October 30, 2009

ON THE TERMINATION OF RUSSELL'S DESCRIPTION ELIMINATION ALGORITHM

CLEMENS GRABMAYER, JOOP LEO, VINCENT VAN OOSTROM, AND ALBERT VISSER

Abstract. In this paper we study the termination behaviour of Russell's description elimination rewrite system. We discuss certain claims made by Saul Kripke in his paper [Kri05].

1. Introduction

'On Denoting' ([Rus05]) is a central paper in the development of Analytical Philosophy. It articulated a solution to a number of problems like the informativeness of identity statements and the problem of negative existentials. Simultaneously, it provided a very influential paradigm of philosophical analysis. Closer study of the paper leads to many questions. What is its precise place in Russell's philosophical development? Does Russell's theory of descriptions really yield a descriptively adequate analysis of descriptions? And what demand of adequacy are we to impose? Are the problems the theory aims to solve really solved? Etcetera.

Well, even if the philosophical situation is riddled with questions and doubts, at least from a logical-technical point of view Russell's achievement is pretty clear, or is it? We think that indeed it is pretty clear, at least if we consider *the disambiguated version of Russell's theory treated in the modern way*: a description, in this view, is simply a binary quantifier. However, if we choose to view Russell's theory as a theory of ambiguous[1] descriptions where we may choose both the order of elimination and the scopes of the descriptions eliminated in the process of elimination, there is still some excitement to be found. Our paper is mainly about this more problematic case.

Let us briefly remind the reader what the theory of descriptions is. More detail will be provided later. Suppose we want to analyze in predicate logic the sentence: *the Q has property P*. For simplicity, we assume that P and Q are unary predicate symbols. A first step is to introduce a symbolic notation for our sentence. We write $P(\boldsymbol{\iota} x\, Qx)$.[2] Here $P(\cdot)$ is the context in which the description $\boldsymbol{\iota} x\, Qx$ occurs. We can eliminate the description by using a paraphrase. E.g., we might paraphrase our sentence by *Russell's paraphrase*, the formula $\exists x\ (\forall y\ (Qx \leftrightarrow x = y) \wedge Px)$. Alternatively, we may choose the logically equivalent paraphrase:

$$\exists x\, Qx \wedge \forall x, y\ ((Qx \wedge Qy) \rightarrow x = y) \wedge \forall x\ (Qx \rightarrow Px).$$

2000 *Mathematics Subject Classification.* 03A05, 03B10, 03B40, 03B65, 03B70, 68Q42.

Key words and phrases. denoting, descriptions, termination, term rewriting.

[1] In this article, the use of the word 'ambiguous' in expressions like 'ambiguous description' does *not* refer to an ambiguity of denoted object as in: "A phrase may denote ambiguously; e.g., 'a man' denotes not many men, but an ambiguous man." [Rus05, p. 479], but it refers to an ambiguity of scope in which descriptions within a sentence have to be understood. This latter ambiguity appears in 'On Denoting' as that between 'primary' and 'secondary' occurrences of descriptions [Rus05, p. 489].

[2] The symbol $\boldsymbol{\iota}$ is *atoi*. The suggestion is that it is inverse to Peano's singleton operator ι.

The first paraphrase has the advantage of brevity. The second one has the advantage of separating the three ingredients *existence*, *uniqueness* and *predication*.[3] In this paper we only consider paraphrases that are logically equivalent to Russell's paraphrase.

One could think that differences in perspicuity and brevity between paraphrases have little bearing on the description elimination process as a whole. However, we will see that this is not true. There are 'good' choices of paraphrase that guarantee that description elimination does always terminate, independent of the order in which descriptions are eliminated and what scopes are chosen in the elimination steps. And there are 'bad' choices of paraphrase for which, on some formulas, description elimination can go on forever with an unlucky pick of elimination steps.

Let us for the moment employ our first paraphrase. The problem of scope arises when we replace Px by a complex formula, e.g., $\neg Px$. The formula $\neg P(\iota x\, Qx)$ can be interpreted in two ways, to wit as $\exists x\ (\forall y\ (Qx \leftrightarrow x = y) \wedge \neg Px)$ and as $\neg\, \exists x\ (\forall y\ (Qx \leftrightarrow x = y) \wedge Px)$. In the first case, we took $\neg\, P(\cdot)$ as our context and in the second case we took $P(\cdot)$, as given by the subformula $P(\iota x\, Qx)$ as context. This choice of context is a choice of *the scope* of the description. This ambiguity was discussed by Russell in 'On Denoting'. In the face of the ambiguity, one can do two things. One can opt for a language in which the ambiguity is resolved. This was already done by Russell and Whitehead in Principia (e.g., [RW70]), who chose a rather awkward way of disambiguation as will be discussed in Section 3. Alternatively, one can make the disambiguating choices part of the process of elimination of descriptions.[4] We will mainly study this last option.

Our paper was inspired by a paper of Saul Kripke. In his paper [Kri05], Kripke discusses Russell's remarks on scope in 'On Denoting' ([Rus05]). Kripke's discussion makes the reader aware that even one hundred years after publication of Russell's great paper, the technicalities of eliminating descriptions and choosing scopes have never been satisfactorily investigated and described. On p. 1032–1033 of [Kri05], Kripke makes a fascinating claim. He says, discussing a predicate logical language with the Sheffer stroke as only propositional connective:

> At any rate, what looks like only *one* occurrence of ϕ and ψ in the conventional Russellian analysis of $\phi(\iota x\, \psi(x))$ is obviously becoming many, many occurrences of ϕ and ψ in the analysis. Now, suppose ϕ or ψ itself, or worse both, contain a description, or one or both contain many (occurrences of) descriptions. With an unfavourable choice of scopes there will be more (occurrences of) descriptions than there were originally in the analysand! The dangers involved in this situation I call the dangers of a *hydra*; that is where you are trying to unpack things you always have more occurrences of descriptions than you had before. [...]
>
> One theorem that I recall I proved is that with a bad choice of scopes, using a Sheffer stroke or its dual will allow for a hydra. Some paths of the tree will go on infinitely and never come to an end.

[3] This separation is useful in the context of the Russell-Strawson discussion, where one could say, very roughly, that Strawson claims that the first two conjuncts are not asserted but rather function as a presupposition.

[4] We have the further possibility to *fix* an elimination strategy like the narrow-scope convention.

Kripke's claim is certainly not immediately clear. It is surely true that one can get more and more occurrences of descriptions for certain choices of the paraphrase. However, in many seemingly analogous cases we do have termination nevertheless, e.g. for the rewrite system for the Hydra battle [Tou98], since the terms one obtains are simpler in some specifiable sense. It turns out that in the present situation the crux is, as becomes clear from Kripke's further remarks, that he considers the case where one chooses at each elimination step both the description to be eliminated and a corresponding scope.[5]

In one respect Kripke's discussion is, perhaps, a bit misleading. The reader may get the impression that the phenomenon of non-termination can only happen for certain choices of *the repertoire of propositional connectives*, like the Sheffer stroke. This is not so. We show that, for every expressively complete choice of connectives, one can find paraphrases that allow non-terminating reduction sequences *and* we can find paraphrases that give us terminating reduction systems.

The main aim of this paper is to provide a reasonably complete picture of the syntactic side of description elimination systems for predicate logic, in particular with respect to termination aspects of the elimination process. More specifically, we show the following five facts[6], the first four of which concern a language with scope ambiguous description terms for which a schematic paraphrase $A[P, Q]$ for $P(\boldsymbol{\iota} x\, Qx)$ that is equivalent to Russell's paraphrase has been fixed:

- F1. Suppose that the schematic paraphrase $A[P, Q]$ contains more than one occurrence of P. Then the description elimination system is not terminating: we can find infinite reduction sequences. (Theorem 5.7)
- F2. Suppose that the schematic paraphrase $A[P, Q]$ contains only one occurrence of P. Then the description elimination system is terminating. (Theorem 5.11)
- F3. In every other language with an expressively complete repertoire of propositional connectives, we can find a schematic paraphrase $A'[P, Q]$ that is equivalent to Russell's paraphrase, and in which P only occurs once. For such a choice of paraphrase, the description elimination system is terminating, by F2. (Theorem 6.10)
- F4. Independently of the form of the schematic paraphrase, the description elimination system is normalising, and there exist computable, normalising, and deterministic strategies for it (such as the narrow-scope[7] strategy). (Theorem 5.13)
- F5. For a language with scope disambiguated descriptions (i.e., descriptions considered as binary quantifiers), we have termination and unique normal forms for the description elimination system, independent of the precise choice of the (schematic) paraphrase. (Theorem 5.17)

To explain (F2), we consider Kripke's example of the Sheffer stroke. Kripke's paraphrase is a translation of the paraphrase $\exists x\,(\forall y\,(Qy \leftrightarrow x = y) \wedge Px)$ into the Sheffer-only language using the following definitions.

- $\neg\, A := A \mid A$.

[5] A concrete example of non-termination of description elimination will be given on page 13.

[6] The terminology of the list below will be explained in detail in the paper. The reader is asked to consider these formulations just as suggestive at this stage.

[7] Besides the terminology 'wide-scope'/'narrow-scope' strategy, in the literature also the expressions 'large-scope'/'small-scope' strategy are used.

- $A \vee B := \neg\, A \mid \neg B$.
- $A \wedge B := \neg\, (A \mid B)$.
- $A \to B := A \mid \neg B$.
- $A \leftrightarrow B := (A \to B) \wedge (B \to A)$.

By doing this, one obtains a paraphrase[8] $A[P, Q]$ with two occurrences of P, with respect to which our result (F1) confirms Kripke's claim. However, this is *praeter necessitatem*: if we define $\neg\, A$ as $A \mid \forall x\, x = x$, we are doing much better. A reasonably efficient example of a schematic paraphrase of $P(\boldsymbol{\imath} x\, Q)$, using just the Sheffer stroke and universal quantification, is as follows:

$$(\dagger) \quad \forall x\, (\, \forall y\, ((Qy \mid x = y) \mid ((Qy \mid \top) \mid (x = y \mid \top))) \mid Px\,) \mid \top\ ,$$

where we define[9] $\top := \forall z\, z = z$. By a polarity argument one can prove that the number of occurrences of Q in $(\dagger)$ is optimal. The representation $(\dagger)$ is, by (F2), good enough to guarantee termination for the ambiguous description elimination algorithm. The point of (F3) is that, for other choices of the connectives, under quite general circumstances, we can always find such good paraphrases.

Willard van Orman Quine in [Qui37] suggests an alternative repertoire of logical constants for set theory, to wit abstraction and the subset relation. This shows that one can imagine setups that do not fit our framework, i.e., setups that do not have the usual division of labor between the propositional connectives and the quantifiers. We did not explore such alternative ideas.

This paper has a second aim. We want to make philosophers acquainted with *term rewriting* and to illustrate the importance of that field. Term Rewriting is a general theory of syntactical transformations, with intuitive concepts and of wide applicability. It has its origin in the work of Church and Curry, and, more generally, in proof theory. Term Rewriting offers a better, richer view of syntax than one usually gains in reading an introduction to Logic. The philosophically salient notion of *occurrence* has its proper home in a theory of syntactical transformations, where, for example, one develops a methodology to trace the fate of occurrences under syntactical transformations. But the importance of term rewriting is not restricted to this; it extends to proof theory and cut-elimination and provides insight into the fundamental notion of computation. Our second aim is, we submit, fully in the spirit of Evert Beth, who stressed throughout his career the importance, for logicians and philosophers, to learn what is going on in neighbouring fields.

Overview The layout of the paper is as follows. We start with an introduction to the problem of eliminating ambiguous definite descriptions (Section 2), and provide some historical remarks on attempts to disambiguate such descriptions (Section 3). An introduction to higher-order term rewriting is given (Section 4), which is subsequently employed to analyse both the ambiguous and disambiguated description elimination from a rewriting perspective (Section 5). We dedicate the one but last section (Section 6) to establish F3, by means of a theorem of propositional logic which is interesting on its own.

[8]Russell and Whitehead in the introduction to the second edition of Principia Mathematica (see e.g. [RW70]) use precisely this translation; and so does Willard van Orman Quine, in his paper [Qui37]. However, Russell and Whitehead provide in $*14$ of Principia Mathematica a disambiguation device. Quine's specification of the elimination of descriptions explicitly demands the narrow-scope strategy, so that one cannot generate infinite reduction paths in Quine's setup.

[9]Clearly, any fixed sentential theorem would do in the role of $\top$. Thus, in a theory without identity like the one presented in [Qui37], we can still find a 'good' definition of negation.

2. Descriptions

In this section, we take a first step in making more precise what the theory of descriptions is, in a classical context. In Section 5, we will adapt these ideas to fit the term rewriting perspective.

An *inclusive signature* Σ for predicate logic is given by a set of predicate symbols, a set of function symbols, a set of atomic propositional function symbols, a set of propositional connectives, a set of quantifiers and a set of term forming operators. All these sets are supposed to be mutually disjoint. We have an arity function on all sets of symbols. We allow arity 0. We only admit unary propositional functions. Let an infinite set of variables be given.

The language for the given inclusive signature is specified as follows.

- $t ::= v \mid f(t, ..., t) \mid \alpha(F, \ldots, F)$.
  Here v ranges over the variables, f ranges over the function symbols, where the sequence of terms they take as inputs is assumed to be of the right arity. The metavariable α ranges over the term forming operators.
- $F ::= G \mid (\lambda v\, A)$.
  Here G ranges over the atomic propositional function symbols.
- $A ::= P(t, \ldots, t) \mid Ft \mid \gamma(A, \ldots, A) \mid \theta(F, \ldots, F)$.
  Here P ranges over the predicate symbols, γ ranges over the propositional connectives, and θ ranges over the quantifiers.

A *basic signature* Σ_0 consists of a set of predicate symbols, an expressively complete repertoire of propositional connectives and either one or two of the unary quantifiers $\forall$ and $\exists$. The signature $\Sigma_0(\boldsymbol{\imath})$ is Σ_0 extended by the unary term forming operation $\boldsymbol{\imath}$. The signature $\Sigma_0(\boldsymbol{\imath})$ is Σ_0 extended by the binary quantifier $\boldsymbol{\imath}$. The signature of contexts over Σ_0 is $\Sigma_0(\boxdot, \odot)$, where $\boxdot$ and $\odot$ are primitive propositional function symbols. A *context* is a formula C in the language of contexts.

The language based on $\Sigma_0(\boldsymbol{\imath})$ is a language for *ambiguous descriptions*. A Russell-style elimination step for this language has the form:

$$G[y := \boldsymbol{\imath}F] \;\mapsto\; C[F, \lambda y.G]\,,$$

given that $\boldsymbol{\imath}F$ is substitutable for y in G, and that $C[\boxdot, \odot]$ is a context in which F and $\lambda y.G$ are substitutible for $\boxdot$ and $\odot$, respectively. A sample elimination step could look like this:

$$G[y := \boldsymbol{\imath}F] \;\mapsto\; \exists x\,(\forall y\,(Fy \leftrightarrow x = y) \wedge Gx)\,.$$

Remark 2.1. In a language which includes λ-calculus the elimination step could be split into two steps:

$$\begin{aligned} G[y := t] &\mapsto (\lambda y.G)t\,,\\ G\boldsymbol{\imath}F &\mapsto C[F, G]\,. \end{aligned}$$

Note that the ambiguity then is located in the fact that we allow β-expansion (in the first half of the step). This corresponds to the higher-order term rewriting perspective on description elimination to be presented in Sections 4 and 5. ❑

The language based on $\Sigma_0(\boldsymbol{\imath})$ is a language of *disambiguated descriptions*. A Russell-style elimination step for $\Sigma_0(\boldsymbol{\imath})$ has the form:

$$\boldsymbol{\imath}(F, G) \;\mapsto\; C[F, G]\,,$$

given that $C[\boxdot, \odot]$ is a context and F and G are substitutable in C for $\boxdot$ and $\odot$, respectively. Suppose that Σ_0 is based on the usual repertoire of quantifiers and propositional connectives, including $\leftrightarrow$. The elimination step corresponding to the sample step above is:

$$\boldsymbol{\imath}(F, G) \;\mapsto\; \exists x\,(\forall y\,(Fy \leftrightarrow x = y) \wedge Gx)\,.$$

3. Disambiguating Ambiguous Descriptions: the History

Apart from various philosophical issues, Russell's theory of descriptions enables us to employ descriptions as a convenient notational device. In a mathematical context, however, we want to use descriptions non-ambiguously. Thus, we have either to change or enrich the basic notational device. From these considerations, one sees that it is not surprising that disambiguated versions were provided at a very early stage.

Russell and Whitehead provide in *Principia* ([RW70]) a disambiguation device that works as follows. They place the description between square brackets directly before the formula in which the description has scope. Thus, they write $[\boldsymbol{\imath}x\,Qx]P(\boldsymbol{\imath}x\,Qx)$ to signal that $\boldsymbol{\imath}x\,Qx$ has scope in $P(\boldsymbol{\imath}x\,Qx)$. Regrettably Russell and Whitehead failed to do the necessary homework, so that we can only guess at the further rules for coherent use of their convention.

A first point is that the elimination conventions are simply not clear. Consider the following examples.

(1) $[\boldsymbol{\imath}x\,Qx]\,P$, where P is a 0-ary predicate symbol. Does this reduce to P, or to $\exists x\,(\forall y\,(Qy \leftrightarrow x = y) \wedge P)$?
(2) $[\boldsymbol{\imath}x\,Qx][\boldsymbol{\imath}x\,Qx]\,P(\boldsymbol{\imath}x\,Qx)$. Do both occurrences of $[\boldsymbol{\imath}x\,Qx]$ link to the third occurrence of $\boldsymbol{\imath}x\,Qx$?
(3) $[\boldsymbol{\imath}x\,Qx]\,P(\boldsymbol{\imath}x\,Qx, \boldsymbol{\imath}x\,Qx)$. Does $[\boldsymbol{\imath}x\,Qx]$ link both to the second and third occurrence of $\boldsymbol{\imath}x\,Qx$?

It is clear that definite answers can be given to these questions, but something needs to be said. However, worse is to come. Consider $[\boldsymbol{\imath}x\,Qxy]\,\exists y\,P(\boldsymbol{\imath}x\,Qxy)$. If we follow Russell and Whitehead's instructions this reduces to

$$\exists x\,(\forall z\,(Qzy \leftrightarrow x = z) \wedge \exists y\,Px).$$

But this is clearly wrong, since the fact that the last occurrence of y in our original formula is bound by an intervening existential quantifier is ignored. Of course, we can formulate an appropriate restriction on the elimination step, but this still leaves us with the embarrassing fact that there is simply no context, under Russell and Whitehead's convention, for the formula $\exists y\,P(\boldsymbol{\imath}x\,Qxy)$ which would lead to elimination of $\boldsymbol{\imath}x\,Qxy$. We also note that we get similar problems (and more) with $[\boldsymbol{\imath}x\,Q(x, \boldsymbol{\imath}y\,Ry)][\boldsymbol{\imath}y\,Ry]\,P(\boldsymbol{\imath}x\,Q(x, \boldsymbol{\imath}y\,Ry))$.

Finally, the notation does not preserve meaning if we intersubstitute logically equivalent formulas. E.g., the formula $[\boldsymbol{\imath}x\,Px]\,Q(\boldsymbol{\imath}x\,Px)$ is not equivalent to the formula $[\boldsymbol{\imath}x\,Px]\,Q(\boldsymbol{\imath}x\,(Px \wedge Px))$. Even worse, meaning is not preserved by α-conversion. E.g., the formula $[\boldsymbol{\imath}x\,Px]\,Q(\boldsymbol{\imath}x\,Px)$ is not equivalent to the formula $[\boldsymbol{\imath}x\,Px]\,Q(\boldsymbol{\imath}y\,Py)$. Further explication of what is intended, would eliminate the problem. We must add a rule to the effect that the linked occurrences of the description term should be treated as the same occurrence when we substitute equivalents or when we α-convert. So, if we want to substitute $(Px \wedge Px)$ for some occurrences of Px in $[\boldsymbol{\imath}x\,Px]\,Q(\boldsymbol{\imath}x\,Px, \boldsymbol{\imath}x\,Px)$, the only way in which this can be done is the one

resulting in $[\boldsymbol{\imath}x\,(Px \wedge Px)]\,Q(\boldsymbol{\imath}x\,(Px \wedge Px), \boldsymbol{\imath}x\,(Px \wedge Px))$. This rule would complicate our elimination algorithm in that it could force us to eliminate, say, one hundred linked descriptions at the same time.

We are sure that with sufficient work all these problems could be solved. But why go through the trouble if we already have well understood conventions for the device of treating $\boldsymbol{\imath}$ as a binary quantifier? Of course, there are subtleties here too, *but these are simply subsumed under the well-studied general rules for handling free and bound variables.* Stephen Neale, in his book [Nea90], opts for binary quantifier notation. Curiously, Neale does not comment at all on the many deficiencies of the explanation that Russell and Whitehead give of their notation.

Other ways of disambiguation are possible. For example, Quine in his paper [Qui37], follows a different strategy of disambiguation. He demands that we always eliminate descriptions using the narrow-scope convention.[10]

A term rewriting perspective on the issues raised here is given in the following two sections. The interest in this is in a precise analysis of the issues and in their resolution by means of tools offered by rewriting theory.

4. Term Rewriting, an Ultra-Short Introduction

In this paper we will employ term rewriting to analyse Russell's description operator, in particular the question whether its elimination process terminates. We model predicate logic by means of a term signature, in the manner of Church's simple theory of types, describe the elimination of the various description operators by means of term rewrite rules, and employ (higher-order) rewriting theory [Ter03] to establish properties of the elimination process. To that end, we now first give an ultra-short introduction to the field of term rewriting, meant for philosophers and logicians, who we assume to be already familiar with Church's simple theory of types [Chu40, Wol93]. For a more extensive overview of (higher-order) term rewriting and pointers to the literature, we refer the reader to [Ter03].

Term rewriting has its origins both in proof theory and in algebra. In proof theory, Church's λ-calculus is a term rewriting system having the β-computation rule as rewrite rule. In algebra, Knuth–Bendix completion aims at deciding equational theories by orienting equations into rewrite rules. More generally, rewrite rules have been used to model and study stepwise processes as diverse as cut-elimination in proof theory, deduction in logic, recursion in computation, and evolution in dynamical systems. In this paper, rewrite rules will be employed to model and study the elimination process of the description operator.

Term rewriting systems can be viewed as (usually finite) specifications of their (usually infinite) associated rewrite relations, and term rewriting theory aims at establishing properties of the latter by means of the former. For instance, Church and Rosser's classical result that the λ-calculus is consistent in the sense that there are λ-terms which are *not* convertible by the β-rewrite relation is, from a term rewriting point of view, a direct consequence of the fact that the β-rewrite rule is *left-linear* and *non-overlapping*.[11] For another example, term rewriting theory yields that an equational theory is decidable if orienting its equations gives rise

[10] In other publications, like [Qui96], Quine defines descriptions via set abstraction. We will not discuss this variant in this paper.

[11] The emphases in this paragraph are meant to draw attention to several useful notions provided by term rewriting theory. Although some of them will be defined below, most will not, and

to a rewriting system such that the *critical pairs* between its rewrite rules are *confluent* and its rules are contained in some *recursive path order*. In this article, we will call upon term rewriting theory to establish that the elimination process of the ambiguous description operator lacks good properties as a consequence of the corresponding term rewrite rule not being a *pattern* rule, but that, on the other hand, the elimination process of the disambiguated description operator is well-behaved as a consequence of the corresponding rule being a *non-creating orthogonal pattern* term rewrite rule.

Concretely, a term rewriting system consists of a signature and a collection of rewrite rules over that signature. An example of a (first-order) term rewriting system, expressing addition of natural numbers, is given by the (first-order) signature[12] $\{0/0, S/1, A/2\}$ and the rewrite rules

$$\begin{array}{rcl} A(x,0) & \to_0 & x \\ A(x,S(y)) & \to_S & S(A(x,y)) \end{array}$$

Notation 4.1. Both rewrite rules and logical implications play an important role in this paper. Unfortunately, the standard notation for both is the arrow $\to$. In order to disambiguate these notations, we will affix a subscript, usually some symbol being eliminated, to the arrow in case of rewrite rules and derived notions. ❑

The rewrite relation associated to a term rewriting system is obtained by substituting arbitrary terms for the variables in a term rewrite rule, and allowing for its application at arbitrary positions in terms. For rewrite steps, i.e. pairs of terms belonging to the rewrite relation, we employ the same notation as for rewrite rules. For instance, the rewrite step

$$S(A(S(0),0)) \to_0 S(S(0))$$

is obtained by applying the first rule above at position 1, substituting $S(0)$ for x. The result $S(S(0))$ is a normal form; no rules apply to it. Rewriting theory yields that the equational theory corresponding to this term rewriting system is decidable, i.e. it is decidable whether one term can be converted by means of a series of rewrite steps into another term. This follows from the fact that the rewriting system has no *critical pairs* and its rules are contained in some *recursive path order*, a fact routinely established by the nowadays available termination tools, e.g. by the Tyrolean Termination Tool T$_{\text{T}}$T$_2$ [KSZM09].

Note that this term rewriting system has two rewrite rules, not two rule schemata. Thus the system is finite, a fact which is essential for automation. Stated differently, the variables x, y in the rules are object-variables, not meta-variables. These object-variables are instantiated by a process of substitution which resides at object-level, i.e. is part of the term rewriting formalism, not at meta-level. Whereas in the case of the above *first-order* term rewriting system this substitution process is simple, it is more complex in the case of *higher-order* term rewriting systems used for describing transformations on terms with binders, and the latter systems are the ones needed in this paper to describe the description operator. We will now first explain this complex process on a simple example.

---

only serve to illustrate the methodology of proving properties of rewrite relations via properties of term rewriting systems.

[12]We employ the notation $A/2$, borrowed from Prolog, to express that A has arity 2.

In higher-order term rewriting the signature consists of symbols to which a simple type has been assigned indicating their 'functionality', and higher-order terms are terms over this signature constrained by these types. More precisely, higher-order terms are $\beta\eta$-equivalence classes of simply typed λ-terms over the simply typed signature. An example of a higher-order signature[13] is $\{0 : \mathsf{nat}, S : \mathsf{nat} \to \mathsf{nat}, A : \mathsf{nat}{\to}\mathsf{nat}{\to}\mathsf{nat}, \Sigma{:}(\mathsf{nat}{\to}\mathsf{nat}){\to}\mathsf{nat}{\to}\mathsf{nat}\}$, and some examples of higher-order terms over this signature are 0, $S\,0$, and $\Sigma\,S\,((\lambda x.\,S\,x)\,0)$. Note that $S\,0\,0$ is not a higher-order term over this signature since it is not well-typed; S can take one argument (of type nat) not two. In fact it is customary to take *long $\beta\eta$-normal forms* as representatives of higher-order terms, i.e. to take normal forms with respect to the following β-reduction and restricted η-expansion rule schemata[14] as representatives:

$$\begin{array}{rcl}(\lambda x.\,M)\,N & \to_\beta & M[x{:=}N] \\ M & \to_{\bar\eta} & \lambda x.\,M\,x \quad \text{if no } \beta\text{-redex is created}\end{array}$$

For instance, the representative of $\Sigma\,S\,((\lambda x.\,S\,x)\,0)$ is found by reducing it with respect to these two schemata:

$$\Sigma\,S\,((\lambda x.\,S\,x)\,0) \to_\beta \Sigma\,S\,(S\,0) \to_{\bar\eta} \Sigma\,(\lambda x.\,S\,x)\,(S\,0)$$

with the final term the representative since it is in long $\beta\eta$-normal form: no further β-step is possible and any η-expansion step would create a β-redex, i.e. would enable a β-step. Note that just from the long $\beta\eta$-normal form one can reconstruct that Σ takes two arguments, $\lambda x.\,S\,x$ and $S\,0$, the first of which is of function type, something which could not be read off before. More generally, a long $\beta\eta$-normal form is a good and informative representative: within its $\beta\eta$-equivalence class it is unique and each symbol in it is applied to the number of arguments as given by its type. This last fact justifies employing functional instead of applicative notation for these symbols, as we will do below, e.g. writing $\Sigma(\lambda x.\,S(x), S(0))$ to denote $\Sigma\,(\lambda x.\,S\,x)\,(S\,0)$.

An example of a higher-order term rewriting system over this signature, expressing summation of functions on natural numbers, is given by the rules

$$\begin{array}{rcl}\Sigma(\lambda x.\,F(x), 0) & \to_0 & F(0) \\ \Sigma(\lambda x.\,F(x), S(y)) & \to_S & A(F(S(y)), \Sigma(\lambda x.\,F(x), y))\end{array}$$

where the variables are typed as $x, y : \mathsf{nat}$ and $F : \mathsf{nat} \to \mathsf{nat}$, with x a bound variable and F, y free variables. The associated rewrite relation is obtained from the rewrite rules as before but now allowing the substitution of higher-order terms. For instance, the rewrite step

$$\Sigma(\lambda x.\,S(0), S(0)) \to_S A(S(0), \Sigma(\lambda x.\,S(0), 0))$$

is obtained by substituting $\lambda z.\,S(0)$ and 0 for the free variables F and y in the second rule above. That the higher-order substitution process is more involved, is witnessed in the example by the substitution of $\lambda z.\,S(0)$ for the first occurrence of F in the right-hand side. It leads, implicitly, to the erasure of its argument $S(y)$ (with 0 substituted for y).

In fact, rule application can be seen as a three-phase process, consisting of matching (finding the appropriate substitution to instantiate the left-hand side with),

[13]The first-order function symbols of the previous example can be seen as being embedded into this higher-order signature by assigning types of shape $\mathsf{nat} \to \ldots \to \mathsf{nat} \to \mathsf{nat}$ to them.

[14]These are schemata since the variables M,N now are meta-variables, not object-variables.

replacement (literally replacing the left-hand side by the right-hand side), and substitution (applying the substitution to the right-hand side), where only the second phase employs an actual rewrite rule whereas the first and third-phase only choose a different representative of a higher-order term within iths $\beta\eta$-equivalence class. For instance, the above step can be decomposed as follows:

(matching) First, β-expansion makes the left-hand side appear literally:

$$\Sigma(\lambda x.\, S(0), S(0)) \mathrel{\twoheadleftarrow_\beta} (\lambda F\, y.\, \underline{\Sigma(\lambda x.\, F(x), S(y))})\, (\lambda z.\, S(0))\, 0$$

(replacement) Next, the left-hand side is literally replaced by its right-hand side:

$$(\lambda F\, y.\, \underline{\Sigma(\lambda x.\, F(x), S(y))})\, (\lambda z.\, S(0))\, 0 \to_S (\lambda F\, y.\, \underline{A(F(S(y)), \Sigma(\lambda x.\, F(x), y))})\, (\lambda z.\, S(0))\, 0$$

(substitution) Then, β-reduction substitutes the arguments in the right-hand side:

$$(\lambda F\, y.\, \underline{A(F(S(y)), \Sigma(\lambda x.\, F(x), y))})\, (\lambda z.\, S(0))\, 0 \mathrel{\twoheadrightarrow_\beta} A(S(0), \Sigma(\lambda x.\, S(0), 0))$$

The structure $(\lambda F\, y.\, \ldots)\, (\lambda z.\, S(0))\, 0$ in the above would conventionally be expressed by means of a higher-order substitution $\ldots[F{:=}\lambda z.\, S(0), y{:=}0]$. In higher-order rewriting, this complex higher-order substitution process is entirely taken care of by the β-reduction of the simply typed λ-calculus, whence the slogan: higher-order rewriting is rewriting modulo simply typed λ-calculus. We will put the above decomposition to good use in our analysis of the various elimination strategies for description operators below.

5. Eliminating descriptions by term rewriting

We present both the language of ambiguous descriptions based on $\Sigma_0(\boldsymbol{\imath})$ and that of disambiguated descriptions based on $\Sigma_0(\boldsymbol{\imath})$ as higher-order term rewriting systems whose rules model their respective elimination steps. We show that the lack of good properties of the former already observed above, is reflected in the lack of good properties of its higher-order term rewriting system presentation. Vice versa, addressing these deficiencies naturally leads to the disambiguated description operator, the well-behavedness of which follows directly from the application of standard higher-order term rewriting theory. To express both languages we proceed in the style of Church [Chu40] as indicated in Section 4, keeping the same basic signature Σ_0 and assigning appropriate types to its elements, i.e. to its connectives, function symbols, etc., and the respective description operators $\boldsymbol{\imath}$ and $\boldsymbol{\imath}$.

Notation 5.1. Both terms and formulas of predicate logic alike will be expressed as higher-order terms over that typed signature. In order to avoid confusing terms of predicate logic with the (more encompassing) higher-order terms, we will refer to the latter in this section as *expressions*, and use e to range over them. ❑

We employ two basic types trm and prp for representing terms and formulas respectively. Denoting the type of functions from type τ to type σ by $\tau \to \sigma$, and letting $\to$ associate to the right, we abbreviate $\tau \to \ldots \to \tau \to \sigma$, with n occurrences of τ as indicated, to $\vec{\tau}^{\,n} \to \sigma$. As usual, $e : \tau$ expresses that the type τ is assigned to the expression e. We assign types to the symbols in the signature of the language of descriptions as follows:

- $v : \mathsf{trm}$, for v a term variable;
- $f : \overrightarrow{\mathsf{trm}}^n \to \mathsf{trm}$, for f an n-ary function symbol;
- $G : \overrightarrow{\mathsf{trm}}^n \to \mathsf{prp}$, for G an atomic propositional function symbol of arity n;

- $\gamma : \overrightarrow{\mathsf{prp}}^n \to \mathsf{prp}$, for γ an n-ary propositional connective;
- $\theta : (\overrightarrow{\mathsf{trm}}^{n_1} \to \mathsf{prp}) \to \ldots \to (\overrightarrow{\mathsf{trm}}^{n_k} \to \mathsf{prp}) \to \mathsf{prp}$, for θ a quantifier taking a number k of propositional functions of arities $n_1,\ldots,n_k$; and either
- $\boldsymbol{\imath} : (\mathsf{trm} \to \mathsf{prp}) \to \mathsf{trm}$, in case of $\Sigma_0(\boldsymbol{\imath})$; or
- $\boldsymbol{\iota} : (\mathsf{trm} \to \mathsf{prp}) \to (\mathsf{trm} \to \mathsf{prp}) \to \mathsf{prp}$, in case of $\Sigma_0(\boldsymbol{\iota})$.

For instance, $\wedge : \mathsf{prp} \to \mathsf{prp} \to \mathsf{prp}$, and $\forall : (\mathsf{trm} \to \mathsf{prp}) \to \mathsf{prp}$. Clearly, these type assignments suffice to faithfully express all terms and formulas of the languages over the basic signatures $\Sigma_0(\boldsymbol{\imath})$ and $\Sigma_0(\boldsymbol{\iota})$, i.e. to render them as expressions.

Example 5.2. The paraphrase $\exists x\,(\forall y\,(Qy \leftrightarrow x = y) \wedge Px)$ is rendered as the, admittedly rather hard to read, expression $\exists(\lambda x.\wedge(\forall(\lambda y.\leftrightarrow(Q(y),=(x,y))),P(x)))$. Adopting, on top of the functional notation, usual conventions such as allowing for infix notation of binary operations and abbreviating $\theta(\lambda x.\,e)$ to $\theta\,x.e$ for a quantifier θ, the expression obtains a more familiar form $\exists\,x.(\forall\,y.Q(y) \leftrightarrow x = y) \wedge P(x)$. ❑

The main remaining notational difference between the original paraphrase and its higher-order rendering resides in the parenthesizing. This is caused by the difference between the binding scope conventions of predicate logic and the λ-calculus. In predicate logic the convention is that the binding scopes of quantifiers extend *minimally* to the right, $\forall x\,\phi \wedge \psi \equiv (\forall x\,\phi) \wedge \psi$, whereas in the λ-calculus the binding scope of abstraction extends *maximally* to the right, $\forall x.\,\phi \wedge \psi \equiv \forall x.(\phi \wedge \psi)$.

We start with discussing the deficiencies, from a term rewriting perspective, of Russell's description elimination.

5.1. **Ambiguous descriptions as term rewriting system.** The elimination step $G[y := \boldsymbol{\imath}F] \;\mapsto\; C[F, \lambda y.G]$ for the language of ambiguous descriptions $\Sigma_0(\boldsymbol{\imath})$ as given in Section 2, gives rise to the higher-order term rewriting rule

$$G(\boldsymbol{\imath}\,x.F(x)) \quad \to_{\boldsymbol{\imath}} \quad e \tag{1}$$

with e the expression corresponding to the paraphrase $C[F, \lambda y.G]$. The assumption that $C[F, \lambda y.G]$ is a paraphrase, entails that the description operator does not occur in the right-hand side e of the rule.

Example 5.3. For the (schematic version of the) paraphrase of Example 5.2 the rewrite rule becomes

$$G(\boldsymbol{\imath}\,x.F(x)) \quad \to_{\boldsymbol{\imath}} \quad \exists\,x.(\forall\,y.F(y) \leftrightarrow x = y) \wedge G(x)$$

Note that the meta-variables G, F of the original schematic elimination step are turned into object-variables, with the same names, of the corresponding rule. ❑

In higher-order term rewriting, to check whether a rule is applicable at some position in an expression, it suffices to try to *match* the left-hand side of the rule with the sub-expression at that position, i.e. to find a substitution to instantiate the left-hand side such that it becomes equal to that sub-expression. We show that the shape of the left-hand side of the above rule causes that matching suffers from several deficiencies, offering a term rewriting explanation for the lack of good properties of the corresponding description elimination process.

Vacuous containment When matching the left-hand side $G(\boldsymbol{\imath}\,x.F(x))$ with an expression, the substitutions allowed for G are *a priori* not restricted in any way. As a consequence, and this is a first deficiency, the idea that for matching to succeed the expression should *contain* $\boldsymbol{\imath}\,x.F(x)$ is not captured. To see this, reconsider

Example (1) of Section 3, i.e. the formula P, where P is a 0-ary predicate symbol. Although it may not look like so, the left-hand side of the rule does match with this formula. To wit, substituting $\lambda y.P$ for G suffices, independently of what is being substituted for F.

Remark 5.4. Consider a β-expansion making the left-hand side of the higher-order rewrite rule appear literally in matching P:

$$P \leftarrow_\beta (\lambda y.P)\,\iota\, x.\bot \leftarrow_\beta (\lambda y.P)\,\iota\, x.(\lambda x.\bot)\,x \leftarrow^2_\beta \underline{(\lambda F\,G.G(\iota\, x.F(x)))}\,(\lambda x.\bot)\,\lambda y.P$$

Observe that the first of the β-expansion steps is not *linear*; y does not occur in P. One could view non-linearity as the culprit of the above anomaly, and contemplate switching to *linear* higher-order term rewriting, taking terms modulo *linear* $\beta\eta$-reduction instead of full $\beta\eta$-reduction. However, apart from disallowing the above erasing matches for G as intended, this would also have the effect of disallowing such matches for F, e.g. it would disallow the elimination of the description operator from $P(\iota\, x.\bot)$ as that would require a non-linear match for F. Here we will not go into the discussion whether this is desirable or not, but do note that the formalisation via term rewriting brings to light that there *is* something to discuss. ❑

By the same considerations, the rule is in fact seen to be applicable to *any* formula, which may lead to anomalous situations. For example, *every* formula ϕ induces an infinite rewrite sequence, independently of the right-hand side e, i.e. the paraphrase, of rule (1) and description elimination is not terminating.

Definition 5.5. A rewrite relation $\to_R$ is *terminating*[15] if it does not allow infinite rewrite sequences, see Figure 1. ❑

$$a_0 \xrightarrow[R]{} a_1 \xrightarrow[R]{} a_2 \xrightarrow[R]{} a_3 \xrightarrow[R]{} \dots$$

FIGURE 1. Non-termination/Infinite rewrite sequence

To wit, defining $\phi_0 = \phi$ and $\phi_{i+1} = e[F{:=}\lambda x.\psi_i, G{:=}\lambda y.\phi_i]$ with y not in ϕ_i and with ψ_i chosen arbitrarily, yields the infinite rewrite sequence $\phi_0 \to_\iota \phi_1 \to_\iota \phi_2 \to_\iota \dots$ But not only does description elimination *never* terminate irrespective of the paraphrase, a statement stronger than F1 in the introduction. Even worse, along the way a true formula could be rewritten into a false one! For instance, instantiating the formula ϕ to $0 = 0$ and the rule to the one of Example 5.3 gives rise to the sequence

$$0 = 0 \to_\iota \underbrace{\exists\, x.(\forall\, y.\bot \leftrightarrow x = y) \wedge 0 = 0}_{\equiv \bot} \to_\iota \dots$$

by substituting $\lambda x.\bot$ for F and $\lambda y.0 = 0$ for G in the first step. One easily calculates that in this sequence only the first formula is true and the others are all false. Now note that in these examples neither P nor $0 = 0$ contains the description operator, at least intuitively so, but still it can be eliminated from them by the rule. There is a mismatch between the intuition of containment and its formalisation (via

[15]Termination is also known as Strong Normalisation.

β-reduction) allowing to substitute arbitrary propositional functions for G, even *erasing* ones such as $\lambda y.\, P$. To avoid the anomaly we should disallow such erasing matches, i.e. only allow substituting propositional functions of the shape $\lambda y.\, \phi$ for G, where y occurs *at least once* in ϕ.

Multiple containment Consider the formula $P(\iota\, x.\, Q(x), \iota\, x.\, Q(x))$. Apart from eliminating each of the ι-operators in it individually, higher-order rewriting also allows to eliminate both of them simultaneously by substituting $\lambda x.\, Q(x)$ for F and $\lambda y.\, P(y, y)$ for G in the left-hand side $G(\iota\, x.\, F(x))$ of rule (1) above, giving rise to the step

$$P(\iota\, x.\, Q(x), \iota\, x.\, Q(x)) \quad \to_\iota \quad \exists\, x.\, (\forall\, y.\, Q(y) \leftrightarrow x = y) \wedge P(x, x)$$

Even if eliminating several occurrences at the same time can be desirable from an efficiency perspective, such simultaneous elimination steps should be conceptually redundant, that is, they should always be sequentialisable into a number of individual elimination steps, leading to an, in some appropriate sense, equivalent result. Thus it should suffice to only allow substituting *affine* expressions for G, i.e. propositional functions of the shape $\lambda y.\, \phi$ where y occurs *at most once* in ϕ. That this constraint is not captured can be seen as a second deficiency of the rule. Combining the constraints with that of the previous paragraph amounts to requiring that y should occur *exactly once* in ϕ or, stated differently, that G is a so-called *context* variable [LV00] (note there is a bijective correspondence between occurrences of subterms and one-hole contexts).

Non-head-definedness A third deficiency of the description elimination rule (1) is that it is not *head-defined*, that is, the head-symbol G of its left-hand side $G(\iota\, x.\, F(x))$ is not a connective or quantifier, but a variable. As a consequence, *scope-widening* holds: if the left-hand side of the rule matches some sub-expression of an expression, it matches the expression itself, as one easily checks.

Remark 5.6. Non-head-definedness can be seen as the source of the ambiguity of the formula $\neg P(\iota\, x.\, Q(x))$ discussed in the introduction. *Because* the description can be eliminated from the sub-formula $P(\iota\, x.\, Q(x))$ it can be eliminated from the formula itself too, yielding the two interpretations $\neg \exists\, x.\, (\forall\, y.\, Q(y) \leftrightarrow x = y) \wedge P(x)$ and $\exists\, x.\, (\forall\, y.\, Q(y) \leftrightarrow x = y) \wedge \neg P(x)$, respectively.

Scopes can only be widened up to occurrences of quantifiers or ι's that bind free variables in the description. For instance, in $\forall\, x.\, P(\iota\, y.\, Q(y, x))$ the scope of the description cannot be widened to the whole formula, and similarly for the inner description in $P(\iota\, x.\, (\iota\, y.\, x + 1 = y) = 2)$. Indeed, matching the left-hand side of the rule fails in both cases as it would require to 'unbind' the variable x. Accordingly, scope widening is constrained by the requirement that the variables free in the expression substituted for F be bound outside the expression matching the left-hand side. ❑

We define the *wide-scope* strategy as the rewrite strategy that rewrites descriptions within their widest possible scope. Exploiting this strategy we now give an example for statement F1 in the introduction. That is, we show that an infinite sequence of description elimination steps is always possible for the ambiguous description operator, in case the right-hand side e of the rule contains (at least) two occurrences of G. Explicitly displaying these occurrences of G in the right-hand side and their arguments, the rule takes the form (cf. Remark 5.8)

$$G(\iota\, x.\, F(x)) \quad \to_\iota \quad C[G(\phi), G(\psi)]$$

To see the problem with such a rule having multiple occurrences of G in its right-hand side, consider its application to the formula $\iota x.Q(x) = \iota x.Q(x)$. Substituting $\lambda x.Q(x)$ for F and $\lambda y.\, y = \iota x.Q(x)$ for G yields the step

$$\iota x.Q(x) = \iota x.Q(x) \quad \to_\iota \quad C[\phi = \iota x.Q(x), \psi = \iota x.Q(x)]$$

Observe that the step eliminates the leftmost but duplicates the rightmost occurrence of $\iota x.Q(x)$ in $\iota x.Q(x) = \iota x.Q(x)$. Applying the wide-scope strategy to the leftmost occurrence of $\iota x.Q(x)$ in a formula of shape $D[\iota x.Q(x), \iota x.Q(x)]$ gives rise to the step

$$D[\iota x.Q(x), \iota x.Q(x)] \quad \to_\iota \quad C[D[\phi, \iota x.Q(x)], D[\psi, \iota x.Q(x)]]$$

in which the rightmost occurrence of $\iota x.Q(x)$ is duplicated. The resulting formula takes on the same shape again, hence defining $D_0 = D$ and $D_{i+1}[\Box_1, \Box_2] = C[D_i[\phi, \Box_1], D_i[\psi, \Box_2]]$, gives rise to the infinite rewrite sequence

$$D_0[\iota x.Q(x), \iota x.Q(x)] \to_\iota D_1[\iota x.Q(x), \iota x.Q(x)] \to_\iota D_2[\iota x.Q(x), \iota x.Q(x)] \to_\iota \ldots$$

Thus we have shown:

Theorem 5.7. *Suppose that the schematic paraphrase $A[P, Q]$ contains more than one occurrence of P. Then the description elimination system is not terminating: applying the wide-scope strategy to a formula containing two (closed) descriptions gives rise to an infinite reduction sequence.* (F1)

We will see that other scope strategies give rise to better elimination behaviour.

Remark 5.8. *A priori*, occurrences of G could occur *nested* in the right-hand side of the above rule, i.e. as $C[G(D[G(\phi)])]$. Below we will show that such situtations cannot occur in predicate logic, but could in more general settings. For now it suffices to note that the same substitutions as above would then give rise to a step

$$\iota x.Q(x) = \iota x.Q(x) \quad \to_\iota \quad C[D[\phi = \iota x.Q(x)] = \iota x.Q(x)]$$

the result of which contains two disjoint occurrences of $\iota x.Q(x)$. This suffices to carry out the above analysis, entailing non-termination of description elimination. ❑

Non-unitary matching A fourth deficiency of the description elimination rule (1) is that, even if G is taken to be a context variable, and a scope strategy and the position where to apply the rule are given, description elimination may still lead to distinct results. To wit, the left-hand side $G(\iota x.F(x))$ of the rule matches with the formula $P(\iota x.Q(x), \iota x.Q(x))$ in two distinct (linear) ways: substituting $\lambda x.Q(x)$ for F and *either* $\lambda y.P(y, \iota x.Q(x))$ *or* $\lambda y.P(\iota x.Q(x), y)$ for G yields

$$\begin{array}{lll} P(\iota x.Q(x), \iota x.Q(x)) & \to_\iota & \exists x.(\forall y.Q(y) \leftrightarrow x = y) \wedge P(x, \iota x.Q(x)) \quad \text{or} \\ P(\iota x.Q(x), \iota x.Q(x)) & \to_\iota & \exists x.(\forall y.Q(y) \leftrightarrow x = y) \wedge P(\iota x.Q(x), x) \end{array}$$

In such cases, when several substitutions can be used for matching, one says that matching is not *unitary*. The culprit is the occurrence of an expression $(\iota x.F(x))$ nested inside a variable (G) in the left-hand side of the rule.

We will see shortly that this entails that the rewrite relation lacks confluence, an important property of rewrite relations. Whereas termination guarantees existence of normal forms, confluence ensures their uniqueness.

Definition 5.9. A rewrite relation $\rightarrow_R$ is *confluent*,[16] if for all objects a, a_1, a_2 such that there are rewrite sequences from a to both a_1 and a_2, there exists an object a' and rewrite sequences from both a_1 and a_2 to that object a', see Figure 2. ❑

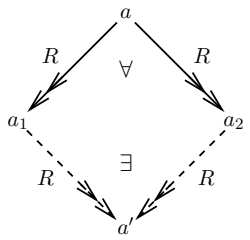

FIGURE 2. Confluence

The example in Remark 5.6 witnesses that non-head-definedness leads to non-confluence. Matching being non-unitary also leads to non-confluence as can be seen by prolonging the above steps from $P(\iota x.Q(x), \iota x.Q(x))$ to rewrite sequences to distinct normal forms:

$$\exists\, x.(\forall\, y.Q(y) \leftrightarrow x = y) \wedge \exists\, x'.(\forall\, y'.Q(y') \leftrightarrow x' = y') \wedge P(x, x')$$

$$\exists\, x.(\forall\, y.Q(y) \leftrightarrow x = y) \wedge \exists\, x'.(\forall\, y'.Q(y') \leftrightarrow x' = y') \wedge P(x', x)$$

However the present case is more benign in the sense that although the resulting formulas are not the same they are logically equivalent, something which does not hold for the example in Remark 5.6. Still, an additional device (constraint, strategy) is desirable to let every formula with description operators stand for a unique formula without them.

Remark 5.10. Trying to apply Russell and Whitehead's disambiguation device (see Section 3) directly to $P(\iota x.Q(x), \iota x.Q(x))$ would yield something like:

$$[\iota x.Q(x)][\iota x.Q(x)]P(\iota x.Q(x), \iota x.Q(x))$$

which is clearly not what is intended. Of course, the two occurrences of $\iota x.Q(x)$ could have been renamed apart first by a change of variables, but then one would give up on the standard convention of working modulo the names of bound variables, a cure which seems worse, or in any case not better, than the disease. ❑

In this section we have argued that Russell's description elimination suffers from existence (termination) issues as well as from uniqueness (confluence) issues. The deficiencies vindicate, in our opinion, our name *ambiguous* description operator. In the next section we will investigate how these problems can be overcome.

[16]Confluence is also known as the Church–Rosser property.

5.2. **Disambiguated descriptions as term rewriting system.** We first discuss how the issues of the ambiguous description elimination rule can be overcome. Guided by this, we then present disambiguated descriptions and the corresponding elimination rule and show that it enjoys all good properties. We will assume throughout that descriptions are eliminated in a one-hole context, i.e. that G in rule (1) is a context-variable.

Non-termination of description elimination was caused on the one hand by paraphrases containing multiple occurrences of the 'context' G, and on the other hand by adhering to the wide-scope convention for descriptions. We show that remedying either suffices to regain termination of the elimination process.

Theorem 5.11. *Suppose that the schematic paraphrase $A[P,Q]$ contains only one occurrence of P. Then the description elimination system is terminating, and normal forms of formulas of predicate logic do not contain description operators.* (F2)

Proof. We claim that all occurrences of F and G in the right-hand side of the ambiguous description elimination rule are disjoint, i.e. they cannot occur nested as e.g. in $C[G(D[G(\phi)])]$. Assuming for the moment the claim holds, the assumption implies the rule must have shape

$$G(\iota\, x.F(x)) \quad \to_\iota \quad C[F(\phi_1),\ldots,F(\phi_i),G(t),F(\phi_{i+1}),\ldots,F(\phi_n)]$$

where all occurrences of F and G are displayed. It suffices to assign a suitable measure to expressions and show that it decreases with every step. We will measure expressions by means of multisets of natural numbers and order these by means of the so-called multiset-extension $>_\#$ of the usual $>$ order on natural numbers.

Multisets, also known as bags, are 'sets' where elements have a multiplicity. For instance, $[2,2,0]$ is a multiset of natural numbers where 2 has multiplicity 2, and 0 has multiplicity 1. The multiset-extension $>_\#$ of $>$ allows to replace a single occurrence of some element by an *arbitrary number of smaller elements with arbitrary multiplicty*, for instance $[2,2,0] >_\# [2,1,1,0] >_\# [2,1,1]$: first an occurrence of 2 is replaced by two occurrences of the smaller element 1 and next an occurrence of 0 is replaced by no elements. In the following, we use that the multiset-extension $>_\#$ of a terminating relation $>$ is terminating [Ter03, Section A.6].

Take as *measure* $|e|$ of an expression e the multiset [Ter03] of depths of occurrences of ι in it, where the *depth* of an occurrence of ι is the number of ι's on the path from the root to that occurrence. For instance, $|P(\iota\, x.x = (\iota\, y.y = 0), \iota\, z.z = 0)| = [1,2,1]$. We show the measure before a step is greater than the measure after it, i.e. we construct a witness to

$$|D[G(\iota\, x.F(x))]| >_\# |D[C[F(\phi_1),\ldots,F(\phi_i),G(t),F(\phi_{i+1}),\ldots,F(\phi_n)]]|$$

by associating to each position p of ι after the step, a position of ι before the step:

- if p occurs in D or G after the step, associate to it the corresponding position in D or G before the step. Note that this yields a bijection between occurrences of ι at the same depth;
- if p occurs in one of the F after the step, associate to it the corresponding position in F before the step (it exists by the assumption that G is a one-hole context); note that to a single occurrence on the left in general many occurrences on the right are associated, but each at a depth one less than on the left;

- p cannot occur in C, t, or one of the ϕ_j since paraphrases were assumed to be description-free.

This shows that the measure decreases in each description elimination step, entailing termination of the elimination process. To conclude, note that a normal form of an expression for a predicate logic formula does not contain ι's: if it would, an outermost such would exist which would be eliminable from the smallest formula encompassing it (which exists by the assumption that the expression is a formula and there are no binding operators in predicate logic, cf. Remark 5.12).

It remains to prove the claim. We show it is entailed by the general requirement that in a higher-order rewrite system the free variables of the right-hand side of a rule be contained in those of the left-hand side (here $\{F, G\}$), combined with the constraints imposed by the specific types of the symbols in the signature $\Sigma_0(\iota)$. To see this, observe that G has type $\mathsf{trm} \to \mathsf{prp}$, hence if it were to occur nested then there would be an expression of type trm having a sub-expression of type prp. We show that is impossible, using the following property.[17]

For a simply typed λ-term $e : \tau$ in β-normal form, the types of the sub-λ-terms of e are all sub-types of either τ or the types of its free variables.

By the property, any sub-expression of an expression of type trm has as type either trm or a sub-type of the type of one of the symbols in the signature, treating occurrences of the latter as free variables. Inspection of the types of the symbols in the signature (note that we assume the description operator ι not to occur in the right-hand side!) then yields that an expression of type $\ldots \to \mathsf{prp}$ can only occur nested into another such expression. In particular, an expressions of type prp can never occur nested in an expression of type trm. ❑

Remark 5.12. The restriction to formulas in the statement of Theorem 5.11 is necessary, since the ambiguous description elimination rule does not allow to eliminate the description operator from terms. For example, $\iota x.x = 0$ is a normal form despite that it contains a description operator. Even stronger, if terms would be allowed to contain binding operators, then the rule would not even suffice to eliminate the description operators from all formulas, e.g. not from $100 < \sum_{x=0}^{44} \iota y.x + 1 = y$.

The result might fail if one goes beyond first-order predicate logic. In particular if the ambiguous description rule (1) could have shape

$$G(\iota x.F(x)) \quad \to_\iota \quad F(C[G(t)])$$

then termination would fail immediately, as witnessed by reducing the formula $P(\iota x.Q(x,x), \iota x.Q(x,x))$ according to the wide-scope strategy, substituting in the first step $\lambda y.P(y, \iota x.Q(x,x))$ for G and $\iota x.Q(x,x)$ for F. Note that the example is based on the possibility, absent from first-order predicate logic, to construct a context C which takes a formula and yields a term. ❑

Clearly, to overcome running into infinite sequences of description eliminations, the wide-scope strategy should be abandoned. A solution is to adhere to the *narrow-scope* strategy, i.e. to rewrite descriptions only within the narrowest of scopes.

[17] The property is known as the *sub-formula* property, the name deriving from the identification of types with formulas through the Curry–Howard isomorphism.

Theorem 5.13. *Independently of the form of the schematic paraphrase, the description elimination system is normalising. In particular, the narrow-scope strategy gives rise to a computable, normalising, and deterministic strategy.* (F4)

Proof. By the narrow-scope convention we can write expressions in an alternative way, by combining a context E consisting of a predicate symbol with all description operators of which the symbol is the head-symbol of its (narrow-)scope, into a new quantifier symbol θ_E, which we call a *combination* symbol. For instance, the expression $\Delta = \forall\, \lambda x.P(1, (\boldsymbol{\imath}\, y.x = y) + (2 + (\boldsymbol{\imath}\, z.z = 3 + (\boldsymbol{\imath}\, w.w = 4))))$ is now rendered as $\forall\, \theta_C(1, \lambda y.x = y, 2, \lambda z.\theta_D(z, 3, \lambda w.w = 4))$ with θ_C representing the combination $C = P(\Box, \boldsymbol{\imath}\Box + (\Box + \boldsymbol{\imath}\Box))$ of the predicate symbol P with its description operators (and the symbols in between, here two $+$-symbols), and θ_D representing the combination $\Box = (\Box + \boldsymbol{\imath}\Box)$ of $=$ with its description operator. The *$\boldsymbol{\imath}$-degree* of a combination symbol θ_E is the number of $\boldsymbol{\imath}$'s in E. For instance, the $\boldsymbol{\imath}$-degrees of θ_C and θ_D are 2 and 1, respectively. Because narrow-scopes are employed when making combinations, each expression can be unambiguously written using these combination symbols. The point is that the ambiguous description elimination rule can then be rendered via rules for combination symbols of shape

$$\theta_E(t_1, \ldots, t_n) \quad \to_{\boldsymbol{\imath}} \quad e$$

for all combination symbols θ_E, where each t_i is either x_i or $\lambda x.F_i(x)$, and the right-hand side constructed from the variables in the left-hand side and combination symbols. The $\boldsymbol{\imath}$-degree of the rule is the $\boldsymbol{\imath}$-degree of θ_E. For instance, the first descriptor can be eliminated from the expression Δ using Russell's paraphrase by means of the following combination rule

$$\theta_C(x_1, \lambda x.F_2(x), x_3, \lambda x.F_4(x)) \to_{\boldsymbol{\imath}} \exists\, x.(\forall\, y.F_2(y) \leftrightarrow x = y) \wedge \theta_{C'}(x_1, x, x_3, \lambda x.F_4(x))$$

with $C' = P(\Box, \Box + (\Box + \boldsymbol{\imath}\Box))$. Note that the $\boldsymbol{\imath}$-degree of $\theta_{C'}$ is one less than the $\boldsymbol{\imath}$-degree of θ_C, simply because one of the $\boldsymbol{\imath}$'s has been eliminated. This holds in general: all combination symbols in the right-hand side of a combination rule have an $\boldsymbol{\imath}$-degree one less than the combination symbol in the left-hand side. That is, applying a rule of $\boldsymbol{\imath}$-degree can only create rule-instance of lesser degree (but rule-instance already existing before the step can be replicated many times). In the terminology of rewriting, this expresses that the combination rules constitute a higher-order term rewriting system, here denoted by $\mathcal{NS}$, *with bounded creation depth*. Since higher-order term rewriting systems with bounded creation depth are terminating, a consequence of the the so-called *Finite Family Developments Theorem* [Oos97], we conclude that $\mathcal{NS}$ is terminating, and hence that that the narrow-scope strategy is indeed a normalising strategy.

The narrow-scope strategy is evidently computable and easily made into a deterministic strategy, e.g. by always selecting to eliminate the leftmost $\boldsymbol{\imath}$-symbol. ❑

Remark 5.14. Theorem 5.13 and its proof go through directly when adjoining binding operators such as summation or quantifiers. In particular, rules with right-hand sides as in Remark 5.8 or in the second part of Remark 5.12 would present no problem for termination, unlike what was the case for Theorem 5.11. But of course the first part of Remark 5.12 still applies: with extra binding operators the elimination process may terminate prematurely, i.e. description operators might remain in the resulting normal form. ❑

Choosing to always eliminate the leftmost among the description operators in a narrow-scope, gives that for every combination symbol there is exactly one rule in the higher-order term rewriting system $\mathcal{NS}$. From a rewriting perspective this causes formulas to be unambiguous: every expression can be rewritten to a unique description-free formula which will be reached in finitely many description elimination steps. Therefore $\mathcal{NS}$ represents a disambiguation of description elimination.

The combination symbols of $\mathcal{NS}$ are by their very nature complex, and a simpler way to disambiguate the ambiguous scope elimination rule is preferable. This raises the question what disambiguation *is*. We view disambiguation as a way to choose in every formula the scope of each description operator such that this choice is *preserved* along the elimination process. Indeed, the culprit of the infinite elimination process witnessing Theorem 5.7 is not choosing *wide-scopes* for description operators, but rather that this choice is not preserved, the *scope-widening* (of the second description operator), which takes place along the elimination. This raises the question how to fix scopes for description operators in such a way that they are preserved along reduction. Note that an occurrence of $G(\iota x.F(x))$, the left-hand side of the ambiguous description elimination rule in a formula, is completely fixed by a pair of positions $\langle p, q\rangle$: the position p of the scope G and the position q of the eliminated description $\iota x.F(x)$. Now such a pair of positions represents a path through the expression and for these to be preserved along reduction, the paths should not have overlap (should not have positions in common).

Example 5.15. The pairs corresponding to the two eliminations in Remark 5.6 are $\langle 1, 2\rangle$ and $\langle \epsilon, 2\rangle$ and overlap on positions 1 and 2. ❑

The disambiguated description operator $\boldsymbol{\imath}$ as presented in Section 3, can be seen as a way to reflect the choice of a non-overlapping set of pairs[18] in the syntax: the notation singles out in advance both the description operator and its scope. Therefore one could expect that the elimination process enjoys good properties. We show this, proceeding as in Section 5.1, but now employing standard higher-order term rewriting theory to establish that the rewrite relation associated to disambiguated description elimination *is* well-behaved, in particular that it is confluent and terminating.

The elimination step $\boldsymbol{\imath}(F, G) \;\mapsto\; C[F, G]$ for the language of disambiguated descriptions $\Sigma_0(\boldsymbol{\imath})$ gives rise to the rewrite rule

$$\boldsymbol{\imath}(\lambda x.F(x), \lambda y.G(y)) \quad \to_{\boldsymbol{\imath}} \quad e \tag{2}$$

where e is the expression corresponding to $C[F, G]$, i.e. a higher-order term over the signature with free variables among $\{F, G\}$ of type $\mathsf{trm} \to \mathsf{prp}$.

Example 5.16. The disambiguated version of the ambiguous rule of Example 5.3 is

$$\boldsymbol{\imath}(\lambda x.F(x), \lambda y.G(y)) \quad \to_{\boldsymbol{\imath}} \quad \exists\, x.(\forall\, y.F(y) \leftrightarrow x = y) \wedge G(x)$$

❑

[18]For this to work, positions of scopes, but only these, must be allowed to overlap. Such overlapping pairs $\langle p, q\rangle, \langle p, q'\rangle$ can be disambiguated by ordering them (in some arbitrary but fixed way). Russell and Whitehead's device serves a similar disambiguation purpose but suffers from several defects as was discussed in Section 3.

Pattern The rewrite system is a so-called higher-order pattern rewrite system. The left-hand side $\imath(\lambda x.F(x), \lambda y.G(y))$ of the rewrite rule (2) is of a special shape, it is a so-called *pattern* [MN98, Def. 3.1] This makes that matching is first-order like: it is decidable (in linear time) and unitary, meaning that substitution can be searched for effectively and if it exists, it exists uniquely.

Orthogonal The rewrite system is *orthogonal*: distinct occurrences of $\imath$ in a formula can be eliminated independently of one another, in the sense that a final result will not depend on the order of their elimination. Technically, orthogonality of the system guarantees that its associated rewrite relation is confluent [MN98, Thm. 6.11]. The fact that the elimination process yields *at most one* formula without occurrences of the description operator, independent of the order of elimination, is a direct consequence of confluence [Ter03, Thm. 1.2.2(i)].

Developments The rewrite system is a so-called higher-order *recursive program scheme* [Kha94]: the left-hand side of the rule consists of just of a single symbol, the description operator, and its right-hand side can be thought of as the definition of the symbol. Since we assume that the description operator $\imath$ does not occur in the right-hand side, definition unfolding (elimination) is even non-recursive: along any rewrite sequence starting from an expression e only copies of occurrences of $\imath$ in e are reduced, making the rewrite sequence a so-called *development* (of those occurrences). This guarantees that the associateted rewrite relation is terminating [Ter03, Thm. 11.5.11]. The fact that the elimination process eventually yields *some* formula without occurrences of the description operator, independent of the order of elimination, is a direct consequence of termination.

These three properties justify speaking of $\imath$ as a *disambiguated* description operator: each of its occurrences in a formula can be eliminated in a unique way and the elimination process is unambiguous as well:

Theorem 5.17. *For a language with scope disambiguated descriptions (i.e., descriptions considered as binary quantifiers), we have termination and unique normal forms for the description elimination system, independent of the precise choice of the (schematic) paraphrase.* (F5)

Observe that higher-order term rewriting is a suitable formalism here: not only does it allow for a concise and precise specification of the language and the elimination process, but after that, the desired properties of the elimination process, confluence and termination, follow from known higher-order term rewriting theory.

6. A Theorem in Propositional Logic

In this section we prove a result that roughly says the following. Let ϕ be a formula of propositional logic with connectives in $\{\top, \neg, \wedge\}$. Then for every other expressively complete repertoire Σ of propositional connectives there is an equivalent formula ψ with connectives in Σ such that every proposition symbol has at most as many occurrences in ψ as in ϕ.

For a precise formulation of this result we make use of the terminology of universal algebra. We start with the definition of Σ-algebra, and of how terms are interpreted in a Σ-algebras.

Definition 6.1. A (first-order) *signature* is a set of function symbols, where each $f \in \Sigma$ is associated with an $n \in \mathbb{N}$, the *arity* of f.

A Σ-*algebra* $\langle A, [\![\cdot]\!]\rangle$ consists of a non-empty domain A, and an *interpretation function* $[\![\cdot]\!]$ that maps every symbol $f \in \Sigma$ with arity n to its *interpretation*, a function $[\![f]\!] : A^n \to A$.

Let $\mathcal{A}$ be a Σ-algebra, and let $X = \{x_1, x_2, x_3, \ldots\}$ be a set of variables. The *interpretation of a term* $t \in \mathcal{T}(\Sigma, X)$ *in* $\mathcal{A}$ is defined as the function that to every valuation $v : X \to A$ assigns $[\![t, v]\!] \in A$, which is inductively defined by:

$$[\![x, v]\!] = v(x) \qquad [\![f(t_1, \ldots, t_n), v]\!] = [\![f]\!]([\![t_1, v]\!], \ldots, [\![t_n, v]\!]) \; .$$

If the interpretation of a term t in a Σ-algebra $\mathcal{C}$ is equal to the interpretation of a term t' in a Σ'-algebra $\mathcal{C}'$, then we say that t *(in* $\mathcal{C}$*) is equivalent to* t' *(in* $\mathcal{C}'$*).* ❑

We define two notions of functional completeness for Σ-algebras.

Definition 6.2. Let Σ be a signature, $X = \{x_1, x_2, x_3, \ldots\}$ a set of variables, and let $\mathcal{A} = \langle A, [\![\cdot]\!]\rangle$ be a Σ-*algebra*.

We say that a term $t \in \mathcal{T}(\Sigma, X)$ *represents in* $\mathcal{A}$ *a function* $\gamma : A^n \to A$ if for all valuations $v : X \to A$:

$$\gamma(v(x_1), \ldots, v(x_n)) = [\![t, v]\!] \; .$$

The Σ-algebra $\mathcal{A}$ is called *functionally complete* (*strictly functionally complete*) if for every function $\gamma : A^n \to A$ there exists a term $t \in \mathcal{T}(\Sigma, X)$ (a term $t \in \mathcal{T}(\Sigma, \{x_1, \ldots, x_n\})$) that represents γ. ❑

The relation between the two notions of functional completeness can be succinctly characterized as follows.

Proposition 6.3. *A* Σ*-algebra is strictly functionally complete if and only if it is functionally complete and* Σ *contains a* 0*-ary function symbol.* ❑

Proof. Suppose that $\mathcal{C} = \langle A, [\![\cdot]\!]\rangle$ is strictly functionally complete. Then clearly $\mathcal{C}$ is also functionally complete. Let $a \in A$ be arbitrary. By strict functional completeness, the 0-ary function $\gamma : A^0 \to A$, $\emptyset \mapsto a$ can be represented by a term $t \in \mathcal{T}(\Sigma, \emptyset)$ without variables. The existence of such a term implies that Σ contains at least one function symbol of arity zero.

Suppose that $\mathcal{C}$ is functionally complete, and that it contains a 0-ary function symbol, say c. For showing that $\mathcal{C}$ is strictly functionally complete, let $\gamma : A^n \to A$ be arbitrary, with some $n \in \mathbb{N}$. We have to show the existence of a term $t \in \mathcal{T}(\Sigma, \{x_1, \ldots, x_n\})$ that represents γ. Since $\mathcal{C}$ is functionally complete, there exists a formula $t' \in \mathcal{T}(\Sigma, \{x_1, x_2, \ldots\})$ that represents γ. But then for the result t of substituting c in t' for each occurrence of a variable other than $x_1, \ldots, x_n$ it holds: $t \in \mathcal{T}(\Sigma, \{x_1, \ldots, x_n\})$, and t represents γ. ❑

For the remainder of this section we restrict our attention to 'boolean Σ-algebras', a subclass of Σ-algebras. In this context, terms over Σ can be viewed as formulas of propositional logic with connectives in Σ, and will be called Σ-*formulas*.

Definition 6.4. A Σ-algebra is called a *boolean* Σ-*algebra* if its domain is $\{0, 1\}$. ❑

While for boolean functions the notion of functional completeness is more widely used in the literature, the results in this section lend themselves better to a formulation for *strictly* functionally complete boolean Σ-algebras (but see Rem. 6.9 for results concerning functionally complete boolean Σ-algebras). For an illustration of the difference between these notions, consider two standard boolean Σ-algebras.

Notation 6.5. We denote by $\Sigma_s^- := \{\neg, \wedge\}$ and by $\Sigma_s := \{\top, \neg, \wedge\}$ two standard signatures for propositional logic. By $\mathcal{C}_s^-$ we denote the boolean Σ_s^--algebra in which $\neg$ and $\wedge$ are interpreted as usual, and by $\mathcal{C}_s$ the Σ_s-algebra that extends $\mathcal{C}_s^-$ by adding the usual interpretation of the boolean constant $\top$. ❑

In contrast with $\mathcal{C}_s$, which is strictly functionally complete, $\mathcal{C}_s^-$ is only functionally complete, because the 0-ary boolean function $\gamma_\top : \{0,1\}^0 \to \{0,1\}$, $\emptyset \mapsto 1$ (the interpretation of $\top$ in $\mathcal{C}_s$) cannot be represented by a Σ_s^--formula without variables. Note, however, that $\gamma_\top$ can be represented, employing a 'dummy variable', by the Σ_s^--formula $\neg(\neg p \wedge p)$).

The next lemma shows that for every strictly functionally complete boolean algebra, negation can be formulated with only one variable.

Lemma 6.6. *Let Σ be a signature, and let $\mathcal{C}$ be a strictly functionally complete boolean Σ-algebra. Then there exists a Σ-formula $N(p)$ that in $\mathcal{C}$ is equivalent to $\neg p$ in $\mathcal{C}_s$, and that has one occurrence of p, and no occurrence of any other variable.*

Proof. Let $N(p, \ldots, p)$ be a Σ-formula that in $\mathcal{C}$ is equivalent to $\neg p$ in $\mathcal{C}_s$, and that has n occurrences of p, each of which is indicated. Since $\mathcal{C}$ is strictly functionally complete, we may assume that $N(p, \ldots, p)$ has no occurrences of other variables.

We show that if $n > 1$, then there exists a Σ-formula that is equivalent to $\neg p$, with at most to $n-1$ occurrences of p, and no occurrence of any other variable.

So, suppose $n > 1$. Let F and T be Σ-formulas that do not contain variables, and that are equivalent to $\bot$ and to $\top$, respectively; such formulas exist since $\mathcal{C}$ is strictly functionally complete. Consider the formulas

$$A(p, \ldots, p) := N(F, p, \ldots, p) \,,$$
$$B(p) := N(p, T, \ldots, T) \,.$$

On the basis that $N(p, \ldots, p)$ defines negation, the following truth tables are possible:

	$A(p,\ldots,p)$	
p	(a)	(b)
0	1	1
1	0	1

	$B(p)$	
p	(c)	(d)
0	0	1
1	0	0

In case (a) we are done, since then $A(p, \ldots, p)$ is equivalent to $\neg p$, and it has $n-1$ occurrences of p. It remains to argue further for case (b). Since $A(T, \ldots, T) = N(F, T, \ldots, T) = B(F)$, the value of B at 0 has to be the same as the value of A at 1, and it follows that case (c) is not possible for B. In the remaining case (d) for B we have again carried out the desired reduction, since then $B(p)$ is equivalent to $\neg p$, and it has just one occurrence of p. ❑

For conjunction we obtain the following similar result: for every strictly functionally complete boolean Σ-algebra, conjunction can be represented by a Σ-formula with only two variables.

Lemma 6.7. *Let Σ be a signature, and let $\mathcal{C}$ be a strictly functionally complete boolean Σ-algebra. Then there exists a Σ-formula $C(p, q)$ that in $\mathcal{C}$ is equivalent to $p \wedge q$ in $\mathcal{C}_s$, and that has one occurrence of p, one occurrence of q, and no occurrences of other propositional variables.*

Proof. Let $C(p,\dots,p,q,\dots,q)$ be a Σ-formula that in $\mathcal{C}$ is equivalent to $p \wedge q$ in $\mathcal{C}_s$, that has n occurrences of p and m occurrences of q, each of which is indicated, and that does not have occurrences of other variables. It suffices to show that if $n+m>2$, then there exists a Σ-formula that is equivalent to $p \wedge q$ that has n' occurrences of p and m' occurrences of q with $n'+m' < n+m$, and that does not have occurrences of other variables.

So suppose that $n+m>2$. Furthermore, assume that $n>1$. In case that $n=1$ and $m>1$ the argument below can be carried out analogously. Let F and T be Σ-formulas that are equivalent to $\bot$ and $\top$, respectively, and that do not contain variables. As guaranteed by Lemma 6.6, let $N(q)$ be a Σ-formula that is equivalent to $\neg q$, that has precisely one occurrence of q, and that does not have occurrences of other variables. Now consider the formulas:

$$A(p,\dots,p,q,\dots,q) := C(F,p,\dots,p,q,\dots,q)\ ,$$
$$B(p,q,\dots,q) := C(p,T,\dots,T,q,\dots,q)\ .$$

On the basis that $C(p,\dots,p)$ defines conjunction, the following truth tables are possible for A and B:

		$A(p,\dots,p,q,\dots,q)$			
p	q	(a)	(b)	(c)	(d)
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	1	1
1	1	0	1	0	1

		$B(p,q,\dots,q)$			
p	q	(e)	(f)	(g)	(h)
0	0	0	0	1	1
0	1	0	1	0	1
1	0	0	0	0	0
1	1	1	1	1	1

We proceed by case distinction on the possibilities (a)–(d) for A, and will refer to the possibilities (e)–(h) for B in order to treat cases for A that cannot be settled directly. In case (b) we are done, because then $A(p,\dots,p,q,\dots,q)$ is equivalent to $p \wedge q$, and it has $n-1$ occurrences of p and m occurrences of q. In case (c), $A(p,\dots,p,q,\dots,q)$ is equivalent to $p \wedge \neg q$. This leads us again to the desired reduction, because then $A(p,\dots,p,N(q),\dots,N(q))$ is equivalent to $p \wedge q$ and it has $n-1$ occurrences of p and m occurrences of q. It remains to argue further for the cases in which the truth table for A is (a) or (d).

We note that $A(T,\dots,T,q,\dots,q) = C(F,T,\dots,T,q,\dots,q) = B(F,q,\dots,q)$, and that therefore the values of B at $\langle 0,0\rangle$ and $\langle 0,1\rangle$ have to be the same as the values of A at $\langle 1,0\rangle$ and $\langle 1,1\rangle$, respectively. This eliminates the cases (f) and (g) for B. Hence for B the cases (e) and (h) remain. However, in case (e) it follows that the formula $B(p,q,\dots,q)$ is equivalent to $p \wedge q$, and hence we are done, because this formula has only one occurrence of p (and thus less than A) and m occurrences of q. In case (h), $B(p,q,\dots,q)$ is equivalent to $(\neg p \vee q)$. Hence the formula $N(A(p,N(q),\dots,N(q)))$ is equivalent to $p \wedge q$, and it has one occurrence of p (recall that $n>1$) and m occurrences of q. Thus we have again obtained a reduction of the desired kind. ❑

As an easy consequence of these two lemmas we obtain the result mentioned at the beginning of this section.

Theorem 6.8. *Let Σ be a signature, and let $\mathcal{C}$ be a strictly functionally complete boolean Σ-algebra.*

For every Σ_s-formula ϕ there exists a Σ-formula ψ, such that ψ in $\mathcal{C}$ is equivalent to ϕ in $\mathcal{C}_s$, and each variable has in ψ at most as many occurrences as it has in ϕ.

Proof. Let F and T be variable-free Σ-formulas that in $\mathcal{C}$ are equivalent to $\bot$ and to $\top$, respectively. As we remarked before, such formulas exist since $\mathcal{C}$ is strictly functionally complete. Let furthermore $N(p)$ and $C(p,q)$ be Σ-formulas that in $\mathcal{C}$ are equivalent to $\neg p$ and and to $p \wedge q$, respectively, with the properties stated in Lemma 6.6 and Lemma 6.7.

Now the theorem follows by induction on the structure of Σ_s-formulas. ❑

The theorem has a straightforward generalization: a similar claim holds if we extend the standard signature Σ_s with $\vee$, $\rightarrow$, and any other connectives that have the property that their interpretation can be represented by a Σ_s-formula in which no propositional variable has more than one occurrence. However, adding $\leftrightarrow$ (bi-implication) to the signature would invalidate the claim of the theorem, because for $p \leftrightarrow q$ there is no equivalent Σ_s-formula with only one occurrence of p and one of q. This in turn follows from the fact that, in each component, $\gamma_{\leftrightarrow}$ is neither monotonic nor anti-monontonic.

Remark 6.9. For functionally complete boolean Σ-algebras $\mathcal{C}$ that are not necessarily strictly functionally complete, the following two statements hold, where (i) is similar to Lemma 6.6 and Lemma 6.7, and (ii) is similar to Theorem 6.8:

(i) There exist Σ-formulas $N(p), C(p,q)$ that in $\mathcal{C}$ are equivalent to $\neg p$ and to $p \wedge q$, respectively, that contain one occurrence of each of their indicated variables, and that otherwise only contain occurrences of a single additional variable.

(ii) For every Σ_s-formula ϕ there exists a Σ-formula ψ such that ψ in $\mathcal{C}$ is equivalent to ϕ in Σ_s, and such that each variable of ϕ has in ψ at most as many occurrences as it has in ϕ, and ψ otherwise only contains occurrences of a single additional variable.

Here (i) is an easy consequence of Lemma 6.6 and Lemma 6.7, and (ii) follows from (i) by an analogous proof as that of Theorem 6.8. ❑

Finally, we apply Theorem 6.8 to obtain the result (F3) stated in the introduction: independent of the choice of expressively complete repertoire of propositional connectives for an inclusive signature of predicate logic there exist paraphrases equivalent to Russell's paraphrase that guarantee termination of $\boldsymbol{\iota}$-elimination.

Theorem 6.10. *Let Σ be an inclusive signature for predicate logic that contains $\exists$ or $\forall$, and let Σ_0 be the signature of propositional connectives contained in Σ. Let $\mathcal{C}_0$ be the strictly functionally complete Σ_0-algebra that is used for interpreting connectives when interpreting formulas over Σ.*

Then there exists a formula $A[P,Q]$ over Σ that is equivalent to Russell's paraphrase, and in which P only occurs once. If $A[P,Q]$ is chosen as paraphrase for $P(\boldsymbol{\iota}x\,Qx)$, then description elimination in formulas over $\Sigma(\boldsymbol{\iota})$ containing ambiguous description operators is terminating. (F3)

Proof. We assume that Σ contains the quantifier $\forall$. In case that Σ contains $\exists$ instead, it can be argued similarly.

Let $\psi_0(p,q) \in \mathcal{T}(\Sigma_0, \{p,q\})$ be a Σ_0-formula that in $\mathcal{C}_0$ is equivalent to $p \leftrightarrow q$; such a formula exists since $\mathcal{C}_0$ is strictly functionally complete. Let A_0 be the Σ-formula $\forall y\, \psi_0(Qy, x = y)$. Then $A_0[Q]$ is in $\mathcal{C}_0$ equivalent to $\forall y\, (Qy \leftrightarrow x = y)$.

Let $\phi(p,q)$ be the Σ_s-formula $\neg(p \wedge q)$. Then by Theorem 6.8 there exists a Σ_0-formula $\psi(p,q)$ with one occurrence of each of p and q such that $\psi(p,q)$ in $\mathcal{C}_0$ is equivalent to $\phi(p,q)$ in $\mathcal{C}_s$. Now let $A[P,Q]$ be the Σ-formula $\neg\forall x\, \psi(A_0[Q], Px)$. Then $A[P,Q]$ is in $\mathcal{C}_0$ equivalent to $\neg\forall x\, \neg(\forall y\, (Qx \leftrightarrow x = y) \wedge Px)$, and hence also to Russell's paraphrase, and P occurs in $A[P,Q]$ only once.

Termination of $\boldsymbol{\iota}$-elimination when choosing $A[P,Q]$ as paraphrase follows from Theorem 5.11. ❑

7. Bethean Afterword

Beth's work shows his strong conviction that one should philosophize in close interaction with the results of the sciences. His work both touches on Empirical Psychology and on Linguistics. Especially, he thought that philosophers should know Logic.

Since Beth's time the situation of Logic changed dramatically. Logic went into *diaspora*. We still have the classical fields *Set Theory*, *Recursion Theory*, *Model Theory*, and *Proof Theory*. Of these Model Theory comes closest to fulfilling the venerable ideal to make logic really a part of Mathematics. However, there are new applied areas of Logic, not primarily directed at either Mathematics or Philosophy, like Logic and Linguistics and Logic and Computer Science. In the case of Computer Science, the boundary between Logic and Computer Science proper often completely disappeared. Obviously, Beth would have applauded these developments.

We submit that the new situation should lead to an extension of the Bethean imperative. Philosophers should know some logic, *also from the new areas like Logic and Linguistics and Logic and Computer Science.* In this paper we tried to illustrate the importance of a branch of Logic & Computer Science, to wit Term Rewriting. Term Rewriting originated from Logic in the work of Church and Curry. It offers a more articulate and more dynamical vision of syntax than what is provided by traditional logic. Term Rewriting is an approach in the foundations of computation. It has a level of generality that covers not only proof theory and, specifically cut-elimination, but, for example, also semantical processes.

We hope to have given the reader at least an impression of how Term Rewriting provides precisely the right framework to think about a technical question that emerges naturally from a classical philosophical contribution: Russell's Theory of Descriptions.

References

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

[Kha94] Zurab Khasidashvili. On higher order recursive program schemes. In Sophie Tison, editor, *Trees in Algebra and Programming, 19th International Colloquium, CAAP'94, Edinburgh, UK, April 1113, 1994*, volume 787 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 1994.

[Kri05] S. Kripke. Russell's Notion of Scope. *Mind*, 114:1005–1037, 2005.

[KSZM09] Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean termination tool 2. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304, Brasília, 2009.

[LV00] Jordi Levy and Mateu Villaret. Linear second-order unification and context unification with tree-regular constraints. In Leo Bachmair, editor, *Rewriting Techniques and*

Applications, 11th International Conference, RTA 2000, Norwich, UK, July 10-12, 2000, Proceedings, volume 1833 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2000.

[MN98] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(1):3–29, 1998.

[Nea90] S. Neal. *Descriptions*. MIT Press, Cambridge, Massachusetts, 1990.

[Oos97] Vincent van Oostrom. Finite family developments. In *Proceedings of the 8th International Conference on Rewriting Techniques and Applications*, volume 1232 of *Lecture Notes in Computer Science*, pages 308–322, Sitges, 1997.

[Qui37] W.V. Quine. New Foundations for Mathematical Logic. *American Mathematical Monthly*, 44:70–80, 1937.

[Qui96] W.V. Quine. *Mathematical Logic*. Harvard University Press, Cambridge, Massachusetts, 1996.

[Rus05] Bertrand Russell. On Denoting. *Mind*, 14:479–493, 1905.

[RW70] Bertrand Russell and Alfred North Whitehead. *Principia Mathematica to* $*56$. Cambridge at the University Press, London, 1970.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, March 2003.

[Tou98] Hélène Touzet. Encoding the hydra battle as a rewrite system. In *Mathematical Foundations of Computer Science*, pages 267–276, 1998.

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

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

E-mail address: clemens.grabmayer | joop.leo | vincent.vanoostrom | albert.visser @phil.uu.nl