Skew and ω-Skew Confluence and Abstract Böhm Semantics

Zena M. Ariola[1] and Stefan Blom[2]

[1] University of Oregon
[2] University of Innsbruck

Abstract. Skew confluence was introduced as a characterization of non-confluent term rewriting systems that had unique infinite normal forms or Böhm like trees. This notion however is not expressive enough to deal with all possible sources of non-confluence in the context of infinite terms or terms extended with letrec. We present a new notion called ω-skew confluence which constitutes a sufficient and necessary condition for uniqueness. We also present a theory that can lift uniqueness results from term rewriting systems to rewriting systems on terms with letrec. We present our results in the setting of Abstract Böhm Semantics, which is a generalization of Böhm like trees to abstract reduction systems.

1 Introduction

For term rewriting systems, it is well-known that uniqueness of normal forms follows from confluence and that given termination, confluence and uniqueness of normal forms are equivalent [27, 32]. Because of this, the normal form of a term is a natural candidate for the semantics of a term. However, there are many term rewriting systems in which there are interesting terms that do not have a normal form. For example, according to the rewriting rule given below:

$$\mathsf{F}\ x \to \mathsf{Cons}(x, \mathsf{F}\ x)$$

the term $\mathsf{F}\ 1$ does not have a normal form. More precisely, it doesn't have a *finite* normal form. The term does have an *infinite* normal form: the infinite list of ones.

There are several ways to define infinite normal forms on terms. An obvious way to define them is by means of infinitary rewriting [23, 24, 32, 16, 26]. Another way is to use a definition similar to that of the Böhm Tree in the lambda calculus [13] or Böhm like trees for term rewriting systems [25]. These infinite normal forms are closely related to the notion of observational equivalence. A detailed study of the relation between different notions of infinite normal form and contextual or observational equivalence is given in [19].

An advantage of the Böhm Tree approach or Böhm semantics over infinitary rewriting is that it allows one to deal in a simple manner with rewrite rules that remove unused definitions (garbage collection). For example, consider the rewrite rule

$$\mathsf{let}\ x = M\ \mathsf{in}\ N \to N, \text{ if } x \text{ does not occur free in } N\ .$$

To define Böhm semantics for a rewrite system containing this rule, it suffices to rewrite to normal form with respect to this rule. It should also be possible to deal with this in the setting of infinitary rewriting. The rule can be seen as a rewrite rule in a combinatory reduction system and although the latest results ([26]) cover fully extended CRS's only, it is known how to deal with non fully extended rules such as the η-rule ([30]), so the only thing which is needed is a combination of these two results.

We also prefer the Böhm Tree approach because in programming languages it is very important if a result can be reached in finitely many steps or not. This is immediately clear in the Böhm Tree approach and needs a compression lemma in the infinitary rewriting case.

The notions of infinite normal form and Böhm semantics are related to the notion of information content, also called instant semantics [36] or direct approximation [28, 34]. These notions determine a prefix of the term, which can never be changed by reduction. If we rewrite term graphs represented as terms with letrec then we can follow the same intuition, but we must be careful by how we interpret the prefix. In the style of calculus used by Ariola and Klop, the letrec bindings will remain at the top of the term and keep changing during the entire reduction. Strictly speaking, the only stable prefix is Ω, our constant for undefined. However, if we forget about the syntax and look at the picture of the graph then we will see stable prefixes as usual. Effectively, we have to ignore the letrec's when we determine the stable prefix. The same situation occurs when we consider a term rewriting system in which a strategy has been encoded as a symbol which keeps traveling up and down the term. In that case the administrative symbol(s) have to be ignored. If we consider abstract rewriting systems rather than term rewriting then the information content of a term is simply an observation about that object. The combination of the information contents of all reachable objects is what we refer to as the *Abstract Böhm Semantics*.

An important property is *uniqueness* (also referred to as soundness) of Böhm semantics. This property is similar to uniqueness of normal forms and states that every two convertible terms have the same Böhm semantics. Confluence implies uniqueness of Böhm semantics. However, confluence is not necessary for guaranteeing uniqueness. Skew confluence[1], as introduced in [4, 14, 6], characterizes rewrite systems that have unique Böhm semantics with respect to a notion of direct approximant or notion of finite information content. The idea behind skew confluence is that if there exists a computation that develops a certain information content, then any other computation can be extended to develop more detailed information content. The theory of skew confluence works well for term rewriting and certain forms of term graph rewriting. However, there are problems in applying the notion to other forms of term graph rewriting and infinitary rewriting. These problems are due to the fact that the information content is not really a single observation. It actually is a set of observations. For example, when we observe a term we actually observe the finite prefixes of that term. If the term is finite, the set of observations is finite and skew confluence works.

[1] The name was suggested to us by Jan Willem Klop.

If the term is infinite, the set of observations can become infinite and problems arise with skew confluence. For example, consider the infinite term M below:

$$M \equiv (\lambda f.f\,x\,(f\,x\,(f\,x\,(\cdots))))(I\,g) \ ,$$

where $I \equiv \lambda x.x$. On one side, M rewrites to an infinite normal form in two steps:

$$\begin{aligned}(\lambda f.f\,x\,(f\,x\,(f\,x\,(\cdots))))(I\,g) &\xrightarrow[\beta]{} (\lambda f.f\,x\,(f\,x\,(f\,x\,(\cdots))))g \\ &\xrightarrow[\beta]{} g\,x\,(g\,x\,(g\,x\,(\cdots)))) \ .\end{aligned}$$

This infinite normal form has itself as its information content. On the other side, M rewrites to M_1 which does not have a finite normalizing sequence:

$$M \equiv (\lambda f.f\,x\,(f\,x\,(f\,x\,(\cdots))))(I\,g) \xrightarrow[\beta]{} I\,g\,x\,(I\,g\,x\,(I\,g\,x\,(\cdots))) \equiv M_1 \ .$$

Moreover, in each reduct of M_1 only finitely many of the $(I\,g)$ redexes have been reduced, which means that each reduct has finite information content. For example, the information content of the terms in the sequence

$$I\,g\,x\,(I\,g\,x\,(\cdots)) \xrightarrow[\beta]{} g\,x\,(I\,g\,x\,(I\,g\,x\,(\cdots))) \xrightarrow[\beta]{} g\,x\,(g\,x\,(I\,g\,x\,(I\,g\,x\,(\cdots)))) \xrightarrow[\beta]{} \cdots$$

is

$$\Omega, g\,x\,\Omega, g\,x\,(g\,x\,\Omega), \cdots \ .$$

Because the information content of any reduct of M_1 is finite and the information content of the infinite normal form is infinite, it is impossible that the information content of any reduct of M_1 exceeds that of the normal form. Hence, we do not have skew confluence.

To solve the problem, we introduce a new variant of skew confluence, called *ω-skew confluence*, which is more suitable to the case of infinite information content. The idea behind ω-skew confluence is that if an object (a) reduces to two other objects (a_1, a_2) then for any observation that can be made about the first reduct (a_1) there exists a reduct (a_2') of the second reduct (a_2) about which the same observation can be made. We call this reduct the covering reduct. For example, given any prefix of the infinite normal form of M, we can find a reduct of M_1 whose information content exceeds the given prefix.

Note that for every observation, we may have a different covering reduct (a_2'). If it is possible to find a reduct that covers all observations then we have skew confluence. Such a reduct will always exist if the set of observations is finite, but as we have seen in the example it might not exist if the set of observations is infinite.

Later in this paper, we will consider term graphs represented by cyclic terms or terms with letrec. Cyclic graphs/terms can represent infinite terms. For example, the infinite list of ones can be represented as the graph in Fig. 1. Thus, the information content of a cyclic term can be infinite as well. Because examples of infinite information content for term graphs are lengthy, we will delay them until Sect. 5.

Proving (ω-)skew confluence of a non-confluent rewrite system can be rather tedious and non-confluence often complicates matters. In the case of cyclic calculi the source of non-confluence is often a subset of the rewrite rules which deals

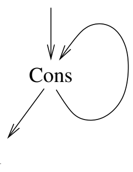


Fig. 1. Finite representation of an infinite list of ones

with substitution and/or unwinding. The idea behind the lifting theory in [15] is to partition the problem into a problem of the substitution rules and a problem of the other rules. The intention is that only the part dealing with substitution is non-confluent. Thus, one deals with non-confluence in a simplified setting and in the more complicated setting one can use confluence. In this paper we present an abstract version of this lifting theory. We start with an abstract rewrite system (ARS for short) equipped with a notion of information content which yields unique Abstract Böhm Semantics. We assume the ARS is equipped with a partial order. (E.g. finite terms with the prefix order.) Next, we consider an extension of the rewrite system where we use the ideal completion of the original set of objects as the semantics of the objects in the extension. (E.g. term graphs using unwinding of a term graph to an infinite term as semantics.) Finally, we define a construction that lifts the notion of information content from the original ARS to the extension and provide conditions under which we have uniqueness of Abstract Böhm Semantics with respect to the lifted notion of information content.

The notions and results about skew confluence are taken from [6]. The notion of abstract Böhm semantics is a modification of the notions in [14, 6]. The results about lifting are abstract versions of the results in [15]. The notion of ω-skew confluence is introduced explicitly for the first time in this paper. It was implicitly present in earlier work, but not identified as a notion.

The paper is organized as follows: We start in Sect. 2 with a few preliminaries. The next section contains an informal description of the different notions of confluence with both abstract examples and naive rewriting examples. In Sect. 4, we formalize skew and ω-skew confluence and the notion of abstract Böhm semantics. In Sect. 5, we present a counterexample to confluence that arises from unwinding a graph in different ways. We discuss how confluence modulo bisimilarity provides a solution. We also explain the need of skew and ω-skew confluence to cope with the loss of confluence when the substitution rules are extended with other rewrite rules. In Sect. 6, we use these notions to show the consistency of the call-by-name and call-by-need cyclic calculi. In Sect. 7, we present an abstract version of the lifting theory. We conclude in Sect. 8.

2 Preliminaries

We will briefly state a few definitions and introduce our notation.

Definition 1. *A partial order is a pair $(S, \leq)$, where $\leq$ is a transitive, reflexive and anti-symmetric binary relation over S. An upper bound of a set $S' \subseteq S$ is* *an element $s \in S$, such that $\forall s' \in S' : s' \leq s$. An element $s \in S$ is the least upper bound of S' (denoted as* $\mathrm{lub}\, S'$*) if and only if s is an upper bound and $s \leq s'$ for all upper bounds s' of S'. A non-empty set $D \subseteq S$ is a directed set if for every finite subset D' of D there exists $d \in D$ such that d is an upper bound of D'. A partial order $(S, \leq)$ is complete if there exists a least element and every directed subset has a least upper bound. A complete partial order is referred to as a CPO. A set $D \subseteq S$ is downward closed if $\forall d \in S, d' \in D : d \leq d' \Rightarrow d \in D$. A non-empty set $I \subseteq S$ is an ideal if I is downward closed and directed. The set of all ideals over S is denoted by $\mathcal{I}(S)$. The set of all ideals over S ordered by inclusion $(\mathcal{I}(S), \subseteq)$ is called the ideal completion of S, denoted $\mathcal{I}(S, \leq)$. The downward closure of $S' \subseteq S$ is given by*

$$\downarrow S' = \{s \in S \mid \exists s' \in S' : \ s \leq s'\} \ .$$

Definition 2. *Given a CPO $(A, \leq)$. An element $a \in A$ is* finite *if for every directed set $D \subseteq A$, such that $a \leq \mathrm{lub}\, D$, we have that there exists $d \in D$, such that $a \leq d$. The set of all finite elements in A is denoted by $\mathcal{F}(A)$. The CPO is algebraic if $\forall a \in A : a = \mathrm{lub}\{a' \in \mathcal{F}(A) \mid a' \leq a\}$.*

In other words, in an algebraic CPO, each element is a directed limit of its "finite" approximations. For an extensive treatment of the use of partial orders in semantics see [21].

Proposition 1. *If $\mathcal{A} \equiv (A, \leq)$ is a partial order with a least element then the ideal completion $\mathcal{A}_{\mathcal{I}} \equiv (\mathcal{I}(A), \subseteq)$ is an algebraic complete partial order.*

Definition 3. *An ARS is a structure $(A, \rightarrow)$, where A is a set of objects and $\rightarrow \subseteq A \times A$ is a relation, called the reduction relation.*
The transitive, reflexive closure of $\rightarrow$ is denoted $\twoheadrightarrow$.
The equivalence relation generated by $\rightarrow$, also called conversion, is denoted by $\xleftrightarrow{*}$ rather than the usual $=$, to avoid overloading of the symbol $=$.

In the lambda calculus, the compatible closure of a relation R is the least relation such that $M \ R \ N \Rightarrow C[M] \ R \ C[N]$ for any context C (see [13]). The constant Ω stands for an undefined term. By replacing an Ω with a larger term you get a "more defined term".

Definition 4. *Let Λ_Ω be the set of lambda calculus terms extended with the constant Ω. We define the order $\leq_\Omega$ as the transitive, reflexive and compatible closure of*

$$\Omega \leq M \ ,$$

where $M \in \Lambda_\Omega$.

Note that $(\Lambda_\Omega, \leq_\Omega)$ is a partial order with a least element (Ω). Hence, its ideal completion is an algebraic CPO. Moreover, the ideal completion is one of the representation of infinite lambda terms.

3 Confluence, Skew Confluence and ω-Skew Confluence

In the following, we give an informal description of the properties of *skew* and ω-*skew* confluence through a series of simple examples. To better understand these new properties, we review the well-established notion of confluence and a version of confluence modulo. We start by defining the set of objects A as the set consisting of the bottom element $\bot$, two copies of the set of natural numbers and infinity:

$$A = \{\bot\} \cup \mathbb{N} \cup \{\underline{n} \mid n \in \mathbb{N}\} \cup \{\infty\} \ .$$

By using possibly underlined numbers, we have both a natural equivalence and a natural order on our set of objects. Moreover, the number functions as the information content as well. For example, the numbers $\underline{2}$ and 2 have the same information content: 2.

For each abstract example, we will give a matching example in term rewriting or infinitary term rewriting. These term rewriting examples are derived from graph rewriting examples given in Sect. 5.

3.1 Confluence

Confluence is an important property since it guarantees the consistency of the rewriting theory. If rewriting formalizes execution, then confluence guarantees that execution of a program has a unique result. In other words, diverging computations with the same starting point can always converge on the same intermediate result. We define the reduction relation $\to$ on A as follows:

$$\bot \to 0,\ \ \bot \to \underline{0},\ n \to n+1,\ \underline{n} \to \underline{n+1},\ 2n \to \underline{2n},\ \underline{2n+1} \to 2n+1\ \ .$$

That is, we rewrite each number in $\mathbb{N}$ and its copy to its successor. Moreover, in addition to replacing $\bot$ with 0 and $\underline{0}$ we have rules to relate the numbers and their copies. We then have that $(A, \to)$ is confluent, which means that divergent computations can always be brought together, as shown in the following commuting diagram:

$$\begin{tikzcd}
\bot \arrow[r] \arrow[dr] & \underline{0} \arrow[r] & \underline{1} \arrow[r] \arrow[d] & \underline{2} \arrow[r] & \underline{3} \arrow[r] \arrow[d] & \cdots\cdots \\
& 0 \arrow[r] \arrow[u] & 1 \arrow[r] & 2 \arrow[r] \arrow[u] & 3 \arrow[r] & \cdots\cdots
\end{tikzcd}$$

A matching concrete example can be found in term rewriting. The reduction graph of the term A in the TRS:

$$\begin{array}{l}
A \to B \\
A \to C \\
C \to B \\
B \to F(C) \\
C \to F(B)
\end{array}$$

is:

$$\begin{tikzcd}
A \arrow[r] \arrow[dr] & B \arrow[r] & F(C) \arrow[r] \arrow[d] & F(F(B)) \arrow[r] & F(F(F(C))) \arrow[r] \arrow[d] & \cdots\cdots \\
 & C \arrow[r] \arrow[u] & F(B) \arrow[r] & F(F(C)) \arrow[r] \arrow[u] & F(F(F(B))) \arrow[r] & \cdots\cdots
\end{tikzcd}$$

3.2 Confluence Modulo

Confluence could fail for reasons that do not impact the end result. For example, if you optimize computations in different ways, you might not get exactly the same intermediate result, but as long as you perform a single unit of work in a single step you should get equivalent results. Similarly, in modeling execution one often reasons about a program modulo the names of bound variables. To continue with our example, we define the reduction relation $\rightarrow$ on A as follows:

$$\perp \rightarrow 0, \;\; \perp \rightarrow \underline{0}, \;\; n \rightarrow n+1, \;\; \underline{n} \rightarrow \underline{n+1} \;\; .$$

Unlike before, there are no reduction rules connecting the two copies of $\mathbb{N}$. This causes confluence to fail. To cope with the situation, one defines an equivalence (*i.e.,* reflexive, symmetric and transitive) relation $\sim$ on A and then shows that divergent reductions can always reach *equivalent* terms, as opposed to the *same* term. This is called confluence modulo $\sim$. For our running example, we define $\sim$ on A as follows:

$$a \sim a', \text{ if } |a| = |a'| \;\; ,$$

where $|.| : A \rightarrow \mathbb{N}$ is defined as:

$$|\perp| = 0, |n| = n+1, |\underline{n}| = n+1 \;\; .$$

In other words, the difference between the two different copies of n is not essential; we can regard it as "syntactic noise". We then obtain that $(A, \rightarrow)$ is confluent modulo $\sim$. Pictorially:

$$\begin{tikzcd}
\perp \arrow[r] \arrow[dr] & \underline{0} \arrow[r] \arrow[d, dash, dotted, "\sim"] & \underline{1} \arrow[r] \arrow[d, dash, dotted, "\sim"] & \underline{2} \arrow[r] \arrow[d, dash, dotted, "\sim"] & \underline{3} \arrow[r] \arrow[d, dash, dotted, "\sim"] & {} \\
 & 0 \arrow[r] & 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & {}
\end{tikzcd}$$

In the context of term graph rewriting, an interesting notion of confluence modulo is confluence modulo bisimilarity [11].

A matching concrete example can be found in term rewriting. The reduction graph of the term A in the TRS:

$$\begin{aligned}
A &\rightarrow B_1 \\
A &\rightarrow B_2 \\
B_1 &\rightarrow F(B_1) \\
B_2 &\rightarrow F(B_2)
\end{aligned}$$

is:

$$\begin{tikzcd}
A \arrow[r] \arrow[dr] & B_1 \arrow[r] \arrow[d, dash, dotted, "\sim"] & F(B_1) \arrow[r] \arrow[d, dash, dotted, "\sim"] & F(F(B_1)) \arrow[r] \arrow[d, dash, dotted, "\sim"] & F(F(F(B_1))) \arrow[r] \arrow[d, dash, dotted, "\sim"] & \cdots \\
& B_2 \arrow[r] & F(B_2) \arrow[r] & F(F(B_2)) \arrow[r] & F(F(F(B_2))) \arrow[r] & \cdots
\end{tikzcd}$$

where $\sim$ is the equivalence relation generated by $B_1 \sim B_2$.

3.3 Skew Confluence

The goal of optimization is of course to do more than one unit of work in a single step. But if you do two units of work in a single step then you can easily get an out-of-sync phenomenon. For example, define the reduction relation $\rightarrow$ on A as follows:

$$\bot \rightarrow 0,\ \ \bot \rightarrow \underline{1},\ \ n \rightarrow n+2,\ \ \underline{n} \rightarrow \underline{n+2}\ \ .$$

We have that $(A, \rightarrow)$ is neither confluent nor confluent modulo $\sim$. The situation is depicted below:

$$\begin{tikzcd}
\bot \arrow[r] \arrow[dr] & \underline{1} \arrow[r] & \underline{3} \arrow[r] & \underline{5} \arrow[r, dash, dotted] & {} \\
& 0 \arrow[r] & 2 \arrow[r] & 4 \arrow[r] & 6 \arrow[r, dash, dotted] & {}
\end{tikzcd}$$

On the top reduction we will always obtain an odd number and on the bottom we will always obtain an even number. However, notice that for every number $\underline{n}$ reached with the top reduction one can always reach a number greater than $\underline{n}$ with the bottom reduction, and vice-versa. Intuitively, it seems that both reductions converge to the same result. That result is infinity and we call it the *abstract Böhm semantics*. In this example, the uniqueness of the abstract Böhm semantics is guaranteed by the notion of skew confluence. Instead of requiring that divergent computations lead to the *same* or *equivalent* term, skew confluence requires that divergent computations reach a result which is *better*. In other words, instead of reasoning up to an equivalence relation we reason up to a quasi-order (*i.e.,* a reflexive and transitive relation). We define a quasi order $\preceq$ on A as follows:

$$a \preceq a',\ \text{if}\ |a| \leq |a'|\ \ ,$$

where $|.| : A \rightarrow \mathbb{N}$ is defined as before. We say that a' is better than a. We then have that $(A, \rightarrow)$ is skew confluent:

$$\begin{tikzcd}
\bot \arrow[rr] \arrow[dr] & & \underline{1} \arrow[rr] \arrow[dr, dash, dotted, "\preceq"] & & \underline{3} \arrow[rr] \arrow[dr, dash, dotted, "\preceq"] & & \underline{5} \arrow[rr] \arrow[dr, dash, dotted, "\preceq"] & & \underline{7} \arrow[r, dash, dotted] \arrow[dr, dash, dotted] & {} \\
& 0 \arrow[rr] \arrow[ur, dash, dotted, "\preceq"'] & & 2 \arrow[rr] \arrow[ur, dash, dotted, "\preceq"'] & & 4 \arrow[rr] \arrow[ur, dash, dotted, "\preceq"'] & & 6 \arrow[r, dash, dotted] \arrow[ur, dash, dotted, "\preceq"'] & {} & {}
\end{tikzcd}$$

A matching concrete example can be found in term rewriting. The reduction graph of the term A in the TRS:

$$\begin{aligned} &A \to B \\ &A \to F(B) \\ &B \to F(F(B)) \end{aligned}$$

is:

$$\begin{tikzcd}
A \arrow[rr] \arrow[dr] & & F(B) \arrow[rr] \arrow[dr, dash, dotted, "\preceq"] & & F(F(F(B))) \arrow[rr] \arrow[dr, dash, dotted, "\preceq"] & & F(F(F(F(F(B))))) \cdots \arrow[dr, dash, dotted, "\preceq"] & \\
& B \arrow[rr] \arrow[ur, dash, dotted, "\preceq"'] & & F(F(B)) \arrow[rr] \arrow[ur, dash, dotted, "\preceq"'] & & F(F(F(F(B)))) \cdots \arrow[ur, dash, dotted, "\preceq"'] & & {}
\end{tikzcd}$$

where $s \preceq t$ if t begins with at least as many F symbols as s.

3.4 ω-Skew-Confluence

Another possible outcome of optimization might be that you are suddenly able to do infinitely many units of work in a single step or the opposite where you throw away the possibility of doing more than finitely many units of work in a single step. Skew confluence is not expressive enough to deal with this situation. For example, define the reduction relation $\to$ on A as follows:

$$\bot \to \infty, \bot \to 0, n \to n+1 \ .$$

For this ARS skew confluence fails: $\bot \twoheadrightarrow \infty$ and $\bot \twoheadrightarrow 60$ and there does not exist an n, such that $60 \twoheadrightarrow n$ and $\infty \preceq n$. Pictorially:

$$\begin{tikzcd}
\infty \arrow[rrrr, dash, dotted] & & & & {} \\
& & & 2 \arrow[ur, dash, dotted] & \\
& & 1 \arrow[ur] & & \\
& 0 \arrow[ur] & & & \\
\bot \arrow[uuuu] \arrow[ur] & & & &
\end{tikzcd}$$

However, for each approximation m of ∞, we have that $60 \twoheadrightarrow m'$ such that $m \preceq m'$. We say that $(A, \to)$ is ω-skew confluent.

A matching concrete example can be found in infinitary rewriting. The reduction graph of the term A in the infinitary TRS:

$$\begin{aligned} A &\to F(G(F(G(\cdots)))) \\ A &\to F(F(G(F(G(\cdots))))) \\ F(F(x)) &\to F(G(x)) \\ G(G(x)) &\to G(F(x)) \end{aligned}$$

is (redexes are underlined):

$$\begin{tikzcd}
F(G(F(G(\cdots)))) \arrow[rrrr, dotted, no head] & & & & {} \\
& & & F(G(\underline{F(F(G(F(G(\cdots)))))})) \arrow[ur, dotted, no head] & \\
& & F(\underline{G(G(F(G(F(\cdots)))))}) \arrow[ur] & & \\
& \underline{F(F(G(F(G(\cdots)))))} \arrow[ur] & & & \\
\underline{A} \arrow[uuuu] \arrow[ur] & & & &
\end{tikzcd}$$

4 Skew and ω-Skew Confluence and Abstract Böhm Semantics

In this section we develop the theory of *Abstract Böhm Semantics*, which is a generalization of the theory of Böhm trees to abstract reduction systems. This generalization is based on the Böhm tree definition of Lévy [28].

4.1 Abstract Böhm Semantics

Seen from an abstract point of view, the first step in the Böhm tree construction of Lévy is to define a notion of information content. Given an ARS $(A, \to)$ and a partial order $(A, \leq)$, we would need to define a monotonic function $\omega : A \to A$, such that $\omega(a) \leq a$. Ketema followed this approach in his paper on Böhm like trees [25], but for us this doesn't work because we also want to compute Böhm semantics for terms with letrec. The problem is that terms with letrec can represent infinite terms. Hence, it is possible that the information content of a term with letrec is an infinite term. This is why one wants the domain of the rewrite system to be different from the domain of the information content.

The difference between finite and infinite information content is important. This difference needs to be reflected in our models. An obvious choice is to model finite objects as a partial order and infinite objects as ideals over finite objects. This however forces us to distinguish between finite and infinite objects, which makes for a cluttered presentation. Hence, we have chosen to use an algebraic complete partial order. In an algebraic CPO we can distinguish between finite and infinite elements as a property on the elements. Moreover, the infinite elements are completely defined by their sets of finite approximations.

Thus, we get the following abstraction of a rewrite systems with information content:

Definition 5. *A structure* $\mathcal{A} \equiv ((A, \to), \omega, (B, \leq))$ *is an* ARS with information content *(ARSI) if* $(A, \to)$ *is an ARS,* $(B, \leq)$ *is an algebraic complete partial* *order and $\omega : A \to B$ is monotonic with respect to $\to$. We say that $\mathcal{A}$ has finite information content if for every $a \in A$ we have that $\omega(a)$ is finite.*

Given an ARSI $((A, \to), \omega, (B, \leq))$ we refer to $\omega(a)$ as the information content of a or as the direct approximation of a. The function ω induces a quasi order $\leq_\omega$ on A, defined by $a \leq_\omega a'$ if $\omega(a) \leq \omega(a')$.

In the introduction we viewed the information content of an object as a set of observations to get some intuition. This intuition is consistent with our formalization of ARSI due to the fact that the power set of any set, ordered by inclusion is an algebraic CPO. Moreover, the finite elements in this CPO are the finite subsets of the original set.

The second step in the Böhm tree construction of Lévy is to define the actual tree or in our case the abstract Böhm semantics. The abstract Böhm semantics of an element a is supposed to be the set of all information that can be found in reducts of that element. Following the definition of Lévy we would formalize that set as:

$$\downarrow \{\omega(b) \mid a \twoheadrightarrow b\} \ .$$

The set $\{\omega(b) \mid a \twoheadrightarrow b\}$ is called the reachable information of a. Lévy already found that it is necessary to take its downward closure because otherwise there would be gaps in the set. For example, the reachable information content of

$$(\lambda x.f\,(I\,y)\,(x\,x))\,(\lambda x.f\,(I\,y)\,(x\,x))$$

is

$$\{\Omega, f\,\Omega\,\Omega, f\,y\,\Omega, f\,\Omega\,(f\,\Omega\,\Omega), \cdots\} \ .$$

But when we rewrite the two $I\,y$ redexes then the reachable information content of the result is

$$\{\Omega, f\,y\,\Omega, f\,y\,(f\,y\,\Omega), \cdots\} \ .$$

These sets are different, but their downward closures are the same.

In our case this is not enough. As we have seen in the introduction and Sect. 4, it is possible that a term allows two sequences: one sequence in which an infinite information content (e.g. ∞) is reached in a few steps and one sequence in which the information content is built up in finite pieces (e.g. $1, 2, 3, \cdots$). The set of finite pieces doesn't contain the infinite result so the downward closures will not be the same. To flatten this difference, we introduce the notion of finite element downward closure:

Definition 6. *Given a complete partial order $(A, \leq)$, we define*

$$\begin{array}{ll} \downarrow_{\mathcal{F}} s = \{a \in \mathcal{F}(A) \mid a \leq s\} & , \forall s \in A \ ; \\ \downarrow_{\mathcal{F}} S = \cup_{s \in S} \downarrow_{\mathcal{F}} (s) & , \forall S \subseteq A \ . \end{array}$$

We refer to $\downarrow_{\mathcal{F}} S$ as the finite element downward closure of S and to $\downarrow_{\mathcal{F}} s$ as the set of finite approximations of s. For example:

$$\downarrow_{\mathcal{F}} \{\infty\} = \downarrow_{\mathcal{F}} \{0, 2, 4, \cdots\} = \{0, 1, 2, \cdots\} \ .$$

Note that the finite element downward closure not only fills the gaps between the even numbers like the downward closure would, but also breaks ∞ up into it's finite approximations. Thus, our definition of abstract Böhm semantics is:

Definition 7. *Given an ARSI* $((A, \rightarrow), \omega, (B, \leq))$*. The Abstract Böhm Semantics* $\mathrm{ABS}(a)$ *of an element* $a \in A$ *is defined by*

$$\mathrm{ABS}(a) = \downarrow_{\mathcal{F}} \{\omega(a') \mid a \twoheadrightarrow a'\} \ .$$

The ARSI has unique abstract Böhm semantics if

$$a \overset{*}{\longleftrightarrow} a' \Rightarrow \mathrm{ABS}(a) = \mathrm{ABS}(a') \ .$$

In the running text, we will often omit the word abstract and just talk about Böhm semantics. As an example of an ARSI, let us define an ARSI whose Böhm semantics is the Böhm tree from the lambda calculus.

Example 1. We consider the ARS $(\Lambda, \underset{\beta}{\rightarrow})$. The function ω_{BT} from lambda terms to possibly infinite lambda terms is defined recursively as follows:

$$\omega_{\mathrm{BT}}(M) = \begin{cases} \lambda x_1 \cdots x_n . x\, \omega_{\mathrm{BT}}(M_1) \cdots \omega_{\mathrm{BT}}(M_k) & \text{, if } M \equiv \lambda x_1 \cdots x_n . x\, M_1 \cdots M_k \\ \Omega & \text{, otherwise} \end{cases}$$

This function is a notion of information content. That is,

$$((\Lambda, \underset{\beta}{\rightarrow}), \omega_{\mathrm{BT}}, \mathcal{I}(\Lambda_\Omega, \leq_\Omega))$$

is an ARSI. Moreover, for all lambda terms M, we have

$$\mathrm{BT}(M) = \mathrm{ABS}_{\omega_{\mathrm{BT}}}(M) \ ,$$

where $\mathrm{BT}(M)$ stands for the Böhm Tree of M.

The notion of uniqueness for abstract Böhm semantics for ARSI's is related to uniqueness of normal forms in ARS's in the following sense. Consider an ARS $(A, \rightarrow)$. We can build an order by adding a bottom element, leaving the original elements incomparable. Next, we define

$$\omega(a) = \begin{cases} a & \text{, if } a \text{ is a normal form} \\ \bot & \text{, otherwise} \end{cases}$$

This gives us an ARSI for any ARS. Moreover, the ARSI has unique abstract Böhm semantics if and only if the ARS has unique normal forms.

Next, we consider sufficient and necessary conditions for uniqueness.

4.2 Skew Confluence

Skew confluence is a sufficient condition for uniqueness. To define skew confluence we need a way of telling if an object is better than another object. We formalize this by considering an ARS and a quasi order.

Definition 8 (skew confluence). *Given an ARS $\mathcal{A} \equiv (A, \rightarrow)$ and a quasi order $(A, \preceq)$. The ARS $\mathcal{A}$ is skew confluent with respect to $\preceq$ if*

$$\forall a_1, a_2, a_3 \in A : a_1 \twoheadrightarrow a_2 \wedge a_1 \twoheadrightarrow a_3 \Rightarrow \exists a_4 : a_2 \preceq a_4 \wedge a_3 \twoheadrightarrow a_4 \ .$$

The commutative diagram for skew confluence is

$$\begin{tikzcd}
a_1 \arrow[r, two heads] \arrow[d, two heads] & a_3 \arrow[d, two heads, dashed] \\
a_2 \arrow[r, dash, dashed, "\preceq"'] & a_4
\end{tikzcd}$$

Confluence implies skew confluence. More precisely, if the reduction relation is increasing in a quasi order then confluence implies skew confluence with respect to that quasi order:

Proposition 2. *Given an ARS $\mathcal{A} \equiv (A, \rightarrow)$ and a quasi order $(A, \preceq)$. If $\rightarrow \subseteq \preceq$ and $\mathcal{A}$ is confluent then $\mathcal{A}$ is skew confluent with respect to $\preceq$.*

The definitions of confluence and skew confluence are easily extended to ARSI's. We say that an ARSI $((A, \rightarrow), \omega, (B, \leq))$ is confluent, if $(A, \rightarrow)$ is confluent and we say that it is skew confluent if $(A, \rightarrow)$ is skew confluent with respect to $\leq_\omega$.

4.3 ω-Skew Confluence

In [6], we defined abstract Böhm semantics (called infinite normal forms in that paper) in a setting where information content was always finite. In that setting, skew confluence is a necessary and sufficient condition for uniqueness of abstract Böhm semantics. In the current setting, it still is a sufficient condition, but it is not necessary. We will now define ω-skew confluence which is a necessary and sufficient condition in the presence of infinite information content. Later, we will show that for finite information content the two properties coincide. Hence, the result in this paper can be seen as an extension of the earlier result.

The definition of ω-skew confluence follows the intuition in the introduction based on observations. An ARSI is ω-skew confluent if given two diverging computations and an observation about the first result, we can find a reduct of the second result which allows the same observation. To make that formal, an observation about an object is defined as a finite element less than or equal to the information content of the object. This results in the following definition:

Definition 9 (ω-skew confluence). *The ARSI $\mathcal{A} \equiv ((A, \rightarrow), \omega, (B, \leq))$ is ω-skew confluent if*

$$\begin{aligned} &\forall a_1, a_2, a_3 \in A, d_1 \in \mathcal{F}(B) : \\ &\qquad a_1 \twoheadrightarrow a_2 \wedge a_1 \twoheadrightarrow a_3 \wedge d_1 \leq \omega(a_2) \Rightarrow \exists a_4 \in A : \ a_3 \twoheadrightarrow a_4 \wedge d_1 \leq \omega(a_4) \ . \end{aligned}$$

To be able to draw diagrams about ω-skew confluence, we introduce two arrows: $\xrightarrow[\omega]{}$ and $\xrightarrow[\omega]{\mathcal{F}}$. The former computes ω, the latter selects a finite element less than the information content:

Definition 10. *Given an ARSI* $((A, \to), \omega, (B, \leq))$*, we define the relations* $\xrightarrow[\omega]{} \subseteq A \times B$ *and* $\xrightarrow[\omega]{\mathcal{F}} \subseteq A \times \mathcal{F}(B)$ *as*

$$\begin{aligned} &\forall a \in A : && a \xrightarrow[\omega]{} \omega(a) \ ; \\ &\forall a \in A\ \forall b \in\downarrow_{\mathcal{F}} (\omega(a)) : && a \xrightarrow[\omega]{\mathcal{F}} b \ . \end{aligned}$$

Based on this definition, we can draw diagrams of skew confluence (SC) and ω-skew confluence (ωSC):

$$\begin{tikzcd} a_1 \arrow[r, two heads] \arrow[d, two heads] & a_3 \arrow[d, dashed, two heads] \\ a_2 \arrow[d, "\omega"'] & a_4 \arrow[d, dashed, "\omega"'] \\ \omega(a_2) \arrow[r, dash, dashed, "\leq"'] & \omega(a_4) \end{tikzcd} \qquad \begin{tikzcd} a_1 \arrow[r, two heads] \arrow[d, two heads] & a_3 \arrow[d, dashed, two heads] \\ a_2 \arrow[d, "\omega"', "\mathcal{F}"] & a_4 \arrow[d, dashed, "\omega"'] \\ d_1 \arrow[r, dash, dashed, "\leq"'] & \omega(a_4) \end{tikzcd}$$

SC $\qquad\qquad$ ωSC

Unfortunately, the diagram of ω-skew confluence is not suitable for diagram proofs using tiling. The reason is that the assumption on the left-hand side involves selecting a finite element and the conclusion on the right-hand side might not yield such an element. Hence, we include the following characterization.

Proposition 3. *For any ARSI* $((A, \to), \omega, (B, \leq))$ *the following diagram equivalence holds:*

$$\begin{tikzcd} {} \arrow[r, two heads] \arrow[d, two heads] & {} \arrow[d, dashed, two heads] \\ {} \arrow[d, "\omega"', "\mathcal{F}"] & {} \arrow[d, dashed, "\omega"'] \\ {} \arrow[r, dash, dashed, "\leq"'] & {} \end{tikzcd} \ \text{iff} \ \begin{tikzcd} {} \arrow[r, two heads] \arrow[d, two heads] & {} \arrow[d, dashed, two heads] \\ {} \arrow[d, "\omega"', "\mathcal{F}"] & {} \arrow[d, dashed, "\omega"', "\mathcal{F}"] \\ {} \arrow[r, dash, dashed, "\equiv"'] & {} \end{tikzcd} \ \text{iff} \ \begin{tikzcd} {} \arrow[r] \arrow[d, two heads] & {} \arrow[d, dashed, two heads] \\ {} \arrow[d, "\omega"', "\mathcal{F}"] & {} \arrow[d, dashed, "\omega"', "\mathcal{F}"] \\ {} \arrow[r, dash, dashed, "\leq"'] & {} \end{tikzcd} \tag{1}$$

$(i) \qquad\qquad (ii) \qquad\qquad (iii)$

Proof. **(i)⇒(iii)** Assume that $d \leq \omega(a)$ for some finite $d \in B$ and some $a \in A$. Because the CPO is algebraic, we have $\omega(a) = \text{lub} \downarrow_{\mathcal{F}} (\omega(a))$. Because d is finite, there exists $d' \in\downarrow_{\mathcal{F}} (\omega(a))$ such that $d \leq d'$. For that d', we have $a \xrightarrow[\omega]{\mathcal{F}} d'$ by definition.

(iii)⇒(ii) If $d \leq d'$ and $a \xrightarrow[\omega]{\mathcal{F}} d'$ then $a \xrightarrow[\omega]{\mathcal{F}} d$ must hold as well.

(ii)⇒(i) If $a \xrightarrow[\omega]{\mathcal{F}} d$ then $d \leq \omega(a)$ because $\omega(a) = \text{lub} \downarrow_{\mathcal{F}} (\omega(a))$.

Next, we prove that ω-skew confluence is a necessary and sufficient condition for uniqueness of abstract Böhm semantics.

Theorem 1. *Given an ARSI* $\mathcal{A} \equiv ((A, \to), \omega, (B, \leq))$*. The ARSI* $\mathcal{A}$ *has unique abstract Böhm semantics iff* $\mathcal{A}$ *is* ω*-skew confluent.*

Proof. From the arrow notation for ω, we can also derive the following equation:

$$\mathrm{ABS}(a) = \{b \mid a \twoheadrightarrow\xrightarrow[\omega]{\mathcal{F}} b\} \ ; \tag{2}$$

$\Rightarrow$ Assume that $\mathcal{A}$ has unique abstract Böhm semantics. Let $a_1, a_2, a_3 \in A$ and $d \in B$ be given such that $a_1 \twoheadrightarrow a_2$, $a_1 \twoheadrightarrow a_3$ and $a_2 \xrightarrow[\omega]{\mathcal{F}} d$.
In particular, we have $a_1 \twoheadrightarrow\xrightarrow[\omega]{\mathcal{F}} d$, so because of Eq. 2, we have that $d \in \mathrm{ABS}(a_1)$. Because of the uniqueness, we have that $\mathrm{ABS}(a_1) = \mathrm{ABS}(a_3)$, so $d \in \mathrm{ABS}(a_3)$. Again by Eq. 2, it follows that $a_3 \twoheadrightarrow\xrightarrow[\omega]{\mathcal{F}} d$. This proves diagram 1.(ii) and hence by the previous proposition ω-skew confluence.

$\Leftarrow$ Assume that $\mathcal{A}$ is ω-skew confluent. Let $a_1, a_2 \in A$ be given. It suffices to show that if $a_1 \rightarrow a_2$ then $\mathrm{ABS}(a_1) = \mathrm{ABS}(a_2)$.
From the definition of abstract Böhm semantics it is obvious that $\mathrm{ABS}(a_2) \subseteq \mathrm{ABS}(a_1)$, so the part we need to show is $\mathrm{ABS}(a_1) \subseteq \mathrm{ABS}(a_2)$. Let $d \in \mathrm{ABS}(a_1)$ be given. Then by Eq. 2, it follows that $a_1 \twoheadrightarrow\xrightarrow[\omega]{\mathcal{F}} d$. Because of ω-skew confluence and the previous proposition, diagram 1.(ii) holds. From this diagram it follows that $a_2 \twoheadrightarrow\xrightarrow[\omega]{\mathcal{F}} d$ and hence by Eq. 2 $d \in \mathrm{ABS}(a_2)$.

The definition of ω-skew confluence allows us to select a different matching reduct for every observation that must be matched. If we can match every observation with the same reduct then this is called uniform ω-skew confluence. This is however not a new property because it is equivalent to skew confluence:

Proposition 4. *Given an ARSI $\mathcal{A} \equiv ((A, \rightarrow), \omega, (B, \leq))$. We say that $\mathcal{A}$ is uniformly ω-skew confluent if*

$$\begin{aligned}&\forall a_1, a_2, a_3 \in A\ \exists a_4 \in A\ \forall d_1 \in \mathcal{F}(B) :\\ &\qquad a_1 \twoheadrightarrow a_2 \wedge a_1 \twoheadrightarrow a_3 \wedge d_1 \leq \omega(a_2) \Rightarrow a_3 \twoheadrightarrow a_4 \wedge d_1 \leq \omega(a_4)\ .\end{aligned}$$

We have that $\mathcal{A}$ is uniformly ω-skew confluent iff $\mathcal{A}$ is skew confluent.

Proof. Follows from the claim that

$$\forall a, a' \in A :\ (\forall d \in \mathcal{F}(B) :\ d \leq \omega(a) \Rightarrow d \leq \omega(a')) \Leftrightarrow a \leq_\omega a'\ .$$

The claim follows from the fact that $\omega(a) = \mathrm{lub} \downarrow_{\mathcal{F}} (\omega(a))$ and $\omega(a') = \mathrm{lub} \downarrow_{\mathcal{F}} (\omega(a'))$ because $(B, \leq)$ is an algebraic CPO.

Hence, it is not surprising that we can prove skew confluence implies ω-skew confluence. We also prove that in the case of finite information content the two notions are equivalent.

Proposition 5. *Given an ARSI $\mathcal{A} \equiv ((A, \rightarrow), \omega, (B, \leq))$.*

(i) If $\mathcal{A}$ is skew confluent then $\mathcal{A}$ is ω-skew confluent.
(ii) If $\mathcal{A}$ is ω-skew confluent and $\mathcal{A}$ has finite information content then $\mathcal{A}$ is skew confluent.

Proof. (i) By the previous proposition skew confluence implies uniform ω-skew confluence. It is obvious that uniform ω-skew confluence implies ω-skew confluence.

(ii) Because the information content is finite, we have

$$\forall a \in A : \; a \xrightarrow[\omega]{\mathcal{F}} \omega(a)$$

From this fact and ω-skew confluence, skew confluence follows easily.

This completes the presentation of the basic theory of abstract Böhm semantics. We continue with an example of an application area: term graph rewriting based on terms with letrec.

5 Lack of Confluence in Term Graph Rewriting

The need for a less restrictive notion of confluence arises in practice if one wants to provide a more accurate foundation of programming languages. To reason about either execution or optimizations one has to deal with the notion of sharing and cyclic structures [12, 31, 2]. As pointed out by Wadsworth [35], these concerns can be accommodated by considering term graph rewriting as opposed to term (or tree) rewriting.

As pointed out in [9, 10], term graphs can be nicely represented as terms with the letrec [2] construct:

$$\langle M \mid x_1 = M_1, \cdots, x_n = M_n \rangle \ .$$

We sometimes refer to the variables $x_1, \cdots, x_n$ as the recursion variables, to the equations and to M as the internal and external part of the letrec construct, respectively. Because of the capability of the letrec to represent graphs with cycles, we refer to terms with letrec's as cyclic terms.

The cyclic structure depicted in Fig. 1 is represented as

$$\langle x \mid x = \mathsf{Cons}(1, x) \rangle \ .$$

The advantage of this representation is that one can apply existing term rewrite rules directly to the cyclic term. However, the old rewrite rules are not enough: we must also use rules that modify the letrec structure to make potential redexes visible [9, 10]. For example, with respect to the rule

$$\mathsf{F}(1) \rightarrow \mathsf{G}(1)$$

the terms:

$$\langle \mathsf{F}(x) \mid x = 1 \rangle \quad \text{and} \quad \langle x \mid x = \mathsf{F}(y), y = 1 \rangle$$

[2] We use the Ariola/Klop notation for letrec ($\langle M \mid E \rangle \equiv \mathsf{letrec}\ E\ \mathsf{in}\ M$).

are in normal form. Whereas, their corresponding graphs contain a redex. To cope with this situation, the following two rules for external and internal substitution are introduced:

$$\begin{array}{rcl} \langle C[x] \mid x = M, E\rangle & \xrightarrow[\text{es}]{} & \langle C[M] \mid x = M, E\rangle \\ \langle M \mid x = C[y], y = N, E\rangle & \xrightarrow[\text{is}]{} & \langle M \mid x = C[N], y = N, E\rangle \end{array}$$

where $C[x]$ stands for a one-hole context filled with variable x, and E for a collection of unordered equations. According to these rules we have:

$$\begin{array}{rcl} \langle \mathsf{F}(x) \mid x = 1\rangle & \xrightarrow[\text{es}]{} & \langle \mathsf{F}(1) \mid x = 1\rangle \\ \langle x \mid x = \mathsf{F}(y), y = 1\rangle & \xrightarrow[\text{is}]{} & \langle x \mid x = \mathsf{F}(1), y = 1\rangle \end{array}$$

Both right-hand sides contain the redex $\mathsf{F}(1)$. One more substitution rule is needed. Consider the following rule:

$$\mathsf{F}(\mathsf{F}(x)) \to \mathsf{G}(x)$$

and the term

$$\langle x \mid x = \mathsf{F}(x)\rangle \ .$$

To make the redex explicit in the internal part, one needs a substitution applied to the equation itself. We call it cyclic substitution:

$$\langle M \mid x = C[x], E\rangle \xrightarrow[\text{cs}]{} \langle M \mid x = C[C[x]], E\rangle$$

We have:

$$\langle x \mid x = \mathsf{F}(x)\rangle \xrightarrow[\text{cs}]{} \langle x \mid x = \mathsf{F}(\mathsf{F}(x))\rangle \to \langle x \mid x = \mathsf{G}(x)\rangle \ .$$

The problem with these three substitution rules is that confluence is lost. The classical example is:

$$\begin{tikzcd} \langle x \mid x = \underset{}{\overset{M \atop \equiv}{}}\mathsf{F}(x)\rangle \arrow[r, "\text{es}"'] \arrow[d, "\text{cs}"] & \langle \mathsf{F}(x) \mid x = \mathsf{F}(x)\rangle \arrow[r, "\text{cs}"'] & \langle \mathsf{F}(x) \mid x = \underset{\equiv \atop M_o}{\mathsf{F}(\mathsf{F}(x))}\rangle \\ \langle x \mid x = \underset{\equiv \atop M_e}{\mathsf{F}(\mathsf{F}(x))}\rangle & & \end{tikzcd}$$

The cyclic terms M_o and M_e do not have a common reduct because any reduct of M_o will contain an odd number of F symbols and any reduct of M_e an even number.

The fact that the three substitution rules aren't confluent is not in itself a big problem. Not only are these rewrite rules confluent modulo bisimulation, but it is also possible to add rewrite rules to regain confluence. Even if we add an orthogonal TRS we can keep the rewrite systems confluent. (See [29]).

However, if we consider combinatory reduction systems or non-orthogonal TRS's then interaction between the rewrite rules and substitution rules becomes a real problem. For example, consider the TRS

$$\begin{array}{l} \mathsf{F}(\mathsf{F}(x)) \;\rightarrow \mathsf{F}(\mathsf{G}(x)) \\ \mathsf{G}(\mathsf{G}(x)) \rightarrow \mathsf{G}(\mathsf{F}(x)) \end{array} \tag{3}$$

This TRS is confluent and terminating, but not orthogonal. When we apply these rewrite rules to M_o and M_e we get:

$$\begin{array}{l} M_e \rightarrow \langle x \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \equiv M'_e \;; \\ M_o \rightarrow \langle \mathsf{F}(x) \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \equiv M'_o \;. \end{array}$$

As before, a count of the symbols leads to the conclusion that these M'_e and M'_o do not have a common reduct.

All reducts of M'_e will be of the form

$$\langle (\mathsf{FG})^n(x) \mid x = (\mathsf{FG})^m(x)\rangle \;,$$

where $(\mathsf{FG})^0(x) = x$ and $(\mathsf{FG})^{n+1}(x) = \mathsf{F}(\mathsf{G}((\mathsf{FG})^n(x)))$. Note that a term of this form has exactly the same number of F's and G's and that there it will not contain a redex of the TRS rules from Eq. 3.

All reducts of M'_o will be of the form

$$\langle (\mathsf{FG})^n(\mathsf{G}((\mathsf{FG})^k(x))) \mid x = (\mathsf{FG})^m(x)\rangle \text{ or } \langle (\mathsf{FG})^n(\mathsf{F}((\mathsf{FG})^k(x))) \mid x = (\mathsf{FG})^m(x)\rangle \;.$$

Note that a term of one of these forms has one more F than G's or one more G than F's. More importantly, a term of these forms always has a redex with respect to the TRS rules.

The fundamental difference between the reduction sequences from M to M'_e and from M to M'_o is that in the first sequence the "correct" redex is exposed and contracted. In the second sequence the "wrong" redex is exposed and contracted. The result of that is that a redex remains, which will create a new redex whenever it is contracted. Because a new redex is created in every step, it is impossible to reduce M'_0 to normal form in finitely many steps.

We will now define a notion of information content for this TRS and show that the resulting ARSI is ω-skew confluent. The function ω from cyclic terms to infinite terms is defined as follows:

$$\omega(M) = \text{lub}\{N \mid M \underset{\omega}{\twoheadrightarrow} N \text{ and } N \text{ is a normal form}\} \;, \tag{4}$$

where $\underset{\omega}{\twoheadrightarrow}$ is defined as follows:

$$\begin{array}{ll} \langle C[x] \mid x = M, E\rangle & \underset{\omega}{\rightarrow} \langle C[M] \mid x = M, E\rangle \\ \mathsf{F}(\mathsf{F}(x)) & \underset{\omega}{\rightarrow} \Omega \\ \mathsf{G}(\mathsf{G}(x)) & \underset{\omega}{\rightarrow} \Omega \\ \langle M \mid x_1 = M_1, \cdots, x_n = M_n\rangle & \underset{\omega}{\rightarrow} M[x_1 := \Omega, \cdots, x_n := \Omega] \end{array}$$

For example:

$$\begin{aligned} M'_o &\equiv \langle \mathsf{F}(x) \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \\ &\xrightarrow[\omega]{} \langle \mathsf{F}(\mathsf{F}(\mathsf{G}(x))) \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \\ &\xrightarrow[\omega]{} \langle \Omega \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \\ &\xrightarrow[\omega]{} \Omega \ . \end{aligned}$$

Because this normal form is unique, we have that

$$\omega(M'_o) = \mathrm{lub}\{\Omega\} = \Omega \ .$$

The normal form of M'_e with respect to $\xrightarrow[\omega]{}$ is not unique. For every n we have

$$M'_e \equiv \langle x \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \xrightarrow[\omega]{}\!\!\!\!\rightarrow (\mathsf{FG})^n(\Omega) \ .$$

This means that

$$\omega(M'_e) = \mathrm{lub}\{(\mathsf{FG})^n(\Omega) \mid n \in \mathbb{N}\} = (\mathsf{FG})^\omega \ .$$

There does not exist any reduct of M'_o, whose information content is $(\mathsf{FG})^\omega$ so the rewrite system is not skew confluent. However, for every n, we can find a reduct of M'_o, such that the information content of the reduct is $(\mathsf{FG})^n(\Omega)$:

$$M^n = \langle (\mathsf{FG})^n(\mathsf{F}(x)) \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \ .$$

These reducts show that the example is ω-skew confluent.

6 Cyclic Lambda Calculi

In this section we consider extensions of the call-by-name lambda calculus and the call-by-need lambda calculus [8, 7] with cyclic structures. These extensions contain a large number of rules. This is in order to be able to address the correctness of compilation by transformation [18]. But before we give the calculi let us start with the basic principles.

We are interested in cyclic lambda terms. That is, lambda terms extended with letrec.

Definition 11. *The set of cyclic lambda terms $\Lambda\circ$ is defined as follows:*

$$\begin{array}{ll} \textit{Terms} & M ::= x \mid \lambda x.M \mid M\,N \mid \langle M \mid E\rangle \ ; \\ \textit{Equations} & E ::= x_1 = M_1, \ldots, x_n = M_n \ . \end{array}$$

where the variables $x_1, \cdots, x_n$, are distinct from each other and the order of the equations does not matter. Terms are taken up to α-conversion.

Because it is not really important where the definitions are placed, we base our lambda calculi on a rewrite system that brings any cyclic term into a standard form. This standard form is:

$$\begin{array}{l} ST ::= x \mid \langle x \mid SE\rangle \ ; \\ SE ::= x = x \mid x = \lambda y.ST \mid x = x_1\,x_2 \mid SE, SE \ . \end{array}$$

Table 1. A rewrite system for normalizing the representation of a graph

Variable substitution:			
$\langle M \mid x = y, E \rangle$	$\xrightarrow[\text{vs}]{}$	$\langle M[x := y] \mid E[x := y] \rangle$	$x \not\equiv y$
Lift:			
$\langle M \mid E \rangle\ N$	$\xrightarrow[\text{lift}]{}$	$\langle M\ N \mid E \rangle$	
$M\ \langle N \mid E \rangle$	$\xrightarrow[\text{lift}]{}$	$\langle M\ N \mid E \rangle$	
$\lambda x.\langle M \mid E_1, E_2 \rangle$	$\xrightarrow[\text{lift}]{}$	$\langle \lambda x.\langle M \mid E_1 \rangle \mid E_2 \rangle$	C1
Merge:			
$\langle \langle M \mid E_1 \rangle \mid E_2 \rangle$	$\xrightarrow[\text{em}]{}$	$\langle M \mid E_1, E_2 \rangle$	
$\langle M \mid x = \langle N \mid E_1 \rangle, E_2 \rangle$	$\xrightarrow[\text{im}]{}$	$\langle M \mid x = N, E_1, E_2 \rangle$	
Garbage collection:			
$\langle M \mid E_1, E_2 \rangle$	$\xrightarrow[\text{gc}]{}$	$\langle M \mid E_1 \rangle$	C2
$\langle M \mid \rangle$	$\xrightarrow[\text{gc}]{}$	M	
Naming:			
$C_{\text{safe}}[\lambda y.M]$	$\xrightarrow[\text{name}]{}$	$C_{\text{safe}}[\langle x \mid x = \lambda y.M \rangle]$	C3
$C_{\text{safe}}[M\,N]$	$\xrightarrow[\text{name}]{}$	$C_{\text{safe}}[\langle x \mid x = M\,N] \rangle$	C3

C1: E_2 is non-empty and neither x nor a variable defined in E_1 occurs free in E_2;
C2: E_2 is non-empty and none of the variables defined in E_2 occur free in E_1 or M;
C3: x is a fresh variable and the rule is *not* closed under contexts;

$$\begin{aligned} C_{\text{safe}} &::= C' \mid C[\lambda x.C'] \mid C[C'\ M] \mid C[M\ C']\ \ ; \\ C' &::= \square \mid \langle C' \mid E \rangle\ \ . \end{aligned}$$

That is, a standard term is either a variable or a letrec with a non-empty list of standard definitions. A standard definition can be a black hole definition ($x = x$), a function definition ($x = \lambda y.ST$) or an application definition ($x = x_1\,x_2$). In Table 1 we present a confluent and terminating rewrite system for computing standard representations. Apart from the usual conditions on lifting and garbage collection it contains a special condition on the naming rules. Unlike the other rules which can be applied in any context these rules are only applicable in safe contexts. This restriction is necessary to guarantee termination.

Representing the same graph is one equivalence. Another equivalence is having the same unwinding. The unwinding of a cyclic term is the unique (infinite) term represented by the cyclic term. The substitution rules compute the unwinding in the sense that we can define a notion of information content such that the unwinding of a cyclic term is the Böhm semantics of the term:

$$\begin{aligned} \omega_{\text{es}}(x) &= x \\ \omega_{\text{es}}(M_1\,M_2) &= \omega_{\text{es}}(M_1)\,\omega_{\text{es}}(M_2) \\ \omega_{\text{es}}(\lambda x.M) &= \lambda x.\omega_{\text{es}}(M) \\ \omega_{\text{es}}(\langle M \mid x_1 = M_1, \cdots, x_n = M_n \rangle) &= \omega_{\text{es}}(M)[x_1 := \Omega, \cdots, x_n := \Omega] \end{aligned}$$

We have used the label es because external substitution is actually the only rule needed to compute the unwinding.

Definition 12. *The unwinding of a cyclic term M, denoted $[\![M]\!]$, is the Böhm semantics of M with respect to the ARSI $((\Lambda\circ, \xrightarrow[\text{es}]{}), \omega_{\text{es}}, \mathcal{I}(\Lambda_\Omega, \leq_\Omega))$:*

$$[\![M]\!] = \text{ABS}_{\text{es}}(M) \ .$$

What we want is a set of rewrite rules such that terms with the same unwinding are convertible. To that end, we introduce a rewrite rule for copying. What copying means is that one duplicates definitions and for every reference to a duplicated variable one can choose to refer to the original definition or to one of the copies. In the following definition, we define the copy rewrite rule on graphs by means of a meta rewrite system.

Definition 13. *On cyclic terms extended with the binary symbol $+$, we define the rewrite relation $\xrightarrow[+]{}$ as follows:*

$$\begin{array}{lll}\langle M \mid x = N, E\rangle & \xrightarrow[+]{} \langle M\sigma \mid y = N\sigma, z = N\sigma, E\sigma\rangle & \textit{where } \sigma = [x := y + z] \\ & & \textit{for fresh variables } y \textit{ and } z \\ y + z & \xrightarrow[+]{} y & \\ y + z & \xrightarrow[+]{} z & \end{array}$$

If $M \xrightarrow[+]{}\!\!\!\!\!\!\!\!\rightarrow N$ and M nor N contain on occurrence of $+$ then

$$M \xrightarrow[\text{cp}]{} N \ .$$

For example:

$$\begin{array}{rl}\lambda z.\langle u \mid u = z\,u\rangle \xrightarrow[+]{} & \lambda z.\langle (x+y) \mid x = z\,(x+y), y = z\,(x+y)\rangle \\ \xrightarrow[+]{}^{3} & \lambda z.\langle x \mid x = z\,y, y = z\,x\rangle \ .\end{array}$$

For a more precise discussion of the issues of representation see [3].

The principle of cyclic lambda calculi is simple. In the beta-rule, instead of a substitution one uses a letrec:

$$(\lambda x.M)\ N \xrightarrow[\beta\circ]{} \langle M \mid x = N\rangle \ .$$

To simulate the normal β-rule, we must obviously include substitution. But this is not enough. Consider the term (I stands for the term $\lambda x.x$):

$$\langle \lambda y.x\ y \mid x = I\rangle\ I \ .$$

It contains a potential redex "$(\lambda y.x\ y)\ I$" which needs to be made explicit by moving the equation "$x = I$" around. This is made possible by the first lift rule:

$$\begin{array}{rl}\langle \lambda y.x\ y \mid x = I\rangle\ I \xrightarrow[\text{lift}]{} & \langle (\lambda y.x\ y) I \mid x = I\rangle \\ \xrightarrow[\beta\circ]{} & \langle\langle x\ y \mid y = I\rangle \mid x = I\rangle \\ \xrightarrow[\text{es}]{} & \langle\langle I\ y \mid y = I\rangle \mid x = I\rangle \\ \xrightarrow[\beta\circ]{} & \langle\langle\langle z \mid z = y\rangle \mid y = I\rangle \mid x = I\rangle \\ \xrightarrow[\text{es}]{}\!\!\!\!\!\!\!\rightarrow & \langle\langle\langle I \mid z = y\rangle \mid y = I\rangle \mid x = I\rangle \ .\end{array}$$

Table 2. The cyclic call-by-name lambda calculus $\lambda\circ$

$\beta\circ$:		
$(\lambda x.M)\ N$	$\xrightarrow[\beta\circ]{} \langle M \mid x = N\rangle$	
Substitution:		
$\langle C[x] \mid x = M, E\rangle$	$\xrightarrow[\text{es}]{} \langle C[M] \mid x = M, E\rangle$	
$\langle M \mid x = C[y], y = N, E\rangle$	$\xrightarrow[\text{is}]{} \langle M \mid x = C[N], y = N, E\rangle$	
Lift:		
$\langle M \mid E\rangle\ N$	$\xrightarrow[\text{lift}]{} \langle M\ N \mid E\rangle$	
$M\ \langle N \mid E\rangle$	$\xrightarrow[\text{lift}]{} \langle M\ N \mid E\rangle$	
$\lambda x.\langle M \mid E_1, E_2\rangle$	$\xrightarrow[\text{lift}]{} \langle \lambda x.\langle M \mid E_1\rangle \mid E_2\rangle$	C1
Merge:		
$\langle\langle M \mid E_1\rangle \mid E_2\rangle$	$\xrightarrow[\text{em}]{} \langle M \mid E_1, E_2\rangle$	
$\langle M \mid x = \langle N \mid E_1\rangle, E_2\rangle$	$\xrightarrow[\text{im}]{} \langle M \mid x = N, E_1, E_2\rangle$	
Garbage collection:		
$\langle M \mid E_1, E_2\rangle$	$\xrightarrow[\text{gc}]{} \langle M \mid E_1\rangle$	C2
$\langle M \mid\rangle$	$\xrightarrow[\text{gc}]{} M$	
Copy:		
M	$\xrightarrow[\text{cp}]{} N$	

C1: E_2 is non-empty and neither x nor a variable defined in E_1 occurs free in E_2;
C2: E_2 is non-empty and none of the variables defined in E_2 occur free in E_1 or M.

Our entire call-by-name cyclic lambda calculus in given in Table. 2. Basically, it consists of the $\beta\circ$-rule, the substitution rules, the representation rules and copying. However, superfluous rules have been removed. For example, the cyclic substitution is derivable from copying, internal substitution and garbage collection:

$$\begin{aligned}\langle M \mid x = C[x], E\rangle &\xrightarrow[\text{cp}]{} \langle M \mid x = C[y], y = C[x], E\rangle \\ &\xrightarrow[\text{is}]{} \langle M \mid x = C[C[x]], y = C[x], E\rangle \\ &\xrightarrow[\text{gc}]{} \langle M \mid x = C[C[x]], E\rangle\ .\end{aligned}$$

Naming is not included since, due to substitution, it is not needed to equate different representations of a graph.

We explained the third lift rule as a rule needed to normalize terms representing graphs. This is not the only explanation of this rule. It is also needed to be able to capture different kinds of evaluation, such as, full laziness [34]. The rule lifts declarations, that do not contain occurrences of the bound variable, out of a lambda body. As an example, consider the following term:

$$\langle f\, I\, (f\, I) \mid f = \lambda x.\langle w\, x \mid w = \underline{(I\, I)}\rangle\rangle\ .$$

If we do not lift the redex $I\,I$ (*i.e.,* the one underlined) out of the lambda body, that redex will be reduced twice. We have:

$$\begin{aligned}&\langle f\, I\, (f\, I) \mid f = \lambda x.\langle w\, x \mid w = (I\, I)\rangle\rangle \\ &\quad\xrightarrow[\text{lift}]{} \langle f\, I\, (f\, I) \mid f = \langle \lambda x.\langle w\, x \mid\rangle \mid w = (I\, I)\rangle\rangle\ .\end{aligned}$$

To complete the work done by the lift rule we apply the internal merge:

$$\begin{array}{l}\langle f\,I\,(f\,I) \mid f = \langle \lambda x.\langle w\,x \mid\rangle \mid w = (I\,I)\rangle\rangle \\ \quad \xrightarrow[\text{im}]{} \langle f\,I\,(f\,I) \mid f = \lambda x.\langle w\,x \mid\rangle, w = (I\,I)\rangle \\ \quad \xrightarrow[\text{es}]{} \langle (\lambda x.\langle w\,x \mid\rangle) I\,(f\,I) \mid f = \lambda x.\langle w\,x \mid\rangle, w = (I\,I)\rangle \ .\end{array}$$

Note that the substitution of f did not cause the duplication of the redex $I\,I$.

In Sect. 5, we already pointed out that the substitution rules lead to non-confluence. However, the cyclic call-by-name lambda calculus is skew-confluent with respect to a notion of finite information content, which returns a lambda calculus term extended with a constant Ω. As usual in the field of programming languages, the information content for the call-by-name lambda calculus is derived from that for the Lévy-Longo tree rather than the Böhm tree.

Definition 14. *Given the ARS $(\Lambda_\circ, \xrightarrow[\lambda\circ]{})$ and the partial order $(\Lambda_\Omega, \leq_\Omega)$. The finite information content $\omega_{\lambda\circ}(M)$ of a term $M \in \Lambda\circ$ is the normal form of M with respect to the following rules:*

$$\begin{array}{llll} (\lambda x.M)N & \xrightarrow[\omega_{\lambda\circ}]{} \Omega & & \beta\omega \\ \langle C[x] \mid x = M, E\rangle & \xrightarrow[\omega_{\lambda\circ}]{} \langle C[\Omega] \mid x = M, E\rangle & & es\omega \\ \Omega M & \xrightarrow[\omega_{\lambda\circ}]{} \Omega & & @\omega \\ \langle M \mid E\rangle & \xrightarrow[\omega_{\lambda\circ}]{} M & C & \mathrm{gc}\omega \end{array}$$

C: none of the variables defined in E occurs free in M.

Examples: $\omega_{\lambda\circ}(\langle \lambda x.y\,z \mid y = I\rangle) = \lambda x.\Omega$, $\omega_{\lambda\circ}(\langle x \mid x = x\rangle) = \Omega$, $\omega_{\lambda\circ}(\langle x\,y \mid y = I\rangle\,x) = (x\,\Omega)\,x$, and $\omega_{\lambda\circ}(\langle x\,x \mid x = I\rangle) = \Omega$. Note that even though $\langle x\,y \mid y = I\rangle\,x$ is a lift redex, its information content is not Ω.

Theorem 2. *The ARSI $((\Lambda_\circ, \xrightarrow[\lambda\circ]{}), \omega_{\lambda\circ}, \mathcal{I}(\Lambda_\Omega, \leq_\Omega))$ is skew confluent.*

This theorem guarantees uniqueness of Böhm semantics. A direct proof can be found in [6]. In the next section, we will develop a theory which allows us to prove uniqueness of Böhm semantics from a list of other properties. First, we present a call-by-need calculus.

One of the features of the call-by-need calculus is that duplication of terms is restricted to the class of values. Thus, we need a version of copying which only duplicates a certain class of terms.

Definition 15. *Let $\mathcal{C}$ be a set of terms. On cyclic terms extended with the binary symbol $+$, we define the rewrite relation $\xrightarrow[+\mathcal{C}]{}$ as follows:*

$$\begin{array}{lll} \langle M \mid x = N, E\rangle & \xrightarrow[+\mathcal{C}]{} \langle M\sigma \mid y = N\sigma, z = N\sigma, E\sigma\rangle & \textit{where } N \in \mathcal{C} \textit{ and} \\ & \qquad \sigma = [x := y + z] \textit{ for fresh variables } y \textit{ and } z & \\ y + z & \xrightarrow[+\mathcal{C}]{} y & \\ y + z & \xrightarrow[+\mathcal{C}]{} z & \end{array}$$

Table 3. The cyclic call-by-need lambda calculus $\lambda\circ_{\text{need}}$

$\beta\circ$:			
$(\lambda x.M)N$	$\xrightarrow[\beta\circ]{}$	$\langle M \mid x = N\rangle$	
Value Substitutions:			
$\langle C[x] \mid x = V, E\rangle$	$\xrightarrow[\text{es}_V]{}$	$\langle C[V] \mid x = V, E\rangle$	
$\langle M \mid x = C[x_1], x_1 = V, E\rangle$	$\xrightarrow[\text{is}_V]{}$	$\langle M \mid x = C[V], x_1 = V, E\rangle$	
Lift:			
$\langle M \mid E\rangle N$	$\xrightarrow[\text{lift}]{}$	$\langle MN \mid E\rangle$	
$M\langle N \mid E\rangle$	$\xrightarrow[\text{lift}]{}$	$\langle MN \mid E\rangle$	
$\lambda x.\langle M \mid E, VE\rangle$	$\xrightarrow[\text{lift}]{}$	$\langle \lambda x.\langle M \mid E\rangle \mid VE\rangle$	C1
Merge:			
$\langle\langle M \mid E\rangle \mid E'\rangle$	$\xrightarrow[\text{em}]{}$	$\langle M \mid E, E'\rangle$	
$\langle M \mid x = \langle N \mid E\rangle, E_1\rangle$	$\xrightarrow[\text{im}]{}$	$\langle M \mid x = N, E, E_1\rangle$	
Garbage collection:			
$\langle M \mid E, E'\rangle$	$\xrightarrow[\text{gc}]{}$	$\langle M \mid E\rangle$	C2
$\langle M \mid \rangle$	$\xrightarrow[\text{gc}]{}$	M	
Value Copying:			
M	$\xrightarrow[\text{cp}_V]{}$	N	
Naming:			
$C_{\text{safe}}[M\ N]$	$\xrightarrow[\text{name}]{}$	$C_{\text{safe}}[\langle x \mid x = M\ N\rangle]$	C3

C1: VE is non-empty and neither x nor a variable defined in E occurs free in VE;
C2: E' is non-empty and none of the variables defined in E' occur free in E or M;
C3: x is a fresh variable and the rule is *not* closed under contexts;

$$\begin{aligned} C_{\text{safe}} &::= C' \mid C[\lambda x.C'] \mid C[C'\ M] \mid C[M\ C'] \ ; \\ C' &::= \square \mid \langle C' \mid E\rangle \ ; \\ V &::= x \mid \lambda x.M \ ; \\ VE &::= x_1 = V_1, \cdots, x_n = V_n \ . \end{aligned}$$

If $M \xrightarrow[+\mathcal{C}]{}\!\!\!\!\rightarrow N$ and M nor N contain on occurrence of $+$ then

$$M \xrightarrow[\text{cp}_{\mathcal{C}}]{} N \ .$$

The cyclic call-by-need lambda calculus is defined in Table 3. We can define a notion of information content for it using the information content of the call-by-name calculus:

Definition 16. *Given the ARS $(\Lambda\circ, \xrightarrow[\lambda\circ_{\text{need}}]{})$ and the partial order $(\Lambda_\Omega, \leq_\Omega)$. The information content $\omega_{\lambda^\circ_{\text{need}}}$ of a term $M \in \Lambda\circ$ is given as follows:*

$$\omega_{\lambda^\circ_{\text{need}}}(M) = \text{lub}\{\omega_{\lambda\circ}(N) \mid M \xrightarrow[\text{es}]{}\!\!\!\!\rightarrow N\} \ .$$

The Böhm semantics of M with respect to $\omega_{\lambda^\circ_{\text{need}}}$ is denoted $\text{ABS}_{\text{need}}(M)$.

Due to the fact that the information content is infinite, we do not have skew confluence. Consider the following two reductions:

$$\begin{aligned} M \equiv \langle x \mid x = \lambda z.z\ y, y = \lambda z'.z'\ (x\ z')\rangle \\ \xrightarrow[\lambda\circ_{\text{need}}]{} \!\!\!\!\rightarrow \langle \lambda z.z\ y \mid y = \lambda z'.z'\ ((\lambda z.z\ y)\ z')\rangle \\ \xrightarrow[\lambda\circ_{\text{need}}]{} \!\!\!\!\rightarrow \langle \lambda z.z\ y \mid y = \lambda z'.z'\ (z'\ y)\rangle \equiv M_1 \end{aligned}$$

and

$$\begin{aligned} M \equiv \langle x \mid x = \lambda z.z\ y, y = \lambda z'.z'\ (x\ z')\rangle \\ \xrightarrow[\lambda\circ_{\text{need}}]{} \!\!\!\!\rightarrow \langle x \mid x = \lambda z.z\ (\lambda z'.z'\ (x\ z'))\rangle \equiv M_2\ . \end{aligned}$$

We have that $\omega_{\lambda^\circ_{\text{need}}}(M_1) = \text{ABS}_{\text{need}}(M_1)$, because the only redexes in M_1 and any of its reducts are value substitutions, which are performed as part of the computation of the information content. However, there cannot exist M_3 such that $M_2 \xrightarrow[\lambda\circ_{\text{need}}]{} \!\!\!\!\rightarrow M_3$ and $\omega_{\lambda^\circ_{\text{need}}}(M_1) \subseteq \omega_{\lambda^\circ_{\text{need}}}(M_3)$ because $\omega_{\lambda^\circ_{\text{need}}}(M_1)$ is infinite whereas the information content of any reduct of M_2 is finite. The reason is that in the unwinding of M we have an infinite number of β-redexes. When we rewrite M into M_1 we do all of those redexes at once and when we rewrite M into M_2 we destroy the opportunity to do them in one step. The consistency of $\lambda^\circ_{\text{need}}$ is guaranteed by the following theorem.

Theorem 3. *The ARSI* $((\Lambda\circ, \xrightarrow[\lambda\circ_{\text{need}}]{}), \omega_{\lambda\circ_{\text{need}}}, \mathcal{I}(\Lambda_\Omega, \leq_\Omega))$ *is ω-skew confluent.*

Uniqueness of Böhm semantics follows from this theorem. A direct proof of an equivalent statement can be found in [5].

7 Lifting Abstract Böhm Semantics

The Böhm semantics of both the cyclic call-by-name and call-by-need lambda calculi are closely related to unwinding. The information content for the cyclic call-by-name calculus can be seen as a two step process. First, one computes the normal form with respect to the $es\omega$ and $gc\omega$ rules given in Definition 14. Second, one applies the notion of information content associated to the lambda calculus [34], which consists of computing the normal form with respect to the $\beta\omega$ and $@\omega$ rules. The call-by-need information content of a term is the information content of the unwinding of the term. In this section, we study how to derive these notions of information content in an abstract setting.

We first introduce in Sect. 7.1 the notion of a finite basis and its properties. In Sect. 7.2 we consider extensions consisting of infinite objects over the basis and objects whose semantics are infinite objects. In Sect. 7.3, we consider abstract Böhm semantics of extensions.

7.1 Finite Basis

We start from an ARSI equipped with a partial order on its objects. The partial order should have a least element to ensure that its ideal completion [20] is an algebraic CPO. The finite elements of the ideal completion are the embeddings of the original partial order, so the information content of the finite elements is already defined. To be able to lift the notion of information content to infinite elements, we must require that the notion of information content and the rewrite relation are also monotonic with respect to the partial order. We formalize this starting point with the notion of a *finite basis*, for which we need one auxiliary definition: monotonicity of a rewrite relation with respect to an order.

Definition 17. *Given an ARS $(A, \rightarrow)$ and a partial order $(A, \leq)$ with a least element, we say that $\rightarrow$ is monotonic with respect to $\leq$ if*

$$a \rightarrow a' \wedge a \leq a'' \Rightarrow \exists a''' : a' \leq a''' \wedge a'' \rightarrow a''' \ .$$

The diagram of monotonicity is

$$\begin{tikzcd}
a \arrow[r, dash, "\leq"] \arrow[d] & a'' \arrow[d, dashed] \\
a' \arrow[r, dash, dashed, "\leq"'] & a'''
\end{tikzcd}$$

Definition 18 (finite basis). *A tuple $(A, \rightarrow, \leq_A, \omega, D, \leq_D)$ is a finite basis if*

- *$((A, \rightarrow), \omega, (D, \leq_D))$ is an ARSI with unique abstract Böhm semantics;*
- *ω is monotonic with respect to $\leq_A$: $a \leq_A a' \Rightarrow \omega(a) \leq_D \omega(a')$;*
- *$\rightarrow$ is monotonic with respect to $\leq_A$.*

Example 2. Let ω_{LL} stand for the function which given a lambda calculus term M returns the normal form of M with respect to the following ω_{LL}-rules [28]:

$$\begin{array}{ll}
(\lambda x.M)\, N & \xrightarrow[\omega_{\mathrm{LL}}]{} \Omega \\
\Omega\, M & \xrightarrow[\omega_{\mathrm{LL}}]{} \Omega
\end{array}$$

Then one has that $(\Lambda_\Omega, \xrightarrow[\beta]{}, \leq_\Omega, \omega_{\mathrm{LL}}, \mathcal{I}(\Lambda_\Omega), \subseteq)$ is a finite basis.

Next, we consider rewrite systems, referred to as extensions, whose objects have infinite objects as semantics. Moreover, we want these rewrite systems to mimic the behavior of their finite counterparts. For the cyclic lambda calculi mimicking the finite lambda calculus meant that the rewrite relation induced by the cyclic calculi was contained in the infinitary lambda calculus and that finite reductions in an approximation could be lifted to reductions in the extension. The equivalent of this involves lifting the reduction in a finite basis to a reduction on its ideal completion.

7.2 Extensions

The set of infinite terms can be seen as the ideal completion of the set of finite terms under the prefix order $\leq_\Omega$. Therefore, we treat the ideal completion of a set of objects as infinite objects. We then define a rewrite relation on ideals as follows: we say that an ideal rewrites to another if every sufficiently large element of the first ideal rewrites to an element of the second and every sufficiently large element of the second ideal can be obtained by rewriting an element of the first. This is in a way similar to how Corradini defined complete developments of an infinite set of redexes in an infinite term [17].

Definition 19. *Given a finite basis $\mathcal{A} = (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$. The operator $[\cdot\rangle : \mathcal{P}(A \times A) \to \mathcal{P}(\mathcal{I}(A) \times \mathcal{I}(A))$ is defined by $I_1[R\rangle I_2$ if*

$$\forall a \in I_1, \exists a' \in I_1, a'' \in I_2 : a \leq_A a' \; R \; a''$$

and

$$\forall a'' \in I_2, \exists a' \in I_2, a \in I_1 : a \; R \; a' \geq_A a'' \ .$$

If for $I \in \mathcal{I}(A)$, we denote $a \in I$ as $I \xrightarrow[\alpha]{} a$ then we can phrase this definition with the following two diagrams:

$$\begin{tikzcd}
I_1 \arrow[rr, dash, "{[R\rangle}"] \arrow[d, "\alpha"'] \arrow[dr, dashed, "\alpha"] & & I_2 \arrow[d, dashed, "\alpha"] & \text{and} & I_1 \arrow[rr, dash, "{[R\rangle}"] \arrow[d, dashed, "\alpha"'] & & I_2 \arrow[dl, dashed, "\alpha"'] \arrow[d, "\alpha"] \\
a \arrow[r, dash, dashed, "\leq_A"'] & a' \arrow[r, dash, dashed, "R"'] & a'' & & a \arrow[r, dash, dashed, "R"'] & a' \arrow[r, dash, dashed, "\geq_A"'] & a''
\end{tikzcd}$$

Example 3. Consider the infinitary lambda calculus term

$$(\lambda x.f\,x\,(f\,x\,(f\,x\,(\cdots))))(I\,I) \ .$$

The reduction of $I\,I$ to I can be matched:

$$(\lambda x.f\,x\,(f\,x\,(f\,x\,(\cdots))))(I\,I) \; [\xrightarrow[\beta]{}\rangle (\lambda x.f\,x\,(f\,x\,(f\,x\,(\cdots))))I$$

because
$$\begin{array}{rcl}
(\lambda x.\Omega)\,(I\,I) & \xrightarrow[\beta]{} & (\lambda x.\Omega)\,I \\
(\lambda x.f\,x\,(\Omega))\,(I\,I) & \xrightarrow[\beta]{} & (\lambda x.f\,x\,(\Omega))\,I \\
(\lambda x.f\,x\,(f\,x\,(\Omega)))\,(I\,I) & \xrightarrow[\beta]{} & (\lambda x.f\,x\,(f\,x\,(\Omega)))\,I \\
\vdots & \vdots & \vdots
\end{array}$$

It is obvious that for any single step we can do this. We also have

$$(\lambda x.f\,x\,(f\,x\,(f\,x\,(\cdots))))I \; [\xrightarrow[\beta]{}\rangle \; f\,I\,(f\,I\,(f\,I\,(\cdots)))$$

and

$$(\lambda x.f\,x\,(f\,x\,(f\,x\,(\cdots))))(I\,I)\ [\underset{\beta}{\rightarrow}\rangle\ f\,(I\,I)\,(f\,(I\,I)\,(f\,(I\,I)\,(\cdots)))\ .$$

By using $[\underset{\beta}{\twoheadrightarrow}\rangle$ rather than $[\underset{\beta}{\rightarrow}\rangle$, we can also develop infinite sets of redexes. The trick is to develop the finite subset of redexes present in suitable finite prefixes. For example, the fact that

$$f\,(I\,I)\,(f\,(I\,I)\,(f\,(I\,I)\,(\cdots)))\ [\underset{\beta}{\twoheadrightarrow}\rangle\ f\,I\,(f\,I\,(f\,I\,(\cdots)))$$

follows from

$$\begin{array}{rcl} f\,(I\,I)\,\Omega & \underset{\beta}{\twoheadrightarrow} & f\,I\,\Omega \\ f\,(I\,I)\,(f\,(I\,I)\,\Omega) & \underset{\beta}{\twoheadrightarrow} & f\,I\,(f\,I\,(\Omega)) \\ f\,(I\,I)\,(f\,(I\,I)\,(f\,(I\,I)\,(\Omega))) & \underset{\beta}{\twoheadrightarrow} & f\,I\,(f\,I\,(f\,I\,(\Omega))) \\ \vdots & \vdots & \vdots \end{array}$$

Now that we can lift any relation from an order to its ideal completion, it is logical to also extend information content from the order to the ideal completion. Because the information contained in an ideal can be infinite, we define the information content of an ideal as the downward closure of the set of information contents of its elements:

Definition 20. *Given a finite basis $\mathcal{A} = (A, \underset{A}{\rightarrow}, \leq_A, \omega, D, \leq_D)$ and $I \in \mathcal{I}(A)$. Let*

$$\omega_\infty(I) = \mathrm{lub}\{\omega(a) \mid a \in I\}\ .$$

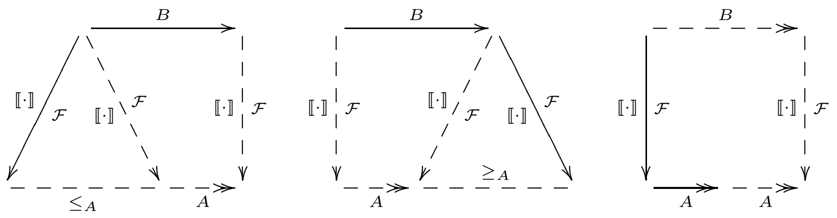
This is well-defined because of the monotonicity of ω with respect to $\leq_A$. Next, we consider the abstract version of an extension which contains objects whose semantics are infinite objects over the basis. Moreover, the reduction relation of the extension should contain a subset that can compute the semantics internally as a normal form. A good example is the call-by-name calculus, where the subset of just external substitution plus ω_{es} as the notion of information content can compute the unwinding. In general, we can always compute the semantics by using the empty (sub)set and the semantics as information content.

Definition 21. *A tuple $\mathcal{B} \equiv (B, \underset{B}{\rightarrow}, \underset{[\![B]\!]}{\longrightarrow}, \omega_{[\![\cdot]\!]}, [\![\cdot]\!])$ is an extension of a finite basis $\mathcal{A} \equiv (A, \underset{A}{\rightarrow}, \leq_A, \omega, D, \leq_D)$, if*

- *$(B, \underset{B}{\rightarrow})$ is an ARS;*
- *$\underset{[\![B]\!]}{\longrightarrow} \subseteq \underset{B}{\rightarrow}$;*
- *$((B, \underset{[\![B]\!]}{\longrightarrow}), \omega_{[\![\cdot]\!]}, \mathcal{I}(A, \leq_A))$ is an ARSI, such that*

$$\forall b \in B:\ \mathrm{ABS}(b) = [\![b]\!]\ .$$

The function $[\![\cdot]\!]$ takes the place of the unwinding. The function $\omega_{[\![\cdot]\!]}$ denotes the visible part of the unwinding. This visible part is used to restrict information



B
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\leq_A$
A
B
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\geq_A$
A
B
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
$\llbracket\cdot\rrbracket$
$\mathcal{F}$
A
A

Fig. 2. Soundness and completeness of an extension as commutative diagrams

content. For example, in the call-by-name calculus the visible part would be ω_{es} and in the call-by-need calculus it would be $\llbracket\cdot\rrbracket$. There must also be a subset of the rewrite relation, such that the semantics can be computed internally.

The cyclic lambda calculi are extensions of the lambda calculus in this sense because the semantics of a cyclic lambda term is its unwinding, which is an infinite term. For an extension to make sense, we require it to be sound and complete with respect to the basis [22, 1]. In other words, the extension cannot do more than the basis (soundness) and the extension can simulate everything the basis can do (completeness). To define soundness we use the $[\cdot\rangle$ operator. To define completeness we use a simple lifting property:

Definition 22. *Given a finite basis* $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ *and an extension* $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \xrightarrow[\llbracket B\rrbracket]{}, \omega_{\llbracket\cdot\rrbracket}, \llbracket\cdot\rrbracket)$. *Then,*

– $\mathcal{B}$ *is infinitarily sound with respect to* $\mathcal{A}$ *if*

$$s \xrightarrow[B]{} t \Rightarrow \llbracket s\rrbracket \; [\underset{A}{\twoheadrightarrow}\rangle \; \llbracket t\rrbracket \;\; ;$$

– $\mathcal{B}$ *is infinitarily complete with respect to* $\mathcal{A}$ *if*

$$\forall a, s : a \in \llbracket s\rrbracket \wedge a \underset{A}{\twoheadrightarrow} a' \Rightarrow \exists t, a'' : s \underset{B}{\twoheadrightarrow} t \wedge a' \underset{A}{\twoheadrightarrow} a'' \in \llbracket t\rrbracket \;\; .$$

In order to be able to draw diagrams, we use the fact that our notation allows us to denote $a \in \llbracket s\rrbracket$ by $s \xrightarrow[\llbracket\cdot\rrbracket]{\mathcal{F}} a$. Thus, the diagrams for soundness and completeness can be drawn as given in Fig. 2.

7.3 Abstract Böhm Semantics

We can now define an abstract Böhm semantics for extensions. The idea is simple: given an object, we compute the visible part of its semantics and apply the infinite extension of the information content of the basis to it.

Definition 23. *Given a finite basis* $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ *and an extension* $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \xrightarrow[\llbracket B\rrbracket]{}, \omega_{\llbracket\cdot\rrbracket}, \llbracket\cdot\rrbracket)$. *Define* $\omega_{\mathcal{B}} : B \to D$ *by*

$$\omega_{\mathcal{B}}(s) = \omega_\infty(\omega_{\llbracket\cdot\rrbracket}(s)) \;\; .$$

The only problem with the above definition is that the visible part and the base information content must fit together to form a proper notion of information content. First, we will give an example that shows that the result might not be an ARSI. Next, we will prove two propositions that help establishing that the result is an ARSI.

Example 4. Consider the cyclic extension

$$\begin{array}{ll} \mathsf{F}(\mathsf{F}(x)) & \to \langle x \mid x = \mathsf{F}(\mathsf{G}(x))\rangle \\ \mathsf{G}(\mathsf{G}(x)) & \to \langle x \mid x = \mathsf{G}(\mathsf{F}(x))\rangle \\ \langle x \mid x = M, E\rangle & \to \langle M \mid x = M, E\rangle \end{array}$$

$$\begin{array}{l} \langle \mathsf{F}(x) \mid x = M, E\rangle \to \mathsf{F}(\langle x \mid x = M, E\rangle) \\ \langle \mathsf{G}(x) \mid x = M, E\rangle \to \mathsf{G}(\langle x \mid x = M, E\rangle) \end{array}$$

and the functions ω, defined in Eq. 4, and ω_{es}, defined below:

$$\begin{array}{ll} \omega_{\text{es}}(x) & = x \\ \omega_{\text{es}}(f(M_1, \cdots, M_n) & = f(\omega_{\text{es}}(M_1), \cdots, \omega_{\text{es}}(M_n)) \\ \omega_{\text{es}}(\langle M \mid x_1 = M_1, \cdots, x_n = M_n\rangle) & = \omega_{\text{es}}(M)[x_1 := \Omega, \cdots, x_n := \Omega] \end{array}$$

The function $\omega \circ \omega_{\text{es}}$ is not a notion of information content:

$$(\omega \circ \omega_{\text{es}})(\mathsf{F}(\mathsf{F}(x))) = \omega(\mathsf{F}(\mathsf{F}(x))) = \mathsf{F}(\Omega)$$

and

$$(\omega \circ \omega_{\text{es}})(\langle x \mid x = \mathsf{F}(\mathsf{G}(x))\rangle) = \omega(\Omega) = \Omega \ .$$

So $\omega \circ \omega_{\text{es}}$ is not monotonic with respect to the reduction relation of the extension.

The following proposition assumes that the visible part of the semantics is the whole semantics.

Proposition 6. *Given a finite basis $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ and an extension $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \emptyset, [\![\cdot]\!], [\![\cdot]\!])$. If $\mathcal{B}$ is infinitarily sound with respect to $\mathcal{A}$ then $\mathcal{L} \equiv ((B, \xrightarrow[B]{}), \omega_{\mathcal{B}}, (D, \leq_D))$ is an ARSI.*

Proof. We have to establish that $\omega_{\mathcal{B}}$ is monotonic with respect to $\xrightarrow[B]{}$. That is, we need to show that if $s \xrightarrow[B]{} s'$ then $\omega_{\mathcal{B}}(s) \leq_D \omega_{\mathcal{B}}(s')$. Unfolding definitions we get

$$\omega_{\mathcal{B}}(s) = \omega_{\infty}([\![s]\!]) = \text{lub}\{\omega(a) \mid a \in [\![s]\!]\}$$

and

$$\omega_{\mathcal{B}}(s') = \text{lub}\{\omega(a) \mid a \in [\![s']\!]\} \ .$$

From the soundness of $\mathcal{B}$, we get that

$$\forall a \in [\![s]\!] : \ \exists a' \in [\![s]\!], a'' \in [\![s']\!] : \ a \leq_A a' \xrightarrow[A]{}\!\!\!\!\to a'' \ .$$

Since $\mathcal{A}$ is a finite basis, we have monotonicity of ω with respect to both $\leq_A$ and $\xrightarrow[A]{}$, so

$$\forall a \in [\![s]\!] : \ \exists a' \in [\![s]\!], a'' \in [\![s']\!] : \ \omega(a) \leq_D \omega(a') \leq_D \omega(a'') \ .$$

Hence

$$\{\omega(a) \mid a \in [\![s]\!]\} \subseteq \{\omega(a) \mid a \in [\![s']\!]\}$$

and also

$$\omega_{\mathcal{B}}(s) \leq_D \omega_{\mathcal{B}}(s') \ .$$

The problem with the counterexample is that the extended rewrite rules destroy a part of the unwinding, which was already part of the information content. The solution therefore is to require that every rewrite step in the extension preserves enough unwinding to compute at least the information content of the left-hand side:

$$\forall s, t \in B : s \xrightarrow[B]{} t \Rightarrow \exists I \in \mathcal{I}(A) : I \subseteq \omega_{[\![\cdot]\!]}(s) \wedge I \subseteq \omega_{[\![\cdot]\!]}(t) \wedge \omega_{\infty}(I) = \omega_{\infty}(\omega_{[\![\cdot]\!]}(s)) \ .$$

So the rewrite step from s to t preserves a part of the unwinding a and a is enough to compute the information content of s.

For the call-by-name calculus this property holds, because if $M \xrightarrow[\lambda\circ]{} N$ then it is either not a $\beta\circ$ step and $\omega_{[\![\cdot]\!]}(M) \leq_{\Omega} \omega_{[\![\cdot]\!]}(N)$ and we can take $a = \omega_{[\![\cdot]\!]}(M)$ or it is a $\beta\circ$ step $C[(\lambda x.P)\, Q] \xrightarrow[\beta\circ]{} C[\langle P \mid x = Q\rangle]$. In this case we can take $a = \omega_{[\![\cdot]\!]}(C[\Omega])$.

This idea give us our second proposition:

Proposition 7. *Given a finite basis* $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ *and an extension* $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \xrightarrow[[\![B]\!]]{}, \omega_{[\![\cdot]\!]}, [\![\cdot]\!])$. *If*

1. $\mathcal{B}$ *is infinitarily sound with respect to* $\mathcal{A}$*;*
2. $\forall s, t \in B : s \xrightarrow[B]{} t \Rightarrow \exists I \in \mathcal{I}(A) : I \subseteq \omega_{[\![\cdot]\!]}(s) \wedge I \subseteq \omega_{[\![\cdot]\!]}(t) \wedge \omega_{\infty}(I) = \omega_{\infty}(\omega_{[\![\cdot]\!]}(s))$*;*

then $\mathcal{L} \equiv ((B, \xrightarrow[B]{}), \omega_{\mathcal{B}}, (D, \leq_D))$ *is an ARSI.*

Proof. We have to establish that $\omega_{\mathcal{B}}$ is monotonic with respect to $\xrightarrow[B]{}$. That is, we need to show that if $s \xrightarrow[B]{} s'$ then $\omega_{\mathcal{B}}(s) \leq_D \omega_{\mathcal{B}}(s')$. By the second condition we can find $I \in \mathcal{I}(A)$, such that

$$I \subseteq \omega_{[\![\cdot]\!]}(s) \wedge I \subseteq \omega_{[\![\cdot]\!]}(s') \wedge \omega_{\infty}(I) = \omega_{\infty}(\omega_{[\![\cdot]\!]}(s)) \ .$$

From the fact that ω is monotone w.r.t. $\leq_A$ it is easy to prove that

$$I \subseteq \omega_{[\![\cdot]\!]}(s') \Rightarrow \omega_{\infty}(I) \leq_D \omega_{\infty}(\omega_{[\![\cdot]\!]}(s')) \ .$$

Hence

$$\omega_{\mathcal{B}}(s) \leq_D \omega_{\mathcal{B}}(s') \ .$$

Once we have established that the result is an ARSI then we can prove uniqueness of the abstract Böhm semantics. We establish a few lemmas first.

Lemma 1. *Given a finite basis* $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ *and an extension* $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \xrightarrow[[\![B]\!]]{}, \omega_{[\![\cdot]\!]}, [\![\cdot]\!])$ *such that*

1. $\mathcal{B}$ is infinitarily sound with respect to $\mathcal{A}$;
2. $\mathcal{B}$ is infinitarily complete with respect to $\mathcal{A}$;
3. $\mathcal{L} \equiv ((B, \xrightarrow[B]{}), \omega_{\mathcal{B}}, (D, \leq_D))$ is an ARSI.

We have:

$$\forall b_1, b_2 \in B, a \in A : b_1 \xrightarrow[B]{} \!\!\!\!\!\rightarrow b_2 \xrightarrow[\llbracket\cdot\rrbracket]{\mathcal{F}} a \Rightarrow \exists a', a'' \in A : b_1 \xrightarrow[\llbracket\cdot\rrbracket]{\mathcal{F}} a'' \xrightarrow[A]{} \!\!\!\!\!\rightarrow a' \geq_A a \enspace .$$

Proof. Follows by induction on the length of $b_1 \xrightarrow[B]{} \!\!\!\!\!\rightarrow b_2$ from soundness and monotonicity of $\xrightarrow[A]{}$ with respect to $\leq_A$. In a diagram:

$$\begin{tikzcd}
 & & b_1 \arrow[dl, two heads, "B"'] \arrow[d, dashed, "\llbracket\cdot\rrbracket"', "\mathcal{F}"] \\
 & \cdot \arrow[dl, two heads, "B"'] \arrow[d, dashed, "\llbracket\cdot\rrbracket"', "\mathcal{F}"] & a'' \arrow[d, dashed, two heads, "A"] \\
b_2 \arrow[d, "\llbracket\cdot\rrbracket"', "\mathcal{F}"] & \cdot \arrow[r, dash, dashed, "\leq_A"'] \arrow[d, dashed, two heads, "A"] & \cdot \arrow[d, dashed, two heads, "A"] \\
a \arrow[r, dash, dashed, "\leq_A"'] & \cdot \arrow[r, dash, dashed, "\leq_A"'] & a'
\end{tikzcd}$$

In this diagram, the left part is the induction hypothesis, the top right part is soundness and the bottom right part is monotonicity.

Lemma 2. *Given a finite basis $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ and an extension $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \xrightarrow[\llbracket B\rrbracket]{}, \omega_{\llbracket\cdot\rrbracket}, \llbracket\cdot\rrbracket)$ such that*

1. $\mathcal{B}$ is infinitarily sound with respect to $\mathcal{A}$;
2. $\mathcal{B}$ is infinitarily complete with respect to $\mathcal{A}$;
3. $\mathcal{L} \equiv ((B, \xrightarrow[B]{}), \omega_{\mathcal{B}}, (D, \leq_D))$ is an ARSI.

We have:

$$\forall b_1, b_2 \in B, a_1, a_1' \in A, d_1 \in D : b_1 \xrightarrow[B]{} \!\!\!\!\!\rightarrow b_2 \wedge b_1 \xrightarrow[\llbracket\cdot\rrbracket]{\mathcal{F}} a_1 \xrightarrow[A]{} \!\!\!\!\!\rightarrow a_1' \xrightarrow[\omega]{\mathcal{F}} d_1 \Rightarrow$$

$$\exists a_2, a_2' \in A, d_2 \in D : b_2 \xrightarrow[\llbracket\cdot\rrbracket]{\mathcal{F}} a_2 \xrightarrow[A]{} \!\!\!\!\!\rightarrow a_2' \xrightarrow[\omega]{\mathcal{F}} d_2 \wedge d_1 \leq_D d_2 \enspace .$$

Proof. By repeating the following diagram:

$$\begin{tikzcd}
\cdot \arrow[rr, "B"] \arrow[d, "\llbracket\cdot\rrbracket"', "\mathcal{F}"] & & \cdot \arrow[d, dashed, "\llbracket\cdot\rrbracket"', "\mathcal{F}"] \\
\cdot \arrow[r, dash, dashed, "\leq_A"'] \arrow[d, two heads, "A"'] & \cdot \arrow[r, dashed, two heads, "A"'] \arrow[d, dashed, two heads, "A"] & \cdot \arrow[d, dashed, two heads, "A"] \\
\cdot \arrow[r, dash, dashed, "\leq_A"'] \arrow[d, "\omega"', "\mathcal{F}"] & \cdot \arrow[d, dashed, "\omega"', "\mathcal{F}"] & \cdot \arrow[d, dashed, "\omega"', "\mathcal{F}"] \\
\cdot \arrow[r, dash, dashed, "\leq_D"'] & \cdot \arrow[r, dash, dashed, "\leq_D"'] & \cdot
\end{tikzcd}$$

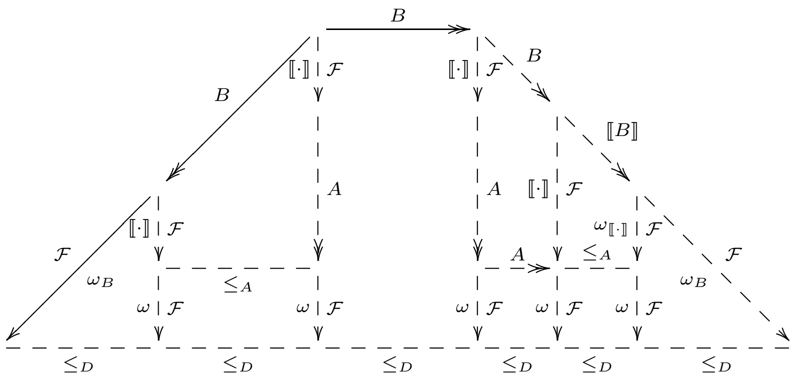
The diagram is built up from an instance of soundness on the top, two instances of monotonicity on the left bottom and an instance of ω-skew confluence of the basis on the right bottom.

Theorem 4. *Given a finite basis* $\mathcal{A} \equiv (A, \xrightarrow[A]{}, \leq_A, \omega, D, \leq_D)$ *and an extension* $\mathcal{B} \equiv (B, \xrightarrow[B]{}, \xrightarrow[\llbracket B \rrbracket]{}, \omega_{\llbracket\cdot\rrbracket}, \llbracket\cdot\rrbracket)$*. If*

1. $\mathcal{B}$ *is infinitarily sound with respect to* $\mathcal{A}$*;*
2. $\mathcal{B}$ *is infinitarily complete with respect to* $\mathcal{A}$*;*
3. $\mathcal{L} \equiv ((B, \xrightarrow[B]{}), \omega_{\mathcal{B}}, (D, \leq_D))$ *is an ARSI;*

then $\mathcal{L}$ *yields unique abstract Böhm semantics.*

Proof. The following diagram proves ω-skew confluence of $\mathcal{L}$:



B
B
$\llbracket\cdot\rrbracket \mid \mathcal{F}$
$\llbracket\cdot\rrbracket \mid \mathcal{F}$
B
$\llbracket B \rrbracket$
A
A
$\llbracket\cdot\rrbracket \mid \mathcal{F}$
$\llbracket\cdot\rrbracket \mid \mathcal{F}$
$\omega_{\llbracket\cdot\rrbracket} \mid \mathcal{F}$
$\mathcal{F}$
$\mathcal{F}$
ω_B
$\leq_A$
A
$\leq_A$
ω_B
$\omega \mid \mathcal{F}$
$\omega \mid \mathcal{F}$
$\omega \mid \mathcal{F}$
$\omega \mid \mathcal{F}$
$\omega \mid \mathcal{F}$
$\leq_D$
$\leq_D$
$\leq_D$
$\leq_D$
$\leq_D$
$\leq_D$

From left to right, we can obtain the sub-diagrams along the top by unfolding the definition of ω_B, the first lemma, the second lemma, completeness, the fact that $\llbracket b \rrbracket = \mathrm{ABS}_{\llbracket\cdot\rrbracket}(b)$ and again unfolding the definition of ω_B. The bottom rectangles are various applications of monotonicity.

This completes the presentation of the abstract lifting theory. It is a fairly complicated theory, but we think that using all of this machinery is easier than giving a direct proof of uniqueness of Böhm semantics in the extension. With a direct proof, we have to deal with effects of non-confluence for most of the proof. When we use the extension approach, we have a lot more statements to prove, but most of them can be proved in a context where the rewrite system is confluent. For example, in the case of the lambda calculus we can use the fact that the lambda calculus is confluent while proving that the lambda calculus is a basis. Proving that the cyclic lambda calculus is infinitarily sound for the $\beta\circ$ rule is made easy by the fact that the rewrite system consisting of external substitution, external lift and $\beta\circ$ is confluent. Proving soundness for the other rules is somewhat harder, because this involves a property of non-confluent rules. However, it should be noted that what we really have to do is to prove that those rules preserve the unwinding. Hence, the result can be reused for other calculi. Finally, to prove completeness it is sufficient to prove a confluent subset complete.

8 Conclusions

We have given a modified version of the notion of abstract Böhm tree, which is capable of dealing with infinite information content. To guarantee the uniqueness of the new form of abstract Böhm semantics, we have introduced a new member of the confluence family: ω-skew confluence, which is a generalization of the existing notion of skew confluence.

Also, we have developed an abstract framework to be able to construct a new system while deriving properties from the old one. In particular, we have defined the notion of a finite basis, consisting of an ARS and a notion of information content with suitable properties. We have defined how to extend the ARS to its ideal completion. We have defined an extension as an ARS, whose objects have the ideal completion of the basis as their semantics and we have defined when such an extension is sound and complete. We have shown that a finite basis and a sound and complete extension give rise to a notion of information content on the extension.

Acknowledgments. We thank the anonymous referees and the editors for their comments which allowed us to make substantial improvements to the original paper. The first author was supported by National Science Foundation grant number CCR-0204389.

References

1. Z. M. Ariola. Relating graph and term rewriting via Böhm models. *Applicable Algebra in Engineering, Communication and Computing*, 7(5), 1996.
2. Z. M. Ariola and Arvind. Properties of a first-order functional language with sharing. *Theoretical Computer Science*, 146:69–108, 1995.
3. Z. M. Ariola and S. Blom. Lambda calculi plus letrec. Technical Report CIS-TR-97-05, Department of computer and information science, University of Oregon.
4. Z. M. Ariola and S. Blom. Cyclic lambda calculi. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes in Computer Science*, pages 77–106. Springer Verlag, Sept. 1997.
5. Z. M. Ariola and S. Blom. Lambda calculi plus letrec. Technical Report IR-434, Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, Oct. 1997.
6. Z. M. Ariola and S. Blom. Skew confluence and the lambda calculus with letrec. *Annals of Pure and Applied Logic*, 117(1-3):95–168, october 2002.
7. Z. M. Ariola and M. Felleisen. The call-by-need lambda calculus. *Journal of Functional Programming*, 7(3), 1997.
8. Z. M. Ariola, M. Felleisen, J. Maraist, M. Odersky, and P. Wadler. The call-by-need lambda calculus. In *Proc. ACM Conference on Principles of Programming Languages*, pages 233–246, 1995.
9. Z. M. Ariola and J. W. Klop. Equational term graph rewriting. *Fundamentae Informaticae*, 26(3,4):207–240, 1996. Extended version: CWI Report CS-R9552.
10. Z. M. Ariola and J. W. Klop. Lambda calculus with explicit recursion. *Information and computation*, 139(2):154–233, Dec. 1997.

11. Z. M. Ariola, J. W. Klop, and D. Plump. Bisimilarity in term graphs rewriting. *Information and Computation*, 156(1/2):2–24, 2000.
12. H. Barendregt, T. Brus, M. van Eekelen, J. Glauert, J. Kennaway, M. van Leer, M. Plasmeijer, and M. R. Sleep. Towards an intermediate language based on graph rewriting. In *Proc. Conference on Parallel Architecture and Languages Europe (PARLE '87), Eindhoven, The Netherlands, Springer-Verlag LNCS 259*, June 1987.
13. H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics.* Elsevier, revised edition, 1984.
14. S. Blom. *Term Graph Rewriting - syntax and semantics.* PhD thesis, Vrije Universiteit Amsterdam, 2001.
15. S. Blom. Lifting Infinite Normal Form Definitions from Term Rewriting to Term Graph Rewriting. In *TERMGRAPH 2002 - International Workshop on Term Graph Rewriting*, 2002.
16. S. Blom. An approximation based approach to infinitary lambda calculi. In van Oostrom [33], pages 221–232.
17. A. Corradini. Term rewriting in CT_{Σ}. In M.-C. Gaudel and J.-P. Jouannaud, editors, *Proc. Colloquium on Trees in Algebra and Programming (CAAP '93), Springer-Verlag LNCS 668*, pages 468–484, 1993.
18. A. L. de Medeiros Santos. *Compilation by Transformation in Non-Strict Functional Languages.* PhD thesis, University of Glasgow, July 1995.
19. M. Dezani-Ciancaglini and E. Giovannetti. From Böhm's theorem to observational equivalences: an informal account. *Electronic Notes in Theoretical Computer Science*, 50(2), 2001.
20. M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.
21. G. Kahn and G. D. Plotkin. Concrete domains. *Theor. Comput. Sci.*, 121(1&2):187–277, 1993.
22. J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. The adequacy of term graph rewriting for simulating term rewriting. In M. R. Sleep, M. J. Plasmeijer, and M. C. D. J. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, pages 157–168. John Wiley & Sons, 1993.
23. J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. Infinitary lambda calculus. In *Proc. Rewriting Techniques and Applications, Kaiserslautern*, 1995.
24. J. R. Kennaway, J. W. Klop, M. R. Sleep, and F. J. de Vries. Transfinite reductions in orthogonal term rewriting systems. *Information and Computation*, 119(1), 1995.
25. J. Ketema. Böhm-like trees for term rewriting systems. In van Oostrom [33], pages 233–248.
26. J. Ketema and J. G. Simonsen. Infinitary combinatory reduction systems. In J. Giesl, editor, *RTA*, volume 3467 of *Lecture Notes in Computer Science*, pages 438–452. Springer, 2005.
27. J. W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume II, pages 1–116. Oxford University Press, 1992.
28. J.-J. Lévy. *Réductions Correctes et Optimales dans le Lambda-Calcul.* PhD thesis, Universite Paris VII, October 1978.
29. D. Plump. Term graph rewriting. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2: Applications, Languages and Tools, chapter 1, pages 3–61. World Scientific, 1999.
30. P. Severi and F.-J. de Vries. An extensional böhm model. In S. Tison, editor, *RTA*, volume 2378 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2002.

31. M. R. Sleep, M. J. Plasmeijer, and M. C. D. J. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice.* John Wiley & Sons, 1993.
32. Terese. *Term Rewriting Systems.* Cambdrige University Press, 2003.
33. V. van Oostrom, editor. *Rewriting Techniques and Applications, 15th International Conference, RTA 2004, Aachen, Germany, June 3-5, 2004, Proceedings*, volume 3091 of *Lecture Notes in Computer Science.* Springer, 2004.
34. C. Wadsworth. *Semantics And Pragmatics Of The Lambda-Calculus.* PhD thesis, University of Oxford, September 1971.
35. C. Wadsworth. The Relation between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda-Calculus. *Theoretical Computer Science*, 5, 1976.
36. P. Welch. Continuous Semantics and Inside-out Reductions. In *λ-Calculus and Computer Science Theory, Italy (Springer-Verlag LNCS 37)*, March 1975.