg}(\Delta) = \text{Deg}(r) = \text{Deg}(\Delta')$. □

4.4 Strongly Normalizing Labelling Schemes

This subsection introduces a sufficient condition for the strong normalization of labelled reduction. The proof method leading to theorem 4.17 is adapted directly from van Oostrom's method for arbitrary-order pattern rewrite systems [vO97], with minor differences to handle the different labelling method as well as the fact that CRS's are only second-order.

Outermost Labels With respect to a labelling scheme $\mathcal{L} = \langle L, c, d, P\rangle$, the *outermost label* $\text{OutLab}(w)$ of a preterm w is α if $\text{Tree}(w)(\epsilon) = \alpha \in L$ and is otherwise undefined. A preterm w is *outermost-labelled* iff $\text{OutLab}(w)$ is defined. A valuation ν is *outermost-labelled* iff $\text{Ran}(\nu) \subseteq \text{DomDef}(\text{OutLab})$, in which case we define the set $\text{OutLab}(\nu) = \text{OutLab}(\text{Ran}(\nu))$.

Well Founded Label Ordering Let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme and let $\# : L \to \mathbb{N}$. The function $\#$ is a *well founded label ordering* for $\mathcal{L}$ iff $\#(c(\alpha_1, \alpha_2)) \leq \#(\alpha_i)$ for $i \in \{1,2\}$, $\#(d(\alpha)) < \#(\alpha)$, and $P(\alpha) \Rightarrow \#(\alpha) > 0$. Given a well founded label ordering $\#$ for $\mathcal{L}$, extend $\#$ to complete $\mathcal{L}$-labellings so that $\#(\theta) = \max\{\, \#(c(\theta(p))) \mid p \in \text{DomDef}(\theta) \,\}$. Extend $\#$ to $\mathcal{L}$-labelled preterms so that $\#(w) = \#(\theta)$ where $\text{TLD}_L(w) = (w', \theta)$ and θ is a complete labelling.

Lemma 4.14 (Outermost Labels Do Not Affect Termination). *Let Σ be a CRS, let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme, let $u \in \text{Ter}(\Sigma^{\mathcal{L}})$ be an outermost-labelled term, let $\vec{\alpha}^n \in L$, let $v \equiv u^{\langle \vec{\alpha}^n \rangle}$, and let $R = \text{Red}(\Sigma^{\mathcal{L}})$. Then R-SN$(u) \Leftrightarrow R$-SN(v).* □

Proof. Remember the definition of how R is formed from $\text{Red}(\Sigma)$ and consider any rule $r = (s \to t) \in R$. It is clear that s is of the form $\alpha(F(s'))$ for some $\alpha \in L$, some $F \notin L$, and some metaterm s', i.e., s has exactly one label in outermost position above a non-label function symbol. Also, t has exactly one label in outermost position. Thus, the labels $\vec{\alpha}$ can not participate in any R-reduction sequence starting from v. In any term, outermost labels beyond the first can not contribute to any reduction step. Thus, v is strongly R-normalizing iff u is. □

Lemma 4.15. *Let Σ be a CRS. Let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme with well founded label ordering $\# : L \to \mathbb{N}$. Let $R = \mathrm{Red}(\Sigma^{\mathcal{L}})$. Let u be a term, let s be a pattern, and let ν be a valuation for s. Let ν' be an outermost-labelled valuation such that $\#(\mathrm{OutLab}(\nu')) \leq k$. Furthermore, let $\nu'(u) \longrightarrow\!\!\!\!\!\rightarrow_R \nu(s)$. Then either $\#(s) \leq k$ or there exists a valuation ν'' such that $u \longrightarrow\!\!\!\!\!\rightarrow_R \nu''(s)$ and $\nu'(\nu''(t)) \longrightarrow\!\!\!\!\!\rightarrow_R \nu(t)$ for any t.* □

Proof. □

Lemma 4.16. *Let Σ be a CRS. Let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme with well founded label ordering $\# : L \to \mathbb{N}$. Let $R = \mathrm{Red}(\Sigma^{\mathcal{L}})$. Then both of the following statements hold:*

1. *Let s be an unlabelled metaterm, let θ be a complete L-labelling for s, and let $t \equiv s^{\theta}$. Let $k = \#(t) = \#(\theta)$. Let ν be an outermost-labelled valuation such that R-SN(ν). Then R-SN($\nu(t)$).*
2. *Let u be a term such that R-SN(u). Let ν be an outermost-labelled valuation such that R-SN(ν) and $\max(\#(\mathrm{OutLab}(\nu))) \leq k$. Then R-SN($\nu(u)$).* □

Proof. The proof proceeds by induction on k as follows. Part 1 will be proven using part 1 at $k-1$ and part 2 at k. Part 2 will be proven using part 1 at $k-1$.

1. The proof of this part will proceed by (nested) induction on s in the ordering $\lhd$ as follows.

   Without loss of generality, it is sufficient to prove this part for the case where the root of s is a metavariable Z, as the following argument shows by cases on the shape of s if its root is not a metavariable:

   (a) Suppose $s \equiv F(\vec{s}^n)$. Let $s' \equiv Z(\vec{s}^n)$ for fresh Z and let $\nu' = \nu[Z \mapsto F(Z_1, \ldots, Z_n)]$. Let $t' \equiv s'^{\theta}$. Observe that R-SN(ν') and $\nu(t) \equiv \nu'(t')$. It is clear that s', θ, t', and ν' satisfy the premises of the lemma and obviously R-SN($\nu'(t')$) iff R-SN($\nu(t)$). Thus, the lemma for this case is implied by the lemma for the case when the root of s is a metavariable.

   (b) Suppose $s \equiv [x]\hat{s}$. Let $s' \equiv Z(\hat{s})$ for fresh Z and let $\nu' = \nu[Z \mapsto [x]Z_1]$. The same reasoning now holds as in the previous case.

   (c) Suppose $s \equiv x$. Then $t \equiv x^{\theta} \equiv \alpha_1(\cdots(\alpha_n(x))\cdots)$ where $\theta(\epsilon) = \langle \vec{\alpha}^n \rangle$. Then $\nu(t) = t$. It is obvious that R-SN($\nu(t)$), by the way the rule set R was constructed.

   So let $s \equiv Z(\vec{s}^n)$. Observe that $t \equiv \alpha_m(\cdots(\alpha_1(Z({s_1}^{\theta.1}, \ldots, {s_n}^{\theta.n})))\cdots)$ where $\theta(\epsilon) = \langle \vec{\alpha}^m \rangle$. It is easy to see that $\nu(t) \equiv \alpha_m(\cdots(\alpha_1(\nu'(\nu(Z))))\cdots) \equiv \nu'(\alpha_m(\cdots(\alpha_1(\nu(Z)))\cdots))$ where $\nu' = \{\, \hat{Z}_i \mapsto u_i \mid 1 \leq i \leq n \,\}$ and $u_i \equiv \nu({s_i}^{\theta.i})$ for $1 \leq i \leq n$. By (the inner) induction hypothesis, R-SN($\nu({s_i}^{\theta.i})$) holds for $1 \leq i \leq n$. Thus, R-SN(ν'). Also, ν' is outermost-labelled because $\theta.i$ is a complete labelling derived from θ for $1 \leq i \leq n$. Furthermore, $\max(\#(\mathrm{OutLab}(\nu'))) \leq k$. Let $\dot{s} = \alpha_m(\cdots(\alpha_1(\nu(Z)))\cdots)$. By lemma 4.14, the fact that ν is outermost-labelled, and R-SN($\nu(Z)$), it holds that R-SN($\dot{s}$). We have that $\nu(t) \equiv \nu'(\dot{s})$.

   Let $u \equiv \hat{\nu}(\dot{s})$ where $\hat{\nu} = \{\, \hat{Z}_i \mapsto x_i \mid 1 \leq i \leq n \,\}$ and $\vec{x}$ are fresh. Let $\bar{\nu} = \{\, x_i \mapsto u_i \mid 1 \leq i \leq n \,\}$. It is clear that $\nu'(\dot{s}) \equiv \bar{\nu}(u)$.

   Now, using $\bar{\nu}$ for ν, we match the premises of part 2 of the lemma. Hence, by part 2, R-SN($\bar{\nu}(u)$), thus proving the desired result that R-SN($\nu(t)$).

2. Let $\ll \;=\; \lhd \cup \leftarrow_R$. The proof of this part proceeds by (nested) induction on u in the relation $\ll$ as follows. The alert reader will notice that $\ll$ is not an order and also that $\gg$ is not well founded. However, because R-SN(u), all descending chains in $\ll$ starting from u are finite (details omitted), so we can perform induction. Let $\nu = \{\, x_i \mapsto u_i \mid 1 \leq i \leq n \,\}$. By cases on the shape of u:

   (a) Suppose $u \equiv x$ for $x \notin \mathrm{DomDef}(\nu)$. Then $\nu(u) \equiv x$. It is obvious that R-SN(x).

   (b) Suppose $u \equiv x_i$ for $1 \leq i \leq n$.[7] Then $\nu(u) \equiv u_i$. We already know that R-SN(u_i).

[7] The proof differs at this point from van Oostrom's proof [vO97] for arbitrary-order pattern rewrite systems. The difference is due to the CRS restriction to second-order function symbols and (meta)variables.

(c) Suppose $s \equiv [x]u'$. By (the inner) induction hypothesis, we know that R-SN($\nu(u')$). It is clear from the definition of a CRS that the outermost binder can not participate in any reduction sequence, hence R-SN($[x]\nu(u')$), which is the desired result.

(d) Suppose $u \equiv F(\vec{v}^m)$. Thus, $\nu(u) \equiv F(\nu(v_1), \ldots, \nu(v_m))$. By (the inner) induction hypothesis, we know that R-SN($\nu(v_i)$) for $1 \leq i \leq m$. We suppose that $\neg R$-SN($\nu(u)$) and we will reach a contradiction. Suppose there is an infinite R-reduction sequence σ beginning with $\nu(u)$. If every redex contracted in σ were not at the root, then we could construct an infinite reduction sequence beginning with $\nu(v_i)$ for some $i \in \{1, \ldots, m\}$, which is impossible because R-SN($\nu(v_i)$). Thus, there must be some reduction step which contracts a redex at the root of the term.

Let $r = (s \to t) \in R$ be the rule used in the first reduction step in σ contracting a redex at the root. By the definition of how R is obtained from Red(Σ), we know that r must in fact be of the form $s \to t = \alpha(F'(\vec{s}')) \to t'^{\theta}$, where $\alpha, \beta \in L$, $F' \notin L$, t' is unlabelled, and θ is an L-labelling for t' such that $\mathrm{Ran}(\theta) = \{\langle\beta\rangle\}$. It further holds that $\#(\beta) < \#(\alpha(F'(\vec{s}')))$. The reduction sequence σ must look like this for some valuation ν':

$$\nu(u) \equiv F(\nu(v_1), \ldots, \nu(v_m)) \twoheadrightarrow_R \nu'(\alpha(F'(\vec{s}'))) \longrightarrow_r \nu'(t'^{\theta}) \twoheadrightarrow_R \cdots$$

The reduction sequence depicted can only happen if in fact $F = \alpha$ and $m = 1$. Thus, σ must look like this:

$$\nu(u) \equiv \alpha(\nu(v_1)) \twoheadrightarrow_R \alpha(F'(\nu'(s_1), \ldots, \nu'(s_l))) \equiv \nu'(s) \longrightarrow_r \nu'(t'^{\theta}) \twoheadrightarrow_R \cdots$$

Because the step using r is the first at the root, we can deduce that

$$\nu(v_1) \twoheadrightarrow_R F'(\nu'(s_1), \ldots, \nu'(s_l))$$

Because R-SN($\nu(v_1)$), we now know that R-SN($F'(\nu'(s_1), \ldots, \nu'(s_l))$). Assume w/o.l.o.g. that $\mathrm{DomDef}(\nu') = \mathrm{MV}(F'(\vec{s}'))$. By induction on the pattern $F'(\vec{s}')$ (details omitted), it holds that ν' is outermost-labelled and that R-SN(ν'). We now consider two cases.

i. Suppose $\#(\alpha(F'(\vec{s}'))) \leq k$. Then $\#(t'^{\theta}) = \#(\theta) = \#(\beta) < k$. Thus, by part 1 at $k-1$ it holds that R-SN($\nu'(t'^{\theta})$), contradicting the claim that σ is infinite.

ii. Suppose $\#(\alpha(F'(\vec{s}'))) > k$. By lemma 4.15, $u \twoheadrightarrow_R \nu''(s) \longrightarrow_r \nu''(t)$ and $\nu(\nu''(t)) \twoheadrightarrow_R \nu'(t)$. By the fact that R-SN(u) it holds that R-SN($\nu''(t)$). Because at least one reduction step occurred, $\nu''(t) \ll u$. Thus, by (the inner) induction hypothesis (using $\nu''(t)$ for u), it holds that R-SN($\nu(\nu''(t))$). By the fact that $\nu(\nu''(t)) \twoheadrightarrow_R \nu'(t)$, it holds that R-SN($\nu'(t)$), contradicting the claim that σ is infinite. □

Theorem 4.17 (Well Founded Label Ordering ⇒ SN). *Let Σ be a CRS. Let $\mathcal{L} = \langle L, c, d, P\rangle$ be a labelling scheme. with well founded label ordering $\# : L \to \mathbb{N}$. Then $\Sigma^{\mathcal{L}}$-reduction is strongly normalizing.* □

Proof. Suppose $\neg$SN($\Sigma^{\mathcal{L}}$). Let $u \in \mathrm{Ter}(\Sigma^{\mathcal{L}})$ be a minimal term w.r.t. $\lhd$ such that $\neg\Sigma^{\mathcal{L}}$-SN(u). Because u is minimal and not strongly normalizing, u is not a variable nor does it have a binder in outermost position. Thus, $u \equiv F(\vec{u}^n)$ for some F and $\vec{u}$. Because u is minimal, it holds that $\Sigma^{\mathcal{L}}$-SN(u_i) for $1 \leq i \leq n$. Let σ be an infinite reduction sequence beginning with u. Because the subterms of u are strongly normalizing, there must be a reduction step at the root in σ, i.e., σ must look like this for some rule $r = (s \to t) \in \mathrm{Red}(\Sigma^{\mathcal{L}})$ and some valuation ν:

$$u \equiv F(\vec{u}^n) \twoheadrightarrow_{\Sigma^{\mathcal{L}}} \nu(s) \longrightarrow_r \nu(t) \twoheadrightarrow_{\Sigma^{\mathcal{L}}} \cdots$$

By the way Red($\Sigma^{\mathcal{L}}$) is formed, we know that $s \to t = \alpha(s') \to t'^{\theta}$ for some label $\alpha \in L$, some metaterm s', some unlabelled metaterm t', and some complete L-labelling θ for t'. Thus, $F = \alpha$ and $n = 1$. So σ looks like this:

$$u \equiv \alpha(u_1) \twoheadrightarrow_{\Sigma^{\mathcal{L}}} \alpha(\nu(s')) \longrightarrow_r \nu(t'^{\theta}) \twoheadrightarrow_{\Sigma^{\mathcal{L}}} \cdots$$

Because $\Sigma^{\mathcal{L}}$-SN(u_1), it holds that $\Sigma^{\mathcal{L}}$-SN$(\nu(s'))$. By induction on the pattern s' (details omitted), it holds that $\Sigma^{\mathcal{L}}$-SN(ν) and that ν is outermost-labelled. Thus, by lemma 4.16, $\Sigma^{\mathcal{L}}$-SN$(\nu(t'^{\theta}))$, contradicting the claim that σ is infinite. □

REMARK 4.18. An earlier version of this paper "proved" an analogue of theorem 4.17 using a theorem of Klop [Klo80, II 6.2.4] which has been shown to be false by Melliès [Mel96]. In addition to the fatal flaw in Klop's theorem pointed out by Melliès, there were a number of technical weaknesses in Klop's approach. □

4.5 Labels for Developments

$\underline{\Sigma}$ Let $L_{\text{dev}} = \{\bullet, \circ\}$, $d_{\text{dev}}(\alpha) = \circ$, $P_{\text{dev}}(\alpha) \Leftrightarrow \alpha \neq \circ$. Let

$$c_{\text{dev}}(\alpha, \beta) = \begin{cases} \circ & \text{if } \circ \in \{\alpha, \beta\}, \\ \bullet & \text{otherwise (i.e., } \{\alpha, \beta\} = \{\bullet\}). \end{cases}$$

Let $\mathcal{L}_{\text{dev}} = \langle L_{\text{dev}}, c_{\text{dev}}, d_{\text{dev}}, P_{\text{dev}} \rangle$ and let $\underline{\Sigma} = \Sigma^{\mathcal{L}_{\text{dev}}}$.

Redex Family Labelling A set $\mathcal{F} = \{\vec{\Delta}\}$ is a *redex family* in u iff $\vec{\Delta}$ are all redex occurrences in u. Let $\mathcal{F}$ range over redex families. The *redex family labelling* for $\mathcal{F}$ in u is the unique $\mathcal{L}_{\text{dev}}$-labelling $\theta_{\mathcal{F}}$ for u such that $\text{Ran}(\theta_{\mathcal{F}}) = \{\langle\bullet\rangle, \langle\circ\rangle\}$ and $\theta_{\mathcal{F}}(p \cdot q) = \langle\bullet\rangle$ iff there is some redex occurrence $(u.p, r) \in \mathcal{F}$ such that $q \in \text{Int}(r)$.

Development Given a redex family $\mathcal{F}$ in u and a Σ-reduction sequence $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow \cdots$, we say that σ is a *development* of $\mathcal{F}$ in u iff $\sigma^{\theta_{\mathcal{F}}}$ is a valid reduction sequence of $\underline{\Sigma}$. Given a finite development $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow v$ of $\mathcal{F}$ in u and its labelled version $\sigma^{\theta_{\mathcal{F}}} = u^{\theta_{\mathcal{F}}} \longrightarrow\!\!\!\!\!\rightarrow v^{\theta}$, we say σ is a *complete development* of $\mathcal{F}$ in u iff $\underline{\Sigma}$-nf(v^{θ}).

Lemma 4.19. *Let $\mathcal{F}$ be a redex family in u.*

1. *A reduction sequence $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow \cdots$ is a development of $\mathcal{F}$ in u iff for any redex occurrence Δ contracted in σ there is some $\Delta' \in \mathcal{F}$ such that Δ is a residual of Δ'.*
2. *A development $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow v$ of $\mathcal{F}$ in u is a complete development of $\mathcal{F}$ in u iff every $\Delta \in \mathcal{F}$ has no residuals in v.* □

REMARK 4.20. If Σ is not orthogonal, then it is possible for σ to be a complete development for $\mathcal{F}$ without either contracting or discarding all residuals of redex occurrences in $\mathcal{F}$. The contraction of an overlapping redex occurrence can destroy another redex occurrence without consuming it completely. □

Theorem 4.21 (SN of $\underline{\Sigma}$). *Reduction in $\underline{\Sigma}$ is strongly normalizing.* □

Proof. By theorem 4.17, using well founded label ordering $\# = \{\bullet \mapsto 1, \circ \mapsto 0\}$. □

Corollary 4.22 (Finiteness of Developments). *If Σ is an orthogonal CRS, then all Σ-developments are finite.* □

Theorem 4.23 (Uniqueness of Complete Developments). *If Σ is an orthogonal CRS, then all complete developments of $\mathcal{F}$ in u have the same result.* □

Proof. By theorem 4.21 and theorem 3.6. □

REMARK 4.24. We can also show the standard result that all complete developments of the same redex family have the same residual relation, but we do not need this for our later results. □

4.6 Hyland/Wadsworth Labels

Σ^{HW} Let $L_{\text{HW}} = \mathbb{N}$, $c_{\text{HW}}(\alpha, \beta) = \min(\alpha, \beta)$, $d_{\text{HW}}(\alpha) = \max(0, \alpha - 1)$, and $P_{\text{HW}}(\alpha) \Leftrightarrow \alpha > 0$. Let $\mathcal{L}_{\text{HW}} = \langle L_{\text{HW}}, c_{\text{HW}}, d_{\text{HW}}, P_{\text{HW}} \rangle$ and let $\Sigma^{\text{HW}} = \Sigma^{\mathcal{L}_{\text{HW}}}$.

REMARK 4.25. Hyland/Wadsworth (HW) labels are a generalization of the notion of developments. A reduction in $\underline{\Sigma}$ can be seen as a reduction in Σ^{HW} except using the labels $\bullet$ and $\circ$ instead of 1 and 0. □

The property of the following theorem 4.26 is called *finite family developments* by van Oostrom [vO97].

Theorem 4.26. *If Σ is orthogonal, then Σ^{HW}-reduction is strongly normalizing.* □

Proof. By theorem 4.17, using well founded label ordering $\# = \{ i \mapsto i \mid i \in \mathbb{N} \}$. □

Adequate Labelling An *adequate labelling* for a Σ-reduction sequence $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow v$ is a complete $\mathcal{L}_{\text{HW}}$-labelling θ for u such that σ^θ is a valid Σ^{HW}-reduction sequence.

Van Oostrom credits an early version of the following lemma to Lévy [Bar84, sec. 14.2].

Lemma 4.27. *Any finite Σ-reduction sequence σ has an adequate labelling.* □

Proof. Let σ be a k-step reduction sequence. Choose the complete labelling θ such that $\text{Ran}(\theta) = \{\langle k \rangle\}$. Let $\sigma^\theta = \langle u_0, \Delta_0, u_1, \Delta_1, \ldots, \Delta_{k-1}, u_k \rangle$. By induction on i, the smallest label in u_i is $k-i$. Thus, $\text{Deg}(\Delta_i) \geq k-i$. Thus every redex occurrence contracted in σ^θ is a valid Σ^{HW}-redex occurrence. Thus, σ^θ is a valid Σ^{HW}-reduction sequence. □

5 Standardization via Redex Orderings

This section makes the notion of standard reduction relative to a redex ordering function $\mathcal{R}$ and identifies sufficient conditions such that $\mathcal{R}$ yields a well-behaved notion of standard reduction and a normalizing reduction strategy. Unsurprisingly, the abstract conditions we identify bear a resemblance to the conditions in [GLM92]. The standardization approach here is to obtain a standard reduction by iterating the replacement of anti-standard pairs (2-step non-standard subsequences) by the corresponding standard complete developments.

The anti-standard-pair-replacement method was first used by Klop [Klo80]. Klop's method can be seen as the restriction of ours that (1) fixes the redex ordering function $\mathcal{R}$ to order redex occurrences by a preorder traversal and (2) sets the sufficient conditions for well-behavedness to be "Σ is left-normal", where *left-normal* means that all function symbols occur to the left of all metavariables in the left-hand sides of rules.[8] The anti-standard-pair-replacement method is convenient to use with redex ordering functions that totally order all redexes in a term; other approaches to proving standardization may be more suitable when working with redex ordering functions that yield partial orders. Our proof that the anti-standard-pair-replacement procedure terminates (theorem 5.9) is significantly simpler than earlier proofs.

Redex Ordering Function Let Σ be a CRS and let $u \in \text{Ter}(\Sigma)$. A *Σ-redex occurrence ordering* of u is a sequence $\rho = [\vec{\Delta}^n]$ where for $1 \leq i \leq n$, $\vec{\Delta}_i$ is a Σ-redex occurrence in u. When Σ is clear from context, we may abbreviate "Σ-redex occurrence ordering" by "redex ordering." A redex ordering ρ is *complete* iff it mentions all of the redex occurrences in u. A *redex occurrence ordering function* for Σ is a function $\mathcal{R}$ such that for all $u \in \text{Ter}(\Sigma)$, it holds that $\mathcal{R}(u)$ is a complete redex ordering of u.

Before, After, First, and Last Let Σ be a CRS, let $\mathcal{R}$ be a redex ordering function for Σ, let $u \in \text{Ter}(\Sigma)$ and let Δ, Δ' be redex occurrences in u. We say that a Δ is *$\mathcal{R}$-before* Δ' in u iff $\Delta \prec_{\mathcal{R}(u)} \Delta'$. When the redex occurrence ordering function $\mathcal{R}$ is clear from context we may omit it. We define *after*, *first*, and *last* similarly.

[8] It is an error in Klop's definition of left-normal that it fails to mention binders and ordinary variables that are not under metavariables.

Freezing and Frozen Redex Occurrences Let Σ be an orthogonal CRS, let $\mathcal{R}$ be a Σ-redex ordering function for Σ and let $\sigma = \langle u_0, \Delta_0, u_1, \Delta_1, \ldots \rangle$ be a Σ-reduction sequence. For $0 \leq i < |\sigma|$ a Σ-redex occurrence Δ in u_i is *$\mathcal{R}$-freezing* in u_i iff Δ is $\mathcal{R}$-before Δ_i in u_i (the to-be-contracted redex occurrence) and for $1 < i \leq |\sigma|$ a redex occurrence Δ in u_i is *$\mathcal{R}$-frozen* in u_i iff there exists a u_j, $1 \leq j < i$, such that Δ is a residual of an $\mathcal{R}$-freezing redex occurrence in u_j. As above, when $\mathcal{R}$ is clear from context, it may be omitted.

REMARK 5.1. A redex occurrence can be both freezing and frozen, because these notions depend separately on the following reduction step and the preceding reduction sequence. □

$\mathcal{R}$-Standard Reduction Sequence Let Σ be an orthogonal CRS, let $\mathcal{R}$ be a Σ-redex ordering function for Σ and let $\sigma = \langle u_0, \Delta_0, u_1, \Delta_1, \ldots \rangle$ be a Σ-reduction sequence. For $0 \leq k < |\sigma|$ the kth step of σ (i.e., $u_k \xrightarrow{\Delta_k} u_{k+1}$) is *$\mathcal{R}$-standard* iff Δ_k is not $\mathcal{R}$-frozen in u_k. We say that σ is *$\mathcal{R}$-standard* iff all of its steps are $\mathcal{R}$-standard.

Good Redex Ordering Function A Σ-redex ordering function $\mathcal{R}$ is *good* for Σ iff for every 1-step Σ-reduction sequence $\sigma = u \xrightarrow{\Delta} v$, both of the following statements hold:

1. Every $\mathcal{R}$-freezing Σ-redex occurrence in u has exactly one ($\mathcal{R}$-frozen) residual in v.
2. All $\mathcal{R}$-frozen Σ-redex occurrences in v occur $\mathcal{R}$-before all non-$\mathcal{R}$-frozen Σ-redex occurrences in v.

Lemma 5.2. *With respect to a good Σ-redex ordering function $\mathcal{R}$, if σ is an $\mathcal{R}$-standard Σ-reduction sequence, then both of the following statements hold:*

1. *In every term in σ, all frozen Σ-redex occurrences are before all non-frozen Σ-redex occurrences.*
2. *Every freezing Σ-redex occurrence in σ has exactly one (frozen) residual in all subsequent terms.* □

Proof.

1. By induction on i for $0 \leq i \leq |\sigma|$. For $i = 0$, $\sigma[i] = \sigma[0]$ has no frozen redex occurrences. For $i = 1$, frozen redex occurrences in $\sigma[1]$ must be from freezing redex occurrences in $\sigma[0]$. Thus, because $\mathcal{R}$ is good, all frozen redex occurrences in $\sigma[1]$ will be before all non-frozen occurrences. For $i \geq 2$, any frozen redex occurrences in $\sigma[i]$ are 1-step residuals of either freezing or frozen redex occurrences in $\sigma[i-1]$. By the induction hypothesis, all frozen redex occurrences are before all non-frozen redex occurrences in $\sigma[i-1]$. Because σ is $\mathcal{R}$-standard, the redex occurrence contracted in $\sigma[i-1]$ is non-frozen and must be after all frozen redex occurrences in $\sigma[i-1]$. Thus, all frozen redex occurrences in $\sigma[i-1]$ are also freezing. Thus, all frozen redex occurrences in $\sigma[i]$ are residuals of freezing redex occurrences in $\sigma[i-1]$. Because $\mathcal{R}$ is good, all frozen redex occurrences must therefore be before all non-frozen redex occurrences in $\sigma[i]$.
2. Let Δ_i be a freezing Σ-redex occurrence in $\sigma[i]$. We prove the claim by induction on j for $i < j \leq |\sigma|$. For $j = i + 1$, because $\mathcal{R}$ is good, Δ_i has a single residual in $\sigma[j]$ which is frozen by definition. For $j > i + 1$, by the induction hypothesis Δ_i has a single frozen residual Δ_{j-1} in $\sigma[j-1]$. Because σ is $\mathcal{R}$-standard, the redex occurrence Δ contracted in $\sigma[j-1]$ is non-frozen. By part one of this lemma, Δ_{j-1} is before Δ and is thus freezing. Because $\mathcal{R}$ is good, Δ_{j-1} has a single (frozen) residual Δ_j in $\sigma[j]$, which must also be the sole residual of Δ_i in $\sigma[j]$. □

Lifting Redex Ordering Functions through Labellings A Σ-redex ordering function $\mathcal{R}$ on $\mathrm{Ter}(\Sigma)$ is lifted to a $\underline{\Sigma}$-redex ordering function $\underline{\mathcal{R}}$ on $\mathrm{Ter}(\underline{\Sigma})$ as follows. For $\underline{\Sigma}$-redexes Δ and Δ' in $\underline{\Sigma}$-term u, it is defined that $\Delta <_{\underline{\mathcal{R}}(u)} \Delta'$ iff $\|\Delta\| <_{\mathcal{R}(\|u\|)} \|\Delta'\|$.

Lemma 5.3. *Let Σ be an orthogonal CRS and let $\mathcal{R}$ be any Σ-redex ordering function. Both of the following statements hold:*

1. *If σ is a $\underline{\Sigma}$-reduction sequence, then σ is $\underline{\mathcal{R}}$-standard iff $\|\sigma\|$ is $\mathcal{R}$-standard.*
2. *If $\mathcal{R}$ is good for Σ, then $\underline{\mathcal{R}}$ is good for $\underline{\Sigma}$.* □

Reduction Strategy A *reduction strategy* $\mathcal{S}$ for Σ is a partial function from $\mathrm{Ter}(\Sigma)$ to Σ-redex occurrences such that $\mathcal{S}(u)$ is a Σ-redex occurrence in u.[9] If $\mathcal{S}$ is a reduction strategy, then define

$$\mathcal{S}^*(u) = \begin{cases} \langle u, \Delta, v\rangle \star \mathcal{S}^*(v) & \text{if } \mathcal{S}(u) = \Delta \text{ and } u \xrightarrow{\Delta} v, \\ \langle u\rangle \quad (0 \text{ steps}) & \text{if } \mathcal{S}(u) \text{ is undefined.} \end{cases}$$

A reduction strategy $\mathcal{S}$ for Σ is *normalizing* iff $\mathcal{S}^*(u)$ ends in a normal form whenever u has a normal form. Given Σ-redex ordering function $\mathcal{R}$, the $\mathcal{R}$-*first* reduction strategy is the least defined 1-step reduction strategy $\mathcal{S}_{\mathcal{R}}$ such that $\mathcal{S}_{\mathcal{R}}(u) = \Delta$ if Δ is the $\mathcal{R}(u)$-first Σ-redex occurrence in u.

Lemma 5.4. *Let Σ be an orthogonal CRS, $\mathcal{R}$ be any Σ-redex ordering function, $u \in \mathrm{Ter}(\Sigma)$, and $\mathcal{F}$ be a Σ-redex family in u. Then there is a $\mathcal{R}$-standard complete Σ-development of $\mathcal{F}$ in u. Furthermore, if $\mathcal{R}$ is good then there is only one such complete development.* □

Proof. We must show that (1) there exists such a standard complete development and (2) that no more than one exists if $\mathcal{R}$ is good.

Let $\theta_{\mathcal{F}}$ be the redex family labelling for $\mathcal{F}$ in u. Let $\hat{\sigma} = \mathcal{S}^*_{\underline{\mathcal{R}}}(u^{\theta_{\mathcal{F}}})$ and let $\sigma = \|\hat{\sigma}\|$. Observe that $\hat{\sigma} = \sigma^{\theta_{\mathcal{F}}}$. Thus, by definition, σ is a Σ-development of $\mathcal{F}$ in u. Observe that $\sigma^{\theta_{\mathcal{F}}}$ is finite by theorem 4.21. Thus, by definition of $\mathcal{S}_{\underline{\mathcal{R}}}$, $\sigma^{\theta_{\mathcal{F}}}$ ends in a $\underline{\Sigma}$-normal form. Thus, σ is a complete Σ-development of $\mathcal{F}$ in u. Now it is necessary to show that σ is $\mathcal{R}$-standard. First we show that $\sigma^{\theta_{\mathcal{F}}}$ is $\underline{\mathcal{R}}$-standard, as follows. By induction on i for $0 \leq i < |\sigma^{\theta_{\mathcal{F}}}|$, we show that if Δ is a $\underline{\Sigma}$-redex occurrence in $\sigma^{\theta_{\mathcal{F}}}[i]$, then Δ is non-frozen. For $i = 0$ this is immediate. For $i \geq 1$, we know that for each $\underline{\Sigma}$-redex occurrence Δ in $\sigma^{\theta_{\mathcal{F}}}[i]$ that Δ is a residual of a $\underline{\Sigma}$-redex occurrence Δ' in $\sigma^{\theta_{\mathcal{F}}}[i-1]$. (There are no created redex occurrences in $\underline{\Sigma}$-reduction sequences.) By the definition of $\mathcal{S}_{\underline{\mathcal{R}}}$, we know that Δ' is $\underline{\mathcal{R}}$-after the contracted redex occurrence in $\sigma^{\theta_{\mathcal{F}}}[i-1]$. Thus, Δ is non-frozen. Thus, all contracted redex occurrences in $\sigma^{\theta_{\mathcal{F}}}$ are non-frozen and $\sigma^{\theta_{\mathcal{F}}}$ is $\underline{\mathcal{R}}$-standard. Thus, by lemma 5.3 part 1, σ is $\mathcal{R}$-standard.

What remains to be shown is the uniqueness of σ if $\mathcal{R}$ is good.

Suppose σ' is an $\mathcal{R}$-standard complete Σ-development of $\mathcal{F}$ in u and that $\sigma' \neq \sigma$. Thus, by lemma 5.3 part 1 and the definition of a complete development, we know that $\sigma'^{\theta_{\mathcal{F}}}$ is an $\underline{\mathcal{R}}$-standard $\underline{\Sigma}$-reduction sequence ending in a $\underline{\Sigma}$-normal form, just like $\sigma^{\theta_{\mathcal{F}}}$. We know that $\sigma^{\theta_{\mathcal{F}}} \neq \sigma'^{\theta_{\mathcal{F}}}$. Let the ith step be the first one where $\sigma^{\theta_{\mathcal{F}}}$ differs from $\sigma'^{\theta_{\mathcal{F}}}$. Let Δ be the $\underline{\Sigma}$-redex occurrence in $\sigma^{\theta_{\mathcal{F}}}[i]$ which is contracted in the ith step of $\sigma^{\theta_{\mathcal{F}}}$. Remember that Δ is the $\underline{\mathcal{R}}$-first $\underline{\Sigma}$-redex occurrence in $\sigma^{\theta_{\mathcal{F}}}[i]$. Let Δ' be the $\underline{\Sigma}$-redex occurrence contracted in the ith step of $\sigma'^{\theta_{\mathcal{F}}}$. We know that $\Delta \neq \Delta'$ and that $\Delta <_{\underline{\mathcal{R}}((\sigma^{\theta_{\mathcal{F}}})[i])} \Delta'$. Thus, Δ is freezing in $\sigma'^{\theta_{\mathcal{F}}}[i]$. By Lemma 5.3 part 2, it holds that $\underline{\mathcal{R}}$ is good for $\underline{\Sigma}$. Because $\sigma'^{\theta_{\mathcal{F}}}$ is $\underline{\mathcal{R}}$-standard, by lemma 5.2 it holds that Δ has a frozen residual in the final term of $\sigma'^{\theta_{\mathcal{F}}}$, which is therefore not in $\underline{\Sigma}$-normal form. Thus σ' is not a complete development, a contradiction. □

Lemma 5.5. *Let $\mathcal{R}$ be a good Σ-redex ordering function for orthogonal CRS Σ. Let σ be a 2-step non-$\mathcal{R}$-standard Σ-reduction sequence. Then σ is a complete development.* □

Proof. Let $\sigma = \langle u_0, \Delta_0, u_1, \Delta_1, u_2\rangle$. Because σ is not $\mathcal{R}$-standard, the redex occurrence Δ_1 must be a residual of a freezing redex occurrence Δ_1' in u_0. Because $\mathcal{R}$ is good, Δ_1 must be the sole residual in u_1 of Δ_1'. Thus, u_2 has no residuals of the redex family $\mathcal{F} = \{\Delta_0, \Delta_1'\}$ in u_0 and all redex occurrences contracted in σ are residuals of redex occurrence in $\mathcal{F}$. Thus, σ is a complete development of $\mathcal{F}$ in u_0. □

$\Longrightarrow$-Reduction With respect to an orthogonal CRS Σ and a good Σ-redex ordering function $\mathcal{R}$, let $\Longrightarrow$ be the smallest relation such that:

> If $\sigma_1 \star \sigma \star \sigma_2$ is a Σ-reduction sequence, σ is a 2-step non-$\mathcal{R}$-standard reduction sequence, which by lemma 5.5 is a complete development of some redex family $\mathcal{F}$, and σ' is the unique (by lemma 5.4) $\mathcal{R}$-standard complete Σ-development of $\mathcal{F}$ such that $\sigma \sim \sigma'$, then $\sigma_1 \star \sigma \star \sigma_2 \Longrightarrow \sigma_1 \star \sigma' \star \sigma_2$.

We will use the terminology of reduction when discussing $\Longrightarrow$.

[9] These are 1-step (a.k.a. sequential) strategies. We do not consider multi-step (a.k.a. parallel) strategies.

Lemma 5.6. *Let Σ be an orthogonal CRS. Let σ be a complete Σ-development of a redex family $\mathcal{F}$ in a term u. Let θ be an adequate labelling for σ. Then there is a Σ^{HW}-redex family $\mathcal{F}'$ in u^θ such that σ^θ is a complete Σ^{HW}-development of $\mathcal{F}'$. Furthermore, for any other complete Σ-development σ' of $\mathcal{F}$ in u, it holds that ${\sigma'}^\theta$ is a complete Σ^{HW}-development of $\mathcal{F}'$.* □

Proof. Let $\mathcal{F}'$ be the Σ^{HW}-redex family in u^θ such that $\|\mathcal{F}'\| = \mathcal{F}$. By the definition of complete development and lemma 4.12, we know that every redex occurrence contracted in σ^θ is a residual of a redex occurrence in $\mathcal{F}'$ and that $\mathcal{F}'$ has no residuals in the final term of σ^θ. Thus, σ^θ is a complete development of $\mathcal{F}'$. Now let σ' be another complete Σ-development of $\mathcal{F}$ in u and consider ${\sigma'}^\theta$. By lemma 4.12 we know that every step in ${\sigma'}^\theta$ contracts a redex occurrence which is a residual of a redex occurrence $\Delta \in \mathcal{F}'$. By lemma 4.13, we know that the contracted redex occurrence has the same degree as Δ. A residual of Δ with the same degree is contracted in σ^θ, so we know the degree is greater than 0. Thus, ${\sigma'}^\theta$ is a valid Σ^{HW}-reduction sequence. We know it is a complete development of $\mathcal{F}'$ for the same reasons that applied to σ^θ. □

Lemma 5.7. *With respect to an orthogonal CRS Σ and a good Σ-redex ordering function $\mathcal{R}$, if σ is a Σ-reduction sequence with adequate labelling θ and $\sigma \Longrightarrow \sigma'$, then θ is adequate for σ'.* □

Proof. We know that $\sigma = \sigma_1 \star \hat{\sigma} \star \sigma_2 \Longrightarrow \sigma_1 \star \hat{\sigma}' \star \sigma_2 = \sigma'$ where $\hat{\sigma}$ and $\hat{\sigma}'$ are complete developments of some redex family $\mathcal{F}$ such that $\hat{\sigma} \sim \hat{\sigma}'$. We also know that σ^θ is a valid Σ^{HW}-reduction sequence and we wish to show that ${\sigma'}^\theta$ is also a valid Σ^{HW}-reduction sequence. It holds that $\sigma^\theta = {\sigma_1}^\theta \star \hat{\sigma}^{\theta_1} \star {\sigma_2}^{\theta_2}$ for some labellings θ_1 and θ_2. By lemma 5.6 and theorem 4.23, we know that $\hat{\sigma}^{\theta_1} \sim (\hat{\sigma}')^{\theta_1}$ and that $(\hat{\sigma}')^{\theta_1}$ is a valid Σ^{HW}-reduction sequence. We then obtain that ${\sigma'}^\theta = {\sigma_1}^\theta \star (\hat{\sigma}')^{\theta_1} \star {\sigma_2}^{\theta_2}$ is a valid Σ^{HW}-reduction sequence, which is the desired result. □

Lemma 5.8 (König). *Let binary relation $\Re$ be finitely branching. An $\Re$-sequence is a sequence $\chi = \langle a_1, a_2, a_3, \ldots \rangle$ where $\Re(a_i, a_{i+1})$ for $1 \leq i < |\chi|$. Suppose there does not exist an upper bound k on the maximum length of $\Re$-sequences. Then $\Re$ is not well founded, i.e., there is an infinite $\Re$-sequence.* □

Theorem 5.9. *With respect to an orthogonal CRS Σ and a good Σ-redex ordering function $\mathcal{R}$, $\Longrightarrow$-reduction on finite Σ-reduction sequences is strongly normalizing.* □

Proof. Let σ be a Σ-reduction sequence. Let θ be an adequate labelling for σ. Let k be the length of the longest Σ^{HW}-reduction sequence beginning with $\sigma[0]^\theta$. We know k exists by theorem 4.26 and König's lemma. By lemma 5.7, we know that if $\sigma \Longrightarrow \sigma'$, then θ is adequate for σ'. Thus, if $\sigma \Longrightarrow\!\!\!\!\Longrightarrow \sigma'$, then σ' is no longer than k steps. Let f compute a metric on reduction sequences as follows: $f(\sigma) = \langle g_0, \ldots, g_{|\sigma|-1} \rangle$ where g_i is the number of Σ-redex occurrences $\mathcal{R}$-before the redex occurrence contracted in step i of σ. Observe that if $\sigma' \Longrightarrow \sigma''$ then $f(\sigma) >_{\mathrm{lex}} f(\sigma')$. Observe also that $>_{\mathrm{lex}}$ is well founded (i.e., has no infinite descending chains) on sequences whose length is bounded by a fixed length k. (This can be shown by the same method used to show that the multiset extension of a well founded ordering is well founded, a result proved in [DM79].) Thus, any $\Longrightarrow$-reduction sequence beginning from any σ must terminate, i.e., $\Longrightarrow$-reduction is strongly normalizing. □

Theorem 5.10. *With respect to an orthogonal CRS Σ and a good Σ-redex ordering function $\mathcal{R}$, a Σ-reduction sequence σ is a $\Longrightarrow$-normal form iff σ is $\mathcal{R}$-standard.* □

Proof. The "if" direction is easy: if σ is $\mathcal{R}$-standard, then σ has no non-$\mathcal{R}$-standard 2-step subsequences and is therefore a $\Longrightarrow$-normal form. This leaves the "only if" direction. Let σ be a $\Longrightarrow$-normal form.

We prove the claim when σ is finite by induction on $|\sigma|$.

1. Suppose $0 \leq |\sigma| \leq 1$. Then σ can not be non-standard.
2. Suppose $|\sigma| = 2$. Suppose σ is non-$\mathcal{R}$-standard. Then σ has a 2-step subsequence (itself) which is non-$\mathcal{R}$-standard. Thus, it is not a $\Longrightarrow$-normal form, a contradiction.

3. Suppose $|\sigma| \geq 3$. By the induction hypothesis, both $\sigma[0..|\sigma|-1]$ and $\sigma[1..|\sigma|]$ are $\mathcal{R}$-standard. Suppose σ is non-$\mathcal{R}$-standard. Let the kth step be non-$\mathcal{R}$-standard. If $k < |\sigma| - 1$, then $\sigma[0..|\sigma| - 1]$ would be non-$\mathcal{R}$-standard, so it must be the case that $k = |\sigma| - 1$. For $0 \leq i < |\sigma|$ let Δ_i be the redex occurrence contracted in step i of σ. Thus, the redex occurrence Δ_k must be frozen, i.e., Δ_k must be a residual of a freezing redex occurrence Δ in $\sigma[j]$ for some $j < k$. If $j > 0$, then $\sigma[1..|\sigma|]$ would be non-$\mathcal{R}$-standard, so it must be the case that $j = 0$. Because $\mathcal{R}$ is good, we know that the freezing redex occurrence Δ in $\sigma[0]$ must have exactly one residual Δ' in $\sigma[1]$, and Δ' must be frozen. Because step 1 (the initial step is step 0) is $\mathcal{R}$-standard and $\mathcal{R}$ is good, the frozen redex occurrence Δ' must be before the contracted redex occurrence Δ_1. Thus, Δ' is also freezing. Because Δ' is the sole residual in $\sigma[1]$ of Δ, we know that Δ' also has Δ_k as a residual. Thus, Δ_k is a residual of a freezing redex occurrence in $\sigma[1]$, contradicting that $\sigma[1..|\sigma|]$ is $\mathcal{R}$-standard.

For infinite σ, we observe that if σ is non-$\mathcal{R}$-standard, then there is some finite prefix $\sigma[0..j]$ that is non-$\mathcal{R}$-standard. Then, because $\sigma[0..j]$ is a $\Longrightarrow$-normal form, we know that $\sigma[0..j]$ is $\mathcal{R}$-standard. Thus, if infinite σ is a $\Longrightarrow$-normal form, it follows that σ is $\mathcal{R}$-standard. □

Theorem 5.11 (Standardization). *Let Σ be an orthogonal CRS and $\mathcal{R}$ a good Σ-redex ordering function. For any finite Σ-reduction sequence σ, there is an $\mathcal{R}$-standard Σ-reduction sequence σ' such that $\sigma \sim \sigma'$.* □

Proof. By theorems 5.9 and 5.10. □

Theorem 5.12 (Normalization). *Let Σ be an orthogonal CRS and $\mathcal{R}$ a good Σ-redex ordering function. Then the $\mathcal{R}$-first reduction strategy $\mathcal{S}_{\mathcal{R}}$ is normalizing for Σ.* □

Proof. First, we claim for any $\mathcal{R}$-standard Σ-reduction sequence σ ending in a Σ-normal form that $\sigma = \mathcal{S}^*_{\mathcal{R}}(\sigma[0])$. We prove the claim by induction on the length of σ.

1. Suppose $|\sigma| = 0$. Then $\Sigma\text{-nf}(\sigma[0])$ and $\mathcal{S}^*_{\mathcal{R}}(\sigma[0]) = \langle\sigma[0]\rangle = \sigma$.

2. Suppose $|\sigma| \geq 1$. Let $\Delta = \mathcal{S}_{\mathcal{R}}(\sigma[0])$ (the $\mathcal{R}$-first Σ-redex occurrence in $\sigma[0]$). Suppose σ does not begin by contracting Δ. Because σ is $\mathcal{R}$-standard, by lemma 5.2, Δ has a residual in all subsequent terms in σ. However, this contradicts that σ ends in a Σ-normal form. Thus, both $\mathcal{S}^*_{\mathcal{R}}(\sigma[0])$ and σ begin by contracting Δ. Let $\sigma[0..1] = \langle\sigma[0], \Delta, \sigma[1]\rangle$. So $\mathcal{S}^*_{\mathcal{R}}(\sigma[0]) = \sigma[0..1] \star \mathcal{S}^*_{\mathcal{R}}(\sigma[1])$. By induction hypothesis, $\sigma[1..] = \mathcal{S}^*_{\mathcal{R}}(\sigma[1])$. This implies that $\sigma = \mathcal{S}^*_{\mathcal{R}}(\sigma[0])$.

We now use the above claim. Suppose $u \xrightarrow{\text{nf}}_{\Sigma} v$. Then there is some Σ-reduction sequence $\sigma = u \longrightarrow\!\!\!\!\!\rightarrow v$. By theorem 5.11, there exists a $\mathcal{R}$-standard Σ-reduction sequence σ' such that $\sigma \sim \sigma'$. By the claim proven above, $\sigma' = \mathcal{S}^*_{\mathcal{R}}(u)$. Thus, $\mathcal{S}^*_{\mathcal{R}}(u)$ ends with v. Thus, $\mathcal{S}_{\mathcal{R}}$ is a normalizing reduction strategy. □

6 Redex Orderings via Subterm Orderings

In order to make use of the results of section 5, this section defines a method for finding good redex ordering functions. In particular, this section (1) shows how to derive redex ordering functions from subterm ordering functions and (2) defines conditions that are sufficient to guarantee that such functions are good and (3) presents a "generic" generator of subterm ordering functions. The methods developed here are general enough for the applications in the companion paper [MW00]. There seems to be a close connection between the material here and the notion of *strong sequentiality* [HL91b], but we have not formally verified this.

Subterm Ordering Function Let Σ be a CRS and let $u \in \text{Ter}(\Sigma)$. A Σ-*subterm occurrence ordering* of u is a sequence $\gamma = [\vec{p}]$ of members of $\text{Skel}(u)$. When Σ is clear from context, we may abbreviate "Σ-subterm occurrence ordering" by "subterm ordering". A subterm ordering γ is *complete* iff $\gamma = \text{Skel}(u)$. A *subterm occurrence ordering function* for Σ is a function Γ such that for all $u \in \text{Ter}(\Sigma)$, $\Gamma(u)$ is a complete subterm ordering for u.

REMARK 6.1. Some subterm ordering functions can be seen as strategies for traversing trees of terms which visit the nodes of a tree in some particular order, e.g., preorder traversal. For practical purposes, we are interested in subterm ordering functions that (1) visit the root first, (2) always visit a child of a previously visited node, and (3) decide which node to visit next using only information on the previously visited nodes. □

Redex Orderings from Subterm Orderings Given a CRS Σ with $u \in \mathrm{Ter}(\Sigma)$, a subterm ordering γ for u determines a redex ordering for u, notation $\lfloor\gamma\rfloor_\Sigma$ (written $\lfloor\gamma\rfloor$ when Σ is obvious), such that if there are Σ-redex occurrences $\Delta = (u.p, r)$ and $\Delta' = (u.p', r')$, then $\Delta \prec_{\lfloor\gamma\rfloor} \Delta'$ iff $p \prec_\gamma p'$. It is clear that if γ is complete then $\lfloor\gamma\rfloor$ is too. If Γ is a subterm ordering function, then $\lfloor\Gamma\rfloor_\Sigma$ (written $\lfloor\Gamma\rfloor$ when Σ is obvious) is the redex ordering function $\{\, u \mapsto \lfloor\Gamma(u)\rfloor \mid u \in \mathrm{Ter}(\Sigma) \,\}$. A subterm ordering function Γ is *good* for Σ iff $\lfloor\Gamma\rfloor$ is a good Σ-redex ordering function.

Let $s \to t \in \mathrm{Red}(\Sigma)$, $u \in \mathrm{Ter}(\Sigma)$, p and $\vec{q}$ be paths. Let the predicate PartialRedexMatch$(s, u, p, [\vec{q}^n])$ hold iff $q_i = p \cdot q_i'$, $q_i' \in \mathrm{Int}(s)$, and $\mathrm{Tree}(u)(q_i) = \mathrm{Tree}(s)(q_i')$ for $1 \le i \le n$.

Sufficient Conditions for Goodness The following conditions on subterm ordering functions will be proven to be sufficient to guarantee that a subterm ordering function Γ is good for an orthogonal CRS Σ.

1. A subterm ordering $\gamma = [\vec{p}^n]$ of u is *top-down* iff for $1 \le i \le n$, if $p_i = q \cdot j$ for some path q and number j, then $q \in \{p_1, ..., p_{i-1}\}$. A subterm ordering function Γ is *top-down* iff for all $u \in \mathrm{DomDef}(\Gamma)$ it holds that $\Gamma(u)$ is a top-down subterm ordering of u.

   Informally, Γ is top-down if it visits a node only after visiting the node's parent.

2. A subterm ordering function Γ is *no-lookahead* iff for all $u, v \in \mathrm{Ter}(\Sigma)$, if $\Gamma(u) = [\vec{p}^m, \vec{q}]$, $\Gamma(v) = [\vec{p}^m, \vec{q'}]$ and $\mathrm{Tree}(u)(p_i) = \mathrm{Tree}(v)(p_i)$ for $1 \le i \le m$, then $q_1 = q_1'$.

   Informally, Γ is no-lookahead iff it only uses information from nodes already visited to decide which node to visit next.

3. A subterm ordering $\gamma = [\vec{p}^n]$ of u is *Σ-redex-directed* (redex directed) iff whenever $s \to t \in \mathrm{Red}(\Sigma)$, $1 \le i < n$, $1 \le j < |\mathrm{Int}(s)|$, and PartialRedexMatch$(s, u, p_i, [p_i, \ldots, p_{i+j-1}])$, then $p_{i+j} = p_i \cdot q$ for some $q \in \mathrm{Int}(s)$. A subterm ordering function Γ is *Σ-redex-directed* (redex-directed) iff for all $u \in \mathrm{Ter}(\Sigma)$, $\Gamma(u)$ is a redex-directed subterm ordering of u.

   Informally, thinking of $\Gamma(u)$ as a traversal of the nodes of $\mathrm{Tree}(u)$ in a particular order, whenever a redex-directed traversal has seen part of the LHS of some reduction rule $r \in \mathrm{Red}(\Sigma)$, the next node visited contributes to determining whether the LHS of r indeed matches the term.

REMARK 6.2. A Σ-redex ordering function $\mathcal{R}$ is *top-down* iff for every pair of Σ-redex occurrences $\Delta = (u.p, r)$ and $\Delta' = (u.p', r')$ in term u, if $p \le p'$ then $\Delta \prec_{\mathcal{R}(u)} \Delta'$. Observe that if Γ is a top-down subterm ordering function, then $\lfloor\Gamma\rfloor_\Sigma$ is a top-down Σ-redex ordering function. □

Lemma 6.3. *Let Σ be a CRS and let $u \xrightarrow{\Delta}_\Sigma v$ where $\Delta = (u.p, r)$. Let Γ be a top-down and no-lookahead subterm ordering function and let $\mathcal{R} = \lfloor\Gamma\rfloor$. Then*

1. *If $\Gamma(u) = [\vec{p}^n, \vec{q}]$ where $p = p_n$, then $\Gamma(v) = [\vec{p}^n, \vec{q'}]$ and for $1 \le i < n$, $\mathrm{Tree}(u)(p_i) = \mathrm{Tree}(v)(p_i)$.*
2. *If $\Delta' = (v.p', r')$ is an $\mathcal{R}$-frozen redex occurrence in v, then $p' \prec_{\Gamma(v)} p$.* □

Proof.

1. Because Γ is top-down, $p = p_n \not\le p_i$ for $1 \le i < n$. The definition of CRS's then implies $\mathrm{Tree}(u)(p_i) = \mathrm{Tree}(v)(p_i)$ for $1 \le i < n$, because none of the positions $\vec{p}^{n-1}$ can be affected by the contraction of Δ. Let $\Gamma(v) = [\vec{\hat{q}}]$. We now prove by induction on i for $1 \le i \le n$ that $\hat{q}_i = p_i$. If $i = 1$, then this follows from the fact that $p_1 = \hat{q}_1 = \epsilon$ which holds because Γ is top-down. Suppose $i \ge 2$. By the induction hypothesis it holds that $[\vec{p}^{i-1}] = [\vec{\hat{q}}^{i-1}]$. Then $p_i = \hat{q}_i$ follows because Γ has the no-lookahead property.

2. By the definition of frozen, Δ' is a residual of a freezing redex occurrence $\Delta'' = (u.p'', r')$ in u. By the definitions of freezing $\Delta'' \prec_{\mathcal{R}(u)} \Delta$ and therefore $p'' \prec_{\Gamma(u)} p$. That is, $\Gamma(u) = [\vec{p}^n, \vec{q}]$ where $p = p_n$ and $p'' \in \{p_1, \ldots, p_{n-1}\}$. The result then follows by part (1) of this lemma. □

Theorem 6.4. *Let Σ be an orthogonal CRS and let Γ be a top-down, no-lookahead, and redex-directed subterm ordering function. Then Γ is good for Σ.* □

Proof. Let $\mathcal{R} = \lfloor\Gamma\rfloor$ and consider the reduction step $u \xrightarrow{\Delta}_\Sigma v$ with $\Delta = (u.p, r)$. Let $\gamma = \Gamma(u)$. Let $r' \in \mathrm{Red}(\Sigma)$. We must show that (1) every $\mathcal{R}$-freezing redex occurrence in u has exactly one residual in v and (2) that every $\mathcal{R}$-frozen redex occurrence in v occurs before every non-frozen redex occurrence in v. We establish 1 and 2 separately.

1. Suppose $\Delta' = (u.p', r')$ is a Σ-redex occurrence in u such that Δ' does not have exactly one residual in v. Because Σ is orthogonal, it must hold that $p < p'$. Then, because Γ is top-down, $\Delta \preceq_{\lfloor\gamma\rfloor} \Delta'$ and $p \preceq_\gamma p'$. Therefore, by definition, Δ' is not freezing.

2. Suppose $\Delta' = (v.p', r')$ and $\Delta'' = (v.p'', r')$ are Σ-redex occurrences in v such that Δ' is non-frozen, Δ'' is frozen and $\Delta' \prec_{\lfloor\Gamma(v)\rfloor} \Delta''$. By lemma 6.3 (2), it holds that $p'' \prec_{\Gamma(v)} p$. Because $\Delta' \prec_{\lfloor\Gamma(v)\rfloor} \Delta''$, it holds that $p' \prec_{\Gamma(v)} p''$. Therefore, by the transitivity of $\prec_{\Gamma(v)}$, and the non-frozeness of Δ', it must hold that Δ' is a created redex occurrence. Because Γ is top-down, $p \not\leq p'$. Because Δ' is created and $p \not\leq p'$, it must hold that $p' < p$. Specifically, it must hold that $p = p' \cdot q$ for some $q \in \mathrm{Int}(r')$. Then, because Γ is Σ-redex-directed and $p' \prec_{\Gamma(v)} p'' \prec_{\Gamma(v)} p$, it must hold that $p'' = p' \cdot q'$ where $q' \in \mathrm{Int}(r')$. Thus, Δ' and Δ'' overlap, contradicting the premise that Σ is an orthogonal CRS. □

$\mathcal{G}(\Sigma, \Theta)(s) = \mathrm{NextOrder}^k_{\Sigma,\Theta,s}([])$ where $k = |\mathrm{Skel}(s)|$ and

$$
\begin{aligned}
\mathrm{NextOrder}_{\Sigma,\Theta,s}(\delta) &= \begin{cases} [\vec{p}^i, p_{i+1}] & \text{if } \delta = [\vec{p}^i] \neq \mathrm{wrong} \text{ and } \mathrm{NextPos}_{\Sigma,\Theta}([\vec{p}^i], s) = p_{i+1} \neq \mathrm{wrong}, \\ \mathrm{wrong} & \text{otherwise}, \end{cases} \\
\mathrm{NextPos}_{\Sigma,\Theta}(\gamma, s) &= \begin{cases} \Theta(\mathrm{Opt}_\Sigma(\gamma, s), \mathrm{Tree}(s) \downarrow \gamma) & \text{if } \mathrm{Opt}_\Sigma(\gamma, s) \neq \mathrm{wrong}, \\ \mathrm{wrong} & \text{otherwise}, \end{cases} \\
\mathrm{Unex}(\gamma, s) &= \min_{\leq}(\mathrm{Skel}(s) \setminus \gamma) \\
\mathrm{Disc}_\Sigma(\gamma, s) &= \{\, (s.p, r) \mid r \in \mathrm{Red}(\Sigma), \forall q \in \mathrm{Int}(r). \\
&\qquad\qquad p \cdot q \in \gamma \text{ and } \mathrm{Tree}(s)(p \cdot q) = \mathrm{Tree}(\mathrm{LHS}(r))(q) \,\} \\
\mathrm{Inv}_\Sigma(\gamma, s) &= \{\, p \mid (s.q, r) \in \mathrm{Disc}_\Sigma(\gamma, s), q' \in \mathrm{Int}(r), p \leq q \cdot q' \,\} \\
\mathrm{Poss}_\Sigma(\gamma, s) &= \{\, (s.p, s' \to t) \mid p \in \gamma \setminus \mathrm{Inv}_\Sigma(\gamma, s), s' \to t \in \mathrm{Red}(\Sigma), \\
&\qquad\qquad \forall q \in \mathrm{Int}(s'). \, p \cdot q \in \gamma \Rightarrow \mathrm{Tree}(s)(p \cdot q) = \mathrm{Tree}(s')(q) \,\} \\
\mathrm{PossUnex}(s.p, r) &= \{\, p \cdot q \mid q \in \mathrm{Int}(r), p \cdot q \in \mathrm{Unex}(\gamma, s) \,\} \\
\mathrm{Mand}_\Sigma(\gamma, s) &= \bigcap_{(s.p,r) \in \mathrm{Poss}_\Sigma(\gamma,s)} \mathrm{PossUnex}(s.p, r) \\
\mathrm{Opt}_\Sigma(\gamma, s) &= \begin{cases} \mathrm{Unex}(\gamma, s) & \text{if } \mathrm{Poss}_\Sigma(\gamma, s) = \varnothing, \\ \mathrm{Mand}_\Sigma(\gamma, s) & \text{if } \mathrm{Poss}_\Sigma(\gamma, s) \neq \varnothing \text{ and } \mathrm{Mand}_\Sigma(\gamma, s) \neq \varnothing, \\ \mathrm{wrong} & \text{if } \mathrm{Poss}_\Sigma(\gamma, s) \neq \varnothing \text{ and } \mathrm{Mand}_\Sigma(\gamma, s) = \varnothing \end{cases}
\end{aligned}
$$

Figure 1: A generic subterm ordering function generator.

A Subterm Ordering Function Generator Figure 1 defines a subterm ordering function generator $\mathcal{G}$. The generator $\mathcal{G}$ can be applied to any CRS Σ and choice function Θ. The choice function Θ may be any fixed total function such that $\Theta(\{\vec{p}\}, \varphi) \in \{\vec{p}\} \cup \{\mathrm{wrong}\}$. The second argument to Θ is extra information that it may use in deciding which member of its first argument to return. If no choice function is specified as in $\mathcal{G}(\Sigma)$, then this stands for $\mathcal{G}(\Sigma, \Theta_{\mathrm{lex}})$ where Θ_{lex} is the choice function such that $\Theta_{\mathrm{lex}}(\{\vec{p}\}, \varphi) = \min_{\mathrm{lex}}\{\vec{p}\}$. In using the various functions defined in figure 1, the subscripts of Σ and Θ will sometimes be omitted when the CRS and choice function being used are clear from the surrounding text.

Intuitively, the meanings of the functions NextOrder, NextPos, Unex, Disc, Inv, Poss, PossUnex, Mand, and Opt are as follows. The function NextOrder either extends a subterm ordering by exploring one additional position in the term or propagates the failure symbol "wrong". Given a term u and a subterm ordering on some subset of $\mathrm{Skel}(u)$, The function NextPos determines the next position to explore in the term, if possible. The function Unex gives the "unexplored frontier positions", Disc the "discovered redex positions",[10] Inv the "invalid positions for undiscovered redexes", Poss the "possible redex occurrences", PossUnex the "unexplored positions on the frontier of a possible redex occurrence", Mand the "must-explore-next positions", and Opt the "options for next position to explore".

Lemma 6.5. *Let Σ be a CRS. Then $\mathcal{G}(\Sigma)$ is a total function on* $\mathrm{Ter}(\Sigma)$*, always returning either a subterm ordering for its input or* wrong. □

Proof. Obvious by inspection of the definition of $\mathcal{G}$. □

EXAMPLE 6.6. Consider the non-strongly-sequential CRS Σ_{bad} [HL91b] with $\mathrm{Red}(\Sigma_{\mathrm{bad}}) = \{r_1, r_2, r_3\}$ where

$$\begin{aligned} r_1 &= F(G,H,Z) \to I, \\ r_2 &= F(Z,G,H) \to I, \\ r_3 &= F(H,Z,G) \to I. \end{aligned}$$

Note that Σ_{bad} is an orthogonal CRS with $\mathrm{Int}(r_1) = \{\epsilon, 1, 2\}$, $\mathrm{Int}(r_2) = \{\epsilon, 2, 3\}$, and $\mathrm{Int}(r_3) = \{\epsilon, 1, 3\}$. Let $u = F(u_1, u_2, u_3) \in \mathrm{Ter}(\Sigma_{\mathrm{bad}})$. Then $\mathrm{Unex}([\epsilon], u) = \{1, 2, 3\}$ and $\mathrm{Poss}([\epsilon], u) = \{(u.\epsilon, r_1), (u.\epsilon, r_2), (u.\epsilon, r_3)\}$. However, $\mathrm{Mand}([\epsilon], u) = \{1,2\} \cap \{2,3\} \cap \{1,3\} = \varnothing$, i.e., $\mathcal{G}(\Sigma_{\mathrm{bad}})(u) =$ wrong so $\mathcal{G}(\Sigma_{\mathrm{bad}})$ is not a good subterm ordering function. □

REMARK 6.7. The first redex occurrence picked by $\mathcal{G}(\Sigma)$ is always the same as the redex occurrence picked by the algorithm of Huet and Lévy for finding a strongly needed redex occurrence [HL91b]. Thus, we conjecture that $\mathcal{G}(\Sigma)$ is total iff Σ is strongly sequential. (Of course, we must first extend the notion of strong sequentiality to CRS's in the natural way.) □

Theorem 6.8. *If Σ is orthogonal and $\Gamma = \mathcal{G}(\Sigma)$ is a subterm ordering function for Σ (i.e.,* wrong $\notin \Gamma(\mathrm{Ter}(\Sigma))$*), then Γ is good for Σ.* □

Proof. By lemma 6.5 and theorem 6.4, it suffices to show that Γ is top-down, no-lookahead and Σ-redex-directed. The generator $\mathcal{G}$ was specifically designed in order to yield subterm ordering functions satisfying these conditions. That Γ is top-down follows from the fact that it starts at the root and in each subsequent iteration, the next node is selected from the unexplored children of already selected nodes. That Γ is no-lookahead follows from the fact that the choice function Θ is constrained to be a fixed function that selects the next node from its first argument using only the nodes considered so far. That Γ is Σ-redex-directed follows from the fact that it always selects a node that is an internal position of the LHS of a reduction rule if the LHS of the rule partially (but incompletely) matches nodes in the tree. □

7 Evaluation

In this section we show how the subterm ordering function generator introduced in the preceding section can be specialized for the class of CRSs known as *constructor systems*. These are the CRSs that one would expect to arise in conjunction with programming language semantics. The subterm ordering function is specialized in that it has been provided a choice function that is biased to fully explore the internal positions of values before considering other nodes. We call such a choice function a "value respecting" choice function. This section also shows how to derive the sets of *values* and *evaluation contexts* directly from the reduction rules of the CRS. These are then used to further define an *evaluation relation* for the CRS.

We say that a CRS Σ is *manageable* if and only if it is orthogonal and the specialization of $\mathcal{G}$ for Σ is a well-defined subterm ordering function. In theorem 7.6 we show that if Σ is a manageable CRS and $u \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v$ then there exists a value v' such that u evaluates to v' and v' reduces to v. We call this property Plotkin-Wadsworth-Felleisen standardization.

[10] If we allowed non-fully-extended reduction rules, we would have to call the result of Disc the "discovered redex-like patterns".

Context/Term Decomposition The *context/term decomposition* of a preterm w at a set of positions P such that $P \cap \mathrm{Skel}(w) \neq \varnothing$, written $\mathrm{Decomp}(w, P)$, is defined as follows:

$$(C^{(n)}, \langle \vec{w}^n \rangle) \in \mathrm{Decomp}(w, P) \iff \begin{cases} C[\vec{w}] \equiv w \\ \text{and } \mathrm{Skel}(C) = \{\, q \in \mathrm{Skel}(w) \mid \nexists p \in P.\, p < q \,\} \\ \text{and } \mathrm{Tree}(C)(P \cap \mathrm{Skel}(C)) = \{\Box\} \end{cases}$$

The set $\mathrm{Decomp}(w, P)$ will contain an infinite number of context/subterm decompositions iff at least one position $p \in \mathrm{Skel}(w) \cap P$ is below a binder in w.

Value Patterns and Values A pattern s *subsumes* a pattern t, written $s \sqsubseteq t$, iff $\mathrm{Tree}(s)(p) = \mathrm{Tree}(t)(p)$ for all $p \in \mathrm{Int}(s)$. Given constructor CRS Σ, define its sets of *value patterns* and *values* as follows:

$$\begin{aligned} \mathrm{ValPatt}(\Sigma) &= \min\nolimits_{\sqsubseteq}\{\, s_i \mid s \to t \in \mathrm{Red}(\Sigma), s \equiv F(\vec{s}^n), 1 \leq i \leq n \,\} \\ \mathrm{Val}(\Sigma) &= \{\, \nu(s) \mid s \in \mathrm{ValPatt}(\Sigma),\ \nu \text{ is a valuation for } s \,\} \end{aligned}$$

Value-Respecting Choice Function Given a constructor CRS Σ, define the following:

$$\begin{aligned} \mathrm{PossVal}_\Sigma(f) &= \{s \in \mathrm{ValPatt}(\Sigma) \mid \exists p' \in \mathrm{Int}(s).f(p') \text{ is undefined}, \\ &\qquad \forall p \in \mathrm{Int}(s).f(p) \text{ defined } \Rightarrow f(p) = \mathrm{Tree}(s)(p)\}, \\ \mathrm{MandVal}_\Sigma(P, f) &= P \cap (\textstyle\bigcap_{s \in \mathrm{PossVal}_\Sigma(f)} \mathrm{Int}(s)), \\ \mathrm{ValChoose}(\Sigma)(P, f) &= \begin{cases} \min_{\mathrm{lex}}(P) & \text{if } \mathrm{PossVal}_\Sigma(f) = \varnothing, \\ \min_{\mathrm{lex}}(\mathrm{MandVal}_\Sigma(P, f)) & \text{if } \mathrm{PossVal}_\Sigma(f) \neq \varnothing \text{ and } \mathrm{MandVal}_\Sigma(P, f) \neq \varnothing, \\ \mathrm{wrong} & \text{if } \mathrm{PossVal}_\Sigma(f) \neq \varnothing \text{ and } \mathrm{MandVal}_\Sigma(P, f) = \varnothing. \end{cases} \\ \mathrm{ValOrder}(\Sigma) &= \mathcal{G}(\Sigma, \mathrm{ValChoose}(\Sigma)) \end{aligned}$$

Given that f represents a partial top-down exploration of some term u, the meaning of $\mathrm{PossVal}_\Sigma(f)$, is that a value pattern $s \in \mathrm{PossVal}_\Sigma(f)$ iff u might be a value by virtue of s matching u, but u has not been explored enough to determine this for certain. The intuitive meaning of $\mathrm{MandVal}_\Sigma(P, f)$ is the set of positions that must be explored. The function $\mathrm{ValChoose}(\Sigma)$ is a "value respecting" choice function, i.e., a choice function that is biased to fully explore the internal positions of values before considering other nodes. $\mathrm{ValOrder}(\Sigma)$ is the subterm ordering function generator specialized with the value respecting choice function.

Let Σ be a constructor CRS, let $u \in \mathrm{Ter}(\Sigma)$, γ be a top-down subterm ordering of u and let $f = \mathrm{Tree}(u) \downarrow \gamma$. Then the predicate $\mathrm{MaybeVal}(f, \Sigma)$ holds if and only if $\mathrm{PossVal}_\Sigma(f) \neq \varnothing$. The predicate $\mathrm{IsVal}(f, \Sigma)$ holds if and only if there exists an $s \in \mathrm{ValPatt}(\Sigma)$, such that s is linear and fully extended, $\mathrm{Int}(s) \subseteq \mathrm{DomDef}(f)$ and $\forall p \in \mathrm{Int}(s)$, $\mathrm{Tree}(s)(p) = f(p)$. We define the predicate $\mathrm{NotVal}(f, \Sigma)$ to hold if and only if neither $\mathrm{MaybeVal}(f, \Sigma)$ nor $\mathrm{IsVal}(f, \Sigma)$ hold.

Manageability A CRS Σ is *manageable* iff Σ is orthogonal and $\mathrm{ValOrder}(\Sigma)$ is a subterm ordering function.

Evaluation Contexts Given a manageable CRS Σ, define:

$$\begin{aligned} \mathrm{CtxtsPos}(s, p, P) &= \{\, C' \mid (C, \chi) \in \mathrm{Decomp}(s, \{p\} \cup P), (C', \chi') \in \mathrm{Decomp}(C[\vec{u}], \{p\}) \,\} \\ \mathrm{ContextsMT}_\Sigma(s) &= \{C \mid \mathrm{ValOrder}(\Sigma)(s) = [\vec{p}^n], 2 \leq i \leq |\mathrm{Int}(s)|, \\ &\qquad C \in \mathrm{CtxtsPos}(s, p_i, \min\nolimits_{\leq}(\{p_{i+1}, \ldots, p_n\} \cup \{\, p \mid \mathrm{Tree}(s)(p) \in \mathrm{MVar} \,\}))\} \\ \mathcal{E}_{\mathrm{red}}(\Sigma) &= \textstyle\bigcup_{s \to t \in \mathrm{Red}(\Sigma)} \mathrm{ContextsMT}_\Sigma(s) \\ \mathcal{E}_{\mathrm{val}}(\Sigma) &= \textstyle\bigcup_{s \in \mathrm{ValPatt}(\Sigma)} \mathrm{ContextsMT}_\Sigma(s) \cup \{\Box\} \\ \mathrm{EvalCont}(\Sigma) &= \{\, C[C_1[\cdots[C_n]\cdots]] \mid C \in \mathcal{E}_{\mathrm{val}}(\Sigma), \vec{C}^n \in \mathcal{E}_{\mathrm{red}}(\Sigma) \,\} \end{aligned}$$

A context $C \in \mathrm{CtxtsPos}(s, p, P)$ iff C is the one-hole context formed from s by chopping s at the positions $\{p\} \cup P$ and filling in the holes at postions P by arbitrary terms (not metaterms), leaving a single hole at p. A context $C \in \mathrm{ContextsMT}_\Sigma(s)$ iff C is a one-hole context formed from s by partially exploring s according to $\mathrm{ValChoose}(\Sigma)$, putting $\Box$ at the next position to explore and replacing all other unexplored positions by arbitrary terms.

Evaluation Given redex occurrence $\Delta = (u.p, r)$, the statement $u \stackrel{\Delta}{\longmapsto}_\Sigma v$ holds iff $u \stackrel{\Delta}{\longrightarrow}_\Sigma v$ where $u \equiv C^p[u']$ and $C \in \mathrm{EvalCont}(\Sigma)$. In this case, we also write $u \stackrel{p}{\longmapsto}_\Sigma v$ and $u \longmapsto_\Sigma v$. Let $\longmapsto\!\!\!\!\rightarrow_\Sigma$ be the transitive, reflexive closure of $\longmapsto_\Sigma$.

Let Σ-eval-nf(u) hold iff there exists no v such that $u \longmapsto_\Sigma v$. The operational semantics for Σ is a function Eval_Σ such that for $u \in \mathrm{Ter}(\Sigma)$,

$$\mathrm{Eval}_\Sigma(u) = \begin{cases} \text{value} & \text{if } \exists v.\ u \longmapsto\!\!\!\!\rightarrow_\Sigma v, v \in \mathrm{Val}(\Sigma), \\ \text{diverges} & \text{if } \nexists v.\ u \longmapsto\!\!\!\!\rightarrow_\Sigma v, \Sigma\text{-eval-nf}(v), \\ \text{stuck} & \text{otherwise.} \end{cases}$$

Two terms u and v are *observationally equivalent*, written $u \simeq_\Sigma v$, iff $\mathrm{Eval}_\Sigma(C[u]) = \mathrm{Eval}_\Sigma(C[v])$ for every context C such that $\{C[u], C[v]\} \subseteq \mathrm{Ter}(\Sigma)$.

A term u is *evaluable* w.r.t. a CRS Σ iff there exists some v such that $u \longmapsto_\Sigma v$. Observe that u is Σ-evaluable iff there exists a rule $s \to t \in \mathrm{Red}(\Sigma)$, an evaluation context $C \in \mathrm{EvalCont}(\Sigma)$, and a valuation ν such that $u \equiv C[\nu(s)]$. A term u is *root-stable* w.r.t. a CRS Σ iff there is no Σ-redex term v such that $u \longrightarrow\!\!\!\!\rightarrow_\Sigma v$.

Lemma 7.1. *Let Σ be manageable. Let $u \longrightarrow\!\!\!\!\rightarrow_\Sigma v$ where $u \notin \mathrm{Val}(\Sigma)$ and $v \in \mathrm{Val}(\Sigma)$. Then $u \equiv C[u']$ where $C \in \mathcal{E}_{\mathrm{val}}(\Sigma)$ and u' is Σ-root-unstable.* □

Proof. Let $v \equiv \nu(s)$ where $s \in \mathrm{ValPatt}(\Sigma)$. Let $[\vec{p}^m] = \mathrm{ValOrder}(\Sigma)(s)$. Let j be the smallest index between 1 and $|\mathrm{Int}(s)|$ such that $\mathrm{Tree}(u)(p_j)$ is defined and $\mathrm{Tree}(u)(p_j) \neq \mathrm{Tree}(s)(p_j)$. Such a j must exist because otherwise $u \in \mathrm{Val}(\Sigma)$. Let $(C, \langle \vec{u}^k \rangle) \in \mathrm{Decomp}(u, \{p_j, \ldots, p_m\})$. By induction (details omitted), $\nu(s) \equiv C[\vec{v}]$ where $u_i \longrightarrow\!\!\!\!\rightarrow_\Sigma v_i$ for $1 \leq i \leq k$. Let the lth hole of C be at p_j. It holds that $\mathrm{Tree}(u_l)(\epsilon) \neq \mathrm{Tree}(s)(p_j) = \mathrm{Tree}(v_l)(\epsilon)$. Thus, in the reduction sequence $u_i \longrightarrow\!\!\!\!\rightarrow_\Sigma v_i$, there must be a step at the root. Hence, u_l is Σ-root-unstable. Let $C' \equiv C[u_1, \ldots, u_{l-1}, \Box, u_{l+1}, \ldots, u_k]$. Observe that $u \equiv C'[u_l]$. It can be checked that $C' \in \mathcal{E}_{\mathrm{val}}(\Sigma)$. □

Lemma 7.2. *Let Σ be manageable. Let u be Σ-root-unstable but not a Σ-redex term. Then $u \equiv C[u']$ where $C \in \mathcal{E}_{\mathrm{red}}(\Sigma)$ and u' is Σ-root-unstable.* □

Proof. Let σ be a reduction sequence $u \longrightarrow\!\!\!\!\rightarrow_\Sigma v$ such that v is the only Σ-redex term in σ. Let $v \equiv \nu(s)$ where $s \to t \in \mathrm{Red}(\Sigma)$. Let $[\vec{p}^m] = \mathrm{ValOrder}(\Sigma)(s)$. Let j be the smallest index between 1 and $|\mathrm{Int}(s)|$ such that $\mathrm{Tree}(u)(p_j)$ is defined and $\mathrm{Tree}(u)(p_j) \neq \mathrm{Tree}(s)(p_j)$. Such a j must exist because otherwise u would be a Σ-redex term. Let $(C, \langle \vec{u}^k \rangle) \in \mathrm{Decomp}(u, \{p_j, \ldots, p_m\})$. By induction (details omitted) using the fact that v is the first term with a redex occurrence at its root, $v \equiv C[\vec{v}]$ where $u_i \longrightarrow\!\!\!\!\rightarrow_\Sigma v_i$ for $1 \leq i \leq k$. Let the lth hole of C be at p_j. It holds that $\mathrm{Tree}(u_l)(\epsilon) \neq \mathrm{Tree}(s)(p_j) = \mathrm{Tree}(v_l)(\epsilon)$. Thus, in the reduction sequence $u_i \longrightarrow\!\!\!\!\rightarrow_\Sigma v_i$, there must be a step at the root. Hence, u_l is Σ-root-unstable. Let $C' \equiv C[u_1, \ldots, u_{l-1}, \Box, u_{l+1}, \ldots, u_k]$. Observe that $u \equiv C'[u_l]$. It can be checked that $C' \in \mathcal{E}_{\mathrm{red}}(\Sigma)$. □

Lemma 7.3. *Let Σ be manageable, $u \longrightarrow\!\!\!\!\rightarrow_\Sigma v$, $v \in \mathrm{Val}(\Sigma)$, and $u \notin \mathrm{Val}(\Sigma)$. Then u is Σ-evaluable.* □

Proof. By lemma 7.1, $u \equiv C[u']$ where $C \in \mathcal{E}_{\mathrm{val}}(\Sigma)$ and u' is root-unstable. By induction on the size of u', we will prove that $u' \equiv C_1[\cdots[C_n[u'']]\cdots]$ for some n where $C_i \in \mathcal{E}_{\mathrm{red}}(\Sigma)$ for $1 \leq i \leq n$ and u'' is a redex term. If u' is a redex term, then by setting $n = 0$ we are done. If u' is not a redex term, then by lemma 7.2, $u' \equiv C_1[\hat{u}]$ where $C_1 \in \mathcal{E}_{\mathrm{red}}(\Sigma)$ and $\hat{u}$ is root-unstable. By induction hypothesis, $\hat{u} \equiv C_2[\cdots[C_n[u'']]\cdots]$ for some n where $C_i \in \mathcal{E}_{\mathrm{red}}(\Sigma)$ for $2 \leq i \leq n$ and u'' is a redex term. Let $C' \equiv C[C_1[\cdots[C_n]\cdots]]$. Observe that $u \equiv C'[u'']$ and that $C' \in \mathrm{EvalCont}(\Sigma)$. Hence, u is evaluable. □

Lemma 7.4. *If Σ is manageable, $\Gamma = \mathrm{ValOrder}(\Sigma)$, $\mathcal{R} = \lfloor \Gamma \rfloor$, and $u \stackrel{\Delta}{\longmapsto}_\Sigma v$, then Δ is the $\mathcal{R}$-first redex occurrence in u.* □

Proof. Let $\Delta = (u.p, r)$. By the definition of evaluation, $u \equiv C^p[u']$ where $C \in \mathrm{EvalCont}(\Sigma)$. Let $C \equiv C'[C_1[\cdots[C_n]\cdots]]$ where $C' \in \mathcal{E}_{\mathrm{val}}(\Sigma)$ and $C_i \in \mathcal{E}_{\mathrm{red}}(\Sigma)$ for $1 \leq i \leq n$.

Let C' be formed from $s \in \mathrm{ValPatt}(\Sigma)$ as in the definition of $\mathcal{E}_{\mathrm{val}}$ as follows. Let $\Gamma(s) = [\vec{\hat{p}}^{\,m}]$ and let the hole in C' be at $\hat{p}_j$. Thus, $C' \in \mathrm{CtxtsPos}(s, \hat{p}_j, \{\hat{p}_{j+1}, \ldots, \hat{p}_m\})$. Observe for $1 \leq i < j$ that $\mathrm{Tree}(s)(\hat{p}_i) = \mathrm{Tree}(u)(\hat{p}_i)$. Let $\Gamma(u) = \gamma = [\vec{p}^{\,m}]$. Because Γ is no-lookahead, this means that the initial subsequence $\vec{p}^{\,j}$ of γ is the same as $\vec{\hat{p}}^{\,j}$.

For $1 \leq i \leq n$, let C_i be formed from the LHS of $s_i \to t_i \in \mathrm{Red}(\Sigma)$ as in the definition of $\mathcal{E}_{\mathrm{red}}$ as follows. Let $\Gamma(s_i) = [\hat{p}_{i,1}, \ldots, \hat{p}_{i,m_i}]$. Let the hole in C_i be at $\hat{p}_{j_i}$. Thus, $C_i \in \mathrm{CtxtsPos}(s_i, \hat{p}_{i,j_i}, \{\hat{p}_{i,j_i+1}, \ldots, \hat{p}_{i,m}\})$. □

Lemma 7.5. *Let Σ be an orthogonal CRS. Let $\mathcal{R}$ be a good top-down Σ-redex ordering function. Let σ be an $\mathcal{R}$-standard Σ-reduction sequence. With respect to σ, let $\Delta = (\sigma[i].p, r)$ be an $\mathcal{R}$-freezing redex occurrence. Then for $q \leq p$ and $i \leq j \leq |\sigma|$, $\mathrm{Tree}(\sigma[i])(q) = \mathrm{Tree}(\sigma[j])(q)$.* □

Proof. By lemma 5.2, Δ has one $\mathcal{R}$-frozen residual in $\sigma[i]$ and all later terms in σ. By the fact that $\mathcal{R}$ is top-down, every contracted redex in $\sigma[i..]$ is a a position p' where $p' \not\leq p$. Thus $\mathrm{Tree}(\sigma[j])(p'')$ for $p'' \leq p$ must remain unchanged. □

Theorem 7.6 (Plotkin-Wadsworth-Felleisen Standardization). *If Σ is manageable, $u \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v$, and $v \in \mathrm{Val}(\Sigma)$, then there exists $v' \in \mathrm{Val}(\Sigma)$ such that $u \longmapsto\!\!\!\!\!\rightarrow_\Sigma v' \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v$.* □

Proof. Let $\Gamma = \mathrm{ValOrder}(\Sigma)$ and let $\mathcal{R} = \lfloor \Gamma \rfloor$. Because Σ is manageableand by theorem 6.8, $\mathcal{R}$ is good for Σ. Thus, by theorem 5.11, there is a $\mathcal{R}$-standard reduction sequence σ of the form $u \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v$. Let j be the least index such that $\sigma[j] \in \mathrm{Val}(\Sigma)$. Let $v' \equiv \sigma[j]$. We now prove by induction on i for $1 \leq i \leq j$ that $u \longmapsto\!\!\!\!\!\rightarrow_\Sigma \sigma[i]$. By cases on i:

1. Suppose $i = 0$. Then $u \longmapsto\!\!\!\!\!\rightarrow_\Sigma \sigma[i]$ in 0 steps.

2. Suppose $i > 0$. By induction hypothesis $u \longmapsto\!\!\!\!\!\rightarrow_\Sigma \sigma[i-1]$. Because $\sigma[i-1]$ is not a value and $\sigma[i-1] \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v'$, by lemma 7.3 $\sigma[i-1]$ is evaluable. Thus, $\sigma[i-1] \overset{\Delta}{\longmapsto}_\Sigma u'$ for some u' and Δ. By lemma 7.4, Δ is the $\mathcal{R}$-first lemma in $\sigma[i-1]$. Let the subsequence $\sigma[i-1..i]$ be the reduction step $\sigma[i-1] \overset{\Delta'}{\longrightarrow}_\Sigma \sigma[i]$. If $\Delta = \Delta'$, then $\sigma[i-1] \longmapsto_\Sigma \sigma[i]$.

   Suppose that $\Delta \neq \Delta'$. Then Δ is a $\mathcal{R}$-freezing redex occurrence. Let $\Delta = (\sigma[i-1].p, r)$ and let $\Delta' = (\sigma[i-1].p', r')$. By lemma 7.5, $\mathrm{Tree}(\sigma[i-1])(\hat{q}) = \mathrm{Tree}(v')(\hat{q})$ for $\hat{q} \leq p$.

   Let $\sigma[i-1] \equiv C^p[\hat{u}]$. Note that $C \in \mathrm{EvalCont}(\Sigma)$. Let $C \equiv C'[C_1[\cdots[C_n]\cdots]]$ where $C' \in \mathcal{E}_{\mathrm{val}}(\Sigma)$ and $\vec{C} \in \mathcal{E}_{\mathrm{red}}(\Sigma)$. Let C' be formed from $s \in \mathrm{ValPatt}(\Sigma)$ as in the definition of $\mathcal{E}_{\mathrm{val}}$ as follows. Let $\Gamma(s) = [\vec{\hat{p}}^{\,m}]$ and let the hole in C' be at $\hat{p}'_j$. Let $C' \in \mathrm{CtxtsPos}(s, \hat{p}_{j'}, \{\hat{p}_{j'+1}, \ldots, \hat{p}_m\})$. Thus, $C' \equiv C''[u_1, \ldots, u_{l-1}, \Box, u_{l+1}, \ldots, u_k]$ where $(C''^{(k)}, \chi) \in \mathrm{Decomp}(s, \{\hat{p}_{j'}, \ldots, \hat{p}_m\})$ and $\hat{p}_{j'}$ is the position of the lth hole in C''.

   Let $\Gamma(\sigma[i-1]) = [\vec{p}]$. Because Γ is top-down and no-lookahead, it can be checked that the initial subsequence $\vec{p}^{\,j'}$ is the same as the initial subsequence $\vec{\hat{p}}^{\,j'}$. Observe that $\mathrm{Tree}(\sigma[i-1])(p_{i'})$ is a constructor for $1 \leq i' < j'$ and is not a constructor for $i' = j'$. By an induction on the steps of $\sigma[i-1..j]$, for $1 \leq i' < j'$ it holds that $\mathrm{Tree}(\sigma[i-1])(p_{i'}) = \mathrm{Tree}(\sigma[i-1])(p_{i'})$.

   Let $v' \equiv \nu(s)$ for $s \in \mathrm{ValPatt}(\Sigma)$. Since $\mathrm{wrong} \notin \mathrm{Ran}(\Gamma)$, s must match the term up to $p_{j'-1}$. Then $p_{j'}$ must be an internal position of s. This is a contradiction.

Thus, $u \longmapsto\!\!\!\!\!\rightarrow_\Sigma v' \longrightarrow\!\!\!\!\!\rightarrow_\Sigma v$, the desired result. □

Appendix A

Figure 2 contains copies of definitions from sections 6 and 7. They are repeated here for ease of reference.

$\mathcal{G}(\Sigma,\Theta)(s) = \mathrm{NextOrder}^{k}_{\Sigma,\Theta,s}([])$ where $k = |\mathrm{Skel}(s)|$ and

$$
\begin{array}{rcl}
\mathrm{NextOrder}_{\Sigma,\Theta,s}(\delta) & = & \begin{cases} [\vec{p}^{\,i}, p_{i+1}] & \text{if } \delta = [\vec{p}^{\,i}] \neq \mathrm{wrong} \text{ and } \mathrm{NextPos}_{\Sigma,\Theta}([\vec{p}^{\,i}], s) = p_{i+1} \neq \mathrm{wrong}, \\ \mathrm{wrong} & \text{otherwise}, \end{cases} \\
\mathrm{NextPos}_{\Sigma,\Theta}(\gamma, s) & = & \begin{cases} \Theta(\mathrm{Opt}_\Sigma(\gamma,s), \mathrm{Tree}(s)\downarrow\gamma) & \text{if } \mathrm{Opt}_\Sigma(\gamma,s) \neq \mathrm{wrong}, \\ \mathrm{wrong} & \text{otherwise}, \end{cases} \\
\mathrm{Unex}(\gamma, s) & = & \min_{\leq}(\mathrm{Skel}(s) \setminus \gamma) \\
\mathrm{Disc}_\Sigma(\gamma, s) & = & \{\, (s.p, r) \mid r \in \mathrm{Red}(\Sigma), \forall q \in \mathrm{Int}(r). \\
 & & \qquad p \cdot q \in \gamma \text{ and } \mathrm{Tree}(s)(p \cdot q) = \mathrm{Tree}(\mathrm{LHS}(r))(q) \,\} \\
\mathrm{Inv}_\Sigma(\gamma, s) & = & \{\, p \mid (s.q, r) \in \mathrm{Disc}_\Sigma(\gamma, s), q' \in \mathrm{Int}(r), p \leq q \cdot q' \,\} \\
\mathrm{Poss}_\Sigma(\gamma, s) & = & \{\, (s.p, s' \to t) \mid p \in \gamma \setminus \mathrm{Inv}_\Sigma(\gamma, s), s' \to t \in \mathrm{Red}(\Sigma), \\
 & & \qquad \forall q \in \mathrm{Int}(s').\ p \cdot q \in \gamma \Rightarrow \mathrm{Tree}(s)(p \cdot q) = \mathrm{Tree}(s')(q) \,\} \\
\mathrm{PossUnex}(s.p, r) & = & \{\, p \cdot q \mid q \in \mathrm{Int}(r), p \cdot q \in \mathrm{Unex}(\gamma, s) \,\} \\
\mathrm{Mand}_\Sigma(\gamma, s) & = & \bigcap_{(s.p,r) \in \mathrm{Poss}_\Sigma(\gamma,s)} \mathrm{PossUnex}(s.p, r) \\
\mathrm{Opt}_\Sigma(\gamma, s) & = & \begin{cases} \mathrm{Unex}(\gamma, s) & \text{if } \mathrm{Poss}_\Sigma(\gamma,s) = \varnothing, \\ \mathrm{Mand}_\Sigma(\gamma, s) & \text{if } \mathrm{Poss}_\Sigma(\gamma,s) \neq \varnothing \text{ and } \mathrm{Mand}_\Sigma(\gamma,s) \neq \varnothing, \\ \mathrm{wrong} & \text{if } \mathrm{Poss}_\Sigma(\gamma,s) \neq \varnothing \text{ and } \mathrm{Mand}_\Sigma(\gamma,s) = \varnothing \end{cases}
\end{array}
$$

$$
(C^{(n)}, \langle \vec{w}^n \rangle) \in \mathrm{Decomp}(w, P) \Longleftrightarrow \begin{cases} C[\vec{w}] \equiv w \\ \text{and } \mathrm{Skel}(C) = \{\, q \in \mathrm{Skel}(w) \mid \nexists p \in P.\, p < q \,\} \\ \text{and } \mathrm{Tree}(C)(P \cap \mathrm{Skel}(C)) = \{\Box\} \end{cases}
$$

$$
\begin{array}{l}
\mathrm{ValPatt}(\Sigma) = \min_{\sqsubseteq}\{\, s_i \mid s \to t \in \mathrm{Red}(\Sigma), s \equiv F(\vec{s}^n), 1 \leq i \leq n \,\} \\
\mathrm{Val}(\Sigma) \quad = \{\, \nu(s) \mid s \in \mathrm{ValPatt}(\Sigma),\ \nu \text{ is a valuation for } s \,\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{CtxtsPos}(s, p, P) & = & \{\, C' \mid (C, \chi) \in \mathrm{Decomp}(s, \{p\} \cup P), (C', \chi') \in \mathrm{Decomp}(C[\vec{u}], \{p\}) \,\} \\
\mathrm{ContextsMT}_\Sigma(s) & = & \{C \mid \ \mathrm{ValOrder}(\Sigma)(s) = [\vec{p}^{\,n}], 2 \leq i \leq |\mathrm{Int}(s)|, \\
 & & \qquad C \in \mathrm{CtxtsPos}(s, p_i, \min_{\leq}(\{p_{i+1}, \ldots, p_n\} \cup \{\, p \mid \mathrm{Tree}(s)(p) \in \mathrm{MVar} \,\}))\} \\
\mathcal{E}_{\mathrm{red}}(\Sigma) & = & \bigcup_{s \to t \in \mathrm{Red}(\Sigma)} \mathrm{ContextsMT}_\Sigma(s) \\
\mathcal{E}_{\mathrm{val}}(\Sigma) & = & \bigcup_{s \in \mathrm{ValPatt}(\Sigma)} \mathrm{ContextsMT}_\Sigma(s) \cup \{\Box\} \\
\mathrm{EvalCont}(\Sigma) & = & \{\, C[C_1[\cdots[C_n]\cdots]] \mid C \in \mathcal{E}_{\mathrm{val}}(\Sigma), \vec{C}^n \in \mathcal{E}_{\mathrm{red}}(\Sigma) \,\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathrm{PossVal}_\Sigma(f) & = & \{s \in \mathrm{ValPatt}(\Sigma) \mid \exists p' \in \mathrm{Int}(s).f(p') \text{ is undefined}, \\
 & & \qquad \forall p \in \mathrm{Int}(s).f(p) \text{ defined } \Rightarrow f(p) = \mathrm{Tree}(s)(p)\}, \\
\mathrm{MandVal}_\Sigma(P, f) & = & P \cap (\textstyle\bigcap_{s \in \mathrm{PossVal}_\Sigma(f)} \mathrm{Int}(s)), \\
\mathrm{ValChoose}(\Sigma)(P, f) & = & \begin{cases} \min_{\mathrm{lex}}(P) & \text{if } \mathrm{PossVal}_\Sigma(f) = \varnothing, \\ \min_{\mathrm{lex}}(\mathrm{MandVal}_\Sigma(P, f)) & \text{if } \mathrm{PossVal}_\Sigma(f) \neq \varnothing \text{ and } \mathrm{MandVal}_\Sigma(P, f) \neq \varnothing, \\ \mathrm{wrong} & \text{if } \mathrm{PossVal}_\Sigma(f) \neq \varnothing \text{ and } \mathrm{MandVal}_\Sigma(P, f) = \varnothing. \end{cases} \\
\mathrm{ValOrder}(\Sigma) & = & \mathcal{G}(\Sigma, \mathrm{ValChoose}(\Sigma))
\end{array}
$$

Figure 2: The generic subterm ordering function generator and related functions.

8 Conclusion

8.1 Acknowledgements

References

[AF97] Zena M. Ariola and Matthias Felleisen. The call-by-need lambda calculus. *J. Funct. Prog.*, 3(7), May 1997.

[ALP96] *Proc. 5th Int'l Conf. Algebraic & Logic Programming*, 1996.

[AM96] Sergio Antoy and Aart Middeldorp. A sequential reduction strategy. *Theor. Comp. Sc.*, 165(1):75–95, 1996.

[Bar84] Hendrik Pieter Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.

[BKKS87] H. P. Barendregt, J. R. Kennaway, Jan Willem Klop, and M. R. Sleep. Needed reduction and spine strategies for the lambda calculus. *Inf. & Comput.*, 75(3):191–231, 1987.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic I*. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1958.

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

[FF89] M. Felleisen and D. P. Friedman. A syntactic theory of sequential state. *Theor. Comp. Sc.*, 69(3):243–287, 1989.

[FH92] Matthias Felleisen and Robert Hieb. The revised report on the syntactic theories of sequential control and state. *Theor. Comp. Sc.*, 102:235–271, 1992.

[GK] John Glauert and Zurab Khasidashvili. Minimal and optimal relative normalization in orthogonal expression reduction systems. Presented at the Sept. 1996 Glasgow Int'l School on Type Theory & Term Rewriting. A refereed version is [GKK00].

[GKK00] John Glauert, Richard Kenneway, and Zurab Khasidashvili. Minimal and optimal relative normalization in orthogonal expression reduction systems. *J. Logic & Comput.*, 10(3), 2000. Special issue on Type Theory and Term Rewriting.

[GLM92] Georges Gonthier, Jean-Jacques Lévy, and Paul-André Melliès. An abstract standardisation theorem. In *Proc. 7th Ann. IEEE Symp. Logic in Computer Sci.*, 1992.

[HL91a] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, I. In Lassez and Plotkin [LP91], pages 395–414.

[HL91b] Gérard Huet and Jean-Jacques Lévy. Computations in orthogonal rewriting systems, II. In Lassez and Plotkin [LP91], pages 415–443.

[HP99] M. Hanus and C. Prehofer. Higher-order narrowing with definitional trees. *Journal of Functional Programming*, 9(1):33–75, 1999.

[JM96] Trevor Jim and Albert R. Meyer. Full abstraction and the context lemma. *SIAM Journal of Computing*, 25(3):663–696, June 1996.

[Kah94] Stefan Kahrs. Compilation of Combinatory Reduction Systems. In *Proc. 1st Int'l Workshop Higher-Order Algebra, Logic, & Term Rewriting*, 1994.

[Ken89] J. R. Kennaway. Sequential evaluation strategies for parallel-or and related reduction systems. *Ann. Pure & Appl. Logic*, 43:31–56, 1989.

[KG96] Zurab Khasidashvili and John Glauert. Discrete normalization and standardization in deterministic residual structures. In ALP '96 [ALP96], pages 135–149.

[Kha90] Zurab Khasidashvili. Expression reduction systems. In *Proceedings of I. Vekua Institute of Applied Mathematics of Tbilisi State University*, volume 36, pages 200–220. 1990. Technical report.

[Klo80] Jan Willem Klop. *Combinatory Reduction Systems.* Mathematisch Centrum, Amsterdam, 1980. Ph.D. Thesis.

[KM91] Jan Willem Klop and Aart Middeldorp. Sequentiality in orthogonal term rewriting systems. *J. Symbolic Comp.*, 12:161–195, 1991.

[KvO95] Zurab Khasidashvili and Vincent van Oostrom. Context-sensitive conditional expression reduction systems. In *Proc. Int'l Workshop Graph Rewriting and Computation, SEGRAGRA '95*, Elec. Notes Comp. Sci., pages 141–150. Elsevier Science B.V., August 1995.

[KvOvR93] Jan Willem Klop, Vincent van Oostrom, and Femke van Raamsdonk. Combinatory Reduction Systems: Introduction and survey. *Theor. Comp. Sc.*, 121(1–2):279–308, 1993.

[LP91] Jean-Louis Lassez and Gordon Plotkin, editors. *Computational Logic: Essays in Honor of Alan Robinson.* MIT Press, 1991.

[Mel96] Paul-André Melliès. *Description Abstraite des Systèmes de Réécriture.* PhD thesis, Université Paris 7, December 1996.

[Mel98] Paul-André Melliès. Axiomatic Rewriting Theory IV: A stability theorem in Rewriting Theory. In *Proc. 13th Ann. IEEE Symp. Logic in Computer Sci.*, pages 287–298, 1998.

[Mid97] Aart Middeldorp. Call by need computations to root-stable form. In *Conf. Rec. POPL '97: 24th ACM Symp. Princ. of Prog. Langs.*, pages 94–105, 1997.

[Mul92] R. Muller. M-LISP: A representation-independent dialect of LISP with reduction semantics. *ACM Trans. on Prog. Langs. and Systs.*, 14(4):589–615, 1992.

[MW00] Robert Muller and J. B. Wells. Two applications of standardization and evaluation in Combinatory Reduction Systems. Submitted for publication, 2000.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In *Proc. 6th Ann. IEEE Symp. Logic in Computer Sci.*, pages 342–349, 1991.

[Nöc94] E. Nöcker. *Efficient Functional Programming: Compilation and Programming Techniques.* PhD thesis, Katholic University of Nijmegen, 1994.

[Plo75] Gordon D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theor. Comp. Sc.*, 1:125–159, 1975.

[SR93] R. C. Sekar and I. V. Ramakrishnan. Programming in equational logic: Beyond strong sequentiality. *Inf. & Comput.*, 104(1):78–109, 1993.

[Suz96] Taro Suzuki. Standardization theorem revisited. In ALP '96 [ALP96], pages 122–134.

[Tak93] M. Takahashi. λ-calculi with conditional rules. In *Proc. Int'l Conf. Typed Lambda Calculi and Applications*, 1993.

[Tay99] Paul Taylor. *Practical Foundations of Mathematics.* Cambridge University Press, 1999.

[vO94] Vincent van Oostrom. *Confluence for Abstract and Higher-Order Rewriting.* PhD thesis, Vrije Universiteit Amsterdam, 1994.

[vO96] Vincent van Oostrom. Higher-order families. In *Proc. 7th Int'l Conf. Rewriting Techniques and Applications*, 1996.

[vO97] Vincent van Oostrom. Finite family developments. In *Proc. 8th Int'l Conf. Rewriting Techniques and Applications*, 1997.

[vR96] Femke van Raamsdonk. *Confluence and Normalisation for Higher-Order Rewriting*. PhD thesis, Vrije Universiteit Amsterdam, 1996.

[Wol93] D. A. Wolfram. *The Clausal Theory of Types*, volume 21 of *Cambridge Tracts in Theoretical Comp. Sci.* Cambridge University Press, 1993.