An Equational Axiomatization of Bisimulation over Regular Expressions

FLAVIO CORRADINI, *Dip. di Mat. Pura ed Appl., Università dell'Aquila, Italy.*
E-mail: flavio@univaq.it

ROCCO DE NICOLA, *Dip. di Sistemi e Informatica, Università di Firenze, Italy.*
E-mail: denicola@dsi.unifi.it

ANNA LABELLA, *Dip. di Scienze dell'Informazione, Università di Roma "La Sapienza", Italy.*
E-mail: labella@dsi.uniroma1.it

Abstract

We provide a finite equational axiomatization for bisimulation equivalence of nondeterministic interpretation of regular expressions. Our axiomatization is heavily based on the one by Salomaa, that provided an implicative axiomatization for a large subset of regular expressions, namely all those that satisfy the non-empty word property (i.e. without 1 summands at the top level) in $*$-contexts. Our restriction is similar, it essentially amounts to recursively requiring that the non-empty word property be satisfied not just at top level but at any depth. We also discuss the impact on the axiomatization of different interpretations of the 0 term, interpreted either as a null process or as a deadlock.

Keywords: Bisimulation, concurency, equational axiomatization, nondeterminism, process algebras, regular expressions.

1 Introduction

Regular expressions were studied by Kleene [13] and axiomatized by Salomaa [18] to obtain the so called Kleene algebras. The axiomatization proposed by Salomaa, that was based on the interpretation that associates a set of strings (traces) over an alphabet of basic actions to each regular expression is reported in Table 1.[1] There, the additional condition is imposed that law (*R) only holds for those regular expressions expressions that satisfy the non-empty word property, *newp*. This amounts to saying that the axiomatization applies only to terms without 1 summands in $*$-contexts. However, this restriction is not essential; axiom (*2) permits eliminating all subterms with 1 summands in $*$-contexts and transforming them into equivalent ones (the associated sets of strings are the same) which do not have such a feature. The axiomatization of Table 1 is implicative; Redko [17] proved that no finite equational axiomatization does exist for Kleene algebras under the classical interpretation.

[1] In Table 1, we are using the implicative rule $Y + X \bullet Z = Z$ implies $X^* \bullet Y = Z$ instead of Salomaa's $Y + Z \bullet X = Z$ implies $Y \bullet X^* = Z$. This does not conceptually change the axiomatization, but makes the comparisons with our previous work easier.

TABLE 1. Complete axiomatization of regular expressions

$X + Y$	$=$	$Y + X$	(C1)
$(X + Y) + Z$	$=$	$X + (Y + Z)$	(C2)
$X + X$	$=$	X	(C3)
$X + 0$	$=$	X	(C4)
$(X \bullet Y) \bullet Z$	$=$	$X \bullet (Y \bullet Z)$	(S1)
$X \bullet 1$	$=$	X	(S2)
$1 \bullet X$	$=$	X	(S3)
$X \bullet 0$	$=$	0	(S4)
$0 \bullet X$	$=$	0	(S5)
$(X + Y) \bullet Z$	$=$	$(X \bullet Z) + (Y \bullet Z)$	(RD)
$X \bullet (Y + Z)$	$=$	$(X \bullet Y) + (X \bullet Z)$	(LD)
$1 + X \bullet X^*$	$=$	X^*	(*1)
$(1 + X)^*$	$=$	X^*	(*2)
$Y + X \bullet Z = Z$	$\Longrightarrow$	$X^* \bullet Y = Z$	(*R)

After it was realized that regular expressions have much in common with the theories of *process algebras*, interest has grown in alternative interpretations of regular expressions that put a different stress on nondeterminism. Indeed, for concurrency modelling, nondeterminism plays a central rôle. The target domain of the alternative interpretation of regular expressions are then labelled trees rather than sets of strings. With tree-based models, the law of distributivity of concatenation over sum, that is crucial for Salomaa's completeness proof, does not hold anymore. Much work has been devoted to finding axiomatizations for the new interpretations as well.

In [14], Milner introduced a complete inference system for a calculus (μ-expressions) without the $*$-operator and with an explicit recursion operator. In the same paper, it is shown that, when restricting recursion to the $*$-operator, most of the axioms used by Salomaa's are sound also for bisimulation. However, there no complete axiomatic characterizations (neither equational nor implicative) of $*$-expressions is provided. Milner's work has prompted many researchers and a few partial solutions of the problem have been proposed. Interesting results have been obtained for restricted variants of regular expressions: prefix iteration [2], multi-exit iteration [1], binary star [11], perpetual loops [10], ... However the problem of finding a complete axiomatization for nondeterministic behaviour of ($*$-)regular expressions (modulo bisimulation) is still open.

In [7], we considered a nondeterministic tree-based semantics that was called *tree equivalence* of behaviours and was analogous to the one used by Bloom and Ésik [5]. We provided a complete equational axiomatization for a large subset of the whole set of regular expressions. The restriction we imposed on the general syntax requires that terms in $*$-contexts do not have derivatives of the form $1 + Q$ for some $Q \neq 0$. This restriction can be seen as a strengthening of the non-empty word property used in [18]; it is a sort of *hereditary non-empty word*

property - *hnewp*, because we demand *newp* not just for each term but also for all of its derivatives. We proved that, for the nondeterministic interpretation of the restricted set of regular expressions there indeed exists a finite equational axiomatization where idempotency of sum does not hold. Unfortunately, differently from the situation of Salomaa's axiomatization based on trace semantics, it turns out that *hnewp* is essential. Since axiom (*2) is not valid for tree-based semantics, we cannot eliminate/transform critical terms (1-summands) in $*$-contexts.

In this paper, we tackle the problem of using techniques similar to that of [7] to axiomatize the tree-based interpretation of regular expressions factorized via the notion of Milner's bisimulation equivalence. This semantics requires that the sum operator be considered idempotent. We shall again concentrate on the subset of regular expressions that enjoy the *hnewp*, and this restriction will again be essential for the the completeness result. It must, however, be said that the induced subset properly includes all the subsets of regular expressions that have been equipped with complete bisimulation-based axiomatization [1, 11, 9, 10].

We shall first consider the language obtained by dropping 0 from the set of regular expressions in [7] since different semantics have been suggested for such a term. Then, we shall consider the impact on the axiomatization of the language with the 0. We shall consider two different interpretations of the latter and discuss their impact on the axiomatization. We will show that bisimulation equivalence over the set of 0-free regular expressions which satisfy the *hnewp* can be axiomatized by a minor variant of the set axioms in Table 1. We need to add

$$X^*\bullet(Y\bullet(X+Y)^*+1) \;=\; (X+Y)^*$$

and to remove the left distributivity law (LD), (*2) and (*R) and, obviously, the three laws for 0, namely (C4), (S4) and (S5). The new $*$-rule, is a variant of a law called BKS3 in [9] and was first proposed in [20]. The same rule was used also in [7]. Actually, the only difference between the axiomatization presented in this paper and that of [7] is addition of the idempotency law $X + X = X$ and, obviously, the removal of the axioms for 0. Even the structure of the proof is similar, but the norm that we use for defining a well founded ordering on terms (that decreases when considering proper subterms) is different and this obviously induces significant differences in the actual proofs of the main results.

After dealing with the 0-free subset, we consider two of the zero's proposed in the literature. The first one is the null operator by Milner that satisfies laws (C4) and (S5) of Table 1. The second one was originally proposed in the context of process algebras by Baeten and Bergstra [3] and is more in line with the classical interpretation of regular expressions. The latter zero, in addition to (C4) and (S5), satisfies (S4) too. We consider the two alternatives separately and show that if the former zero is added to the language then bisimulation equivalence does not have a finite equational axiomatization. The counterexample by Sewell [19] can be easily reproduced. If the zero of [3] is considered instead, then the proof of finite equational axiomatizability of bisimulation equivalence over 0-free regular expressions immediately scales up.

The rest of the paper is organized as follows. The next section briefly recalls a few basic notions; namely, regular expressions, finite state automata and bisimulation equivalence. Section 3 provides a finite equational axiomatization for bisimulation equivalence over 0-free regular expressions enjoying hnewp in $*$-contexts, while Section 4 studies the impact (on the axiomatization) of the introduction of 'a zero' in the language. Section 5 contains a few concluding remarks and discusses the differences of this work with that presented in other papers by us.

2 Regular expressions and bisimulation equivalence

This section is entirely devoted to briefly recall the definitions of regular expressions, finite state automaton and bisimulation equivalence.

Regular expressions on an alphabet $A = \{a, b, c, \ldots\}$ generate regular languages under the classical interpretation [16]. They are terms of the following grammar:

$$E ::= \quad 0 \;\Big|\; 1 \;\Big|\; a \;\Big|\; E + E \;\Big|\; E{\bullet}E \;\Big|\; E^* \qquad \text{where } a \in A.$$

Regular expressions have also been a direct inspiration for many of the constructs and axiomatizations of concurrency models such as *CCS*, *CSP* and *ACP* (see [15, 12, 4] and references therein). It is possible to interpret its operator symbols in terms of (behaviours of) basic processes and operators for process composition:

- 0 denotes a deadlock process,
- 1 denotes a process successfully terminated,
- a denotes the process that executes action a and then successfully terminates,
- $E + F$ denotes the nondeterministic composition of processes E and F,
- $E{\bullet}F$ denotes the sequential composition of processes E and F, and
- E^* denotes the iteration of process E.

The rules in Table 3 associate a finite state automata with each regular expression. They make use of a 'termination' predicate over regular expressions that, given a regular expression E, says if the agent denoted by E can immediately terminate ($E\surd$). The rules defining such a predicate are given in Table 2. In particular, rule Ter says that 1 can immediately terminate (unlike 1, process a and 0 cannot immediately terminate). Rules $TSum_1$ and $TSum_2$ say that a non deterministic process $E + F$ can immediately terminate if at least one of its alternatives E and F can immediately terminate, while rule TSeq says that a sequential process $E{\bullet}F$ can immediately terminate if both E and F can immediately terminate. Finally, rule $TKleene$ says that E^* can immediately terminate.

TABLE 2. Termination predicate

$$Ter\ \frac{}{1\surd} \qquad TSum_1\ \frac{E\surd}{E+F\surd} \qquad TSum_2\ \frac{F\surd}{E+F\surd}$$

$$TSeq\ \frac{E\surd, F\surd}{E{\bullet}F\surd} \qquad TKleene\ \frac{}{E^*\surd}$$

The rules in Table 3 should be self-explanatory. Briefly, rule Act says that process a can perform an action a and then successfully terminates. The 0 term does not have transitions. It cannot perform any action.

TABLE 3. Transitional rules

$$Act\ \frac{}{a \xrightarrow{a} 1}$$

$$Sum_1\ \frac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'} \qquad Sum_2\ \frac{F \xrightarrow{a} F'}{E + F \xrightarrow{a} F'}$$

$$Seq_1\ \frac{E \xrightarrow{a} 1}{E{\bullet}F \xrightarrow{a} F} \qquad Seq_2\ \frac{E \xrightarrow{a} E',\ E' \neq 1}{E{\bullet}F \xrightarrow{a} E'{\bullet}F} \qquad Seq_3\ \frac{E\surd, F \xrightarrow{a} F'}{E{\bullet}F \xrightarrow{a} F'}$$

$$Kleene_1\ \frac{E \xrightarrow{a} 1}{E^* \xrightarrow{a} E^*} \qquad Kleene_2\ \frac{E \xrightarrow{a} E',\ E' \neq 1}{E^* \xrightarrow{a} E'{\bullet}E^*}$$

Rules Sum_1 and Sum_2 deal with nondeterministic composition. Sum_1 says that if component E can perform an action a leading to E' then also $E + F$ can perform a to become E'. Sum_2 is symmetric.

Rules Seq_1, Seq_2 and Seq_3 deal with sequential composition. Seq_1 says that if E performs an action a to become 1 then $E{\bullet}F$ performs a to become F. Similarly, Seq_2 says that if E performs an action a to become E' and $E' \neq 1$ then $E{\bullet}F$ performs a to become $E'{\bullet}F$. Seq_3 says that if E can immediately terminate and F can perform an action a to become F' then $E{\bullet}F$ can perform a to become F'.

Rules $Kleene_1$ and $Kleene_2$ deal with the Kleene star. If E can perform an action a to become E' then E^* can perform the same action to become either E^*, if $E' = 1$, or $E'{\bullet}E^*$, if $E' \neq 1$.

We are now ready to define the set of derivatives and proper derivatives of a regular expression E. These are all the terms that are reachable from E via applications of the rules in Table 3.

DEFINITION 2.1 (Derivatives and Proper Derivatives)

- E' is a *derivative* of E if $E = E_0 \xrightarrow{a_1} E_1 \ldots E_{n-1} \xrightarrow{a_{n-1}} E_n = E'$ and $n \geq 0$.
- E' is a *proper derivative* of E if E' is a derivative of E and $n > 0$.
- The set of proper derivatives of E is denoted by $der(E)$.

On top of such transitional semantics of regular expressions, bisimulation equivalence [15] can be defined.

DEFINITION 2.2

1. A binary symmetric relation $\Re$ over the set of regular expressions is a *bisimulation* relation if and only if for each $(E, F) \in \Re$:
   (a) $E\surd$ implies $F\surd$;
   (b) $E \xrightarrow{a} E'$ implies $F \xrightarrow{a} F'$ and $(E', F') \in \Re$.
2. Two regular expressions E and F are *bisimilar*, $E \sim F$, if and only if there exists a bisimulation relation $\Re$ such that $(E, F) \in \Re$.

3 Axioms for bisimulation equivalence

In this section we prove that the axioms in Table 4 form a complete axiomatization for Milner's bisimulation over a large subset of regular expressions.

TABLE 4. Axioms for bisimulation over regular expressions

$X+Y$	$=$	$Y+X$	(C1)
$(X+Y)+Z$	$=$	$X+(Y+Z)$	(C2)
$X+X$	$=$	X	(C3)
$(X\bullet Y)\bullet Z$	$=$	$X\bullet(Y\bullet Z)$	(S1)
$X\bullet 1$	$=$	X	(S2)
$1\bullet X$	$=$	X	(S3)
$(X+Y)\bullet Z$	$=$	$(X\bullet Z)+(Y\bullet Z)$	(RD)
X^*	$=$	$1+X\bullet X^*$	(*1)
$(X+Y)^*$	$=$	$X^*\bullet(Y\bullet(X+Y)^*+1)$	(*2′)

We only consider terms within $*$-contexts that do not have derivatives of the form $1+Q$ for some $Q \neq 0$. The wanted set of terms is determined by the boundedness predicate defined below.

DEFINITION 3.1 (hnewp property and Boundedness predicate)
Let $hnewp$ and *bounded* be the least predicates over regular expressions that satisfy

- - $hnewp(a)$; $hnewp(0)$;
  - $hnewp(E)$ and $hnewp(F)$ imply $hnewp(E+F)$;
  - $hnewp(F)$ implies $hnewp(E\bullet F)$
- - $bounded(a)$, $bounded(0)$ and $bounded(1)$;
  - $bounded(E)$ and $bounded(F)$ imply $bounded(E+F)$ and $bounded(E\bullet F)$
  - $bounded(E)$ and $hnewp(E)$ implies $bounded(E^*)$.

Notice that the newp of Salomaa coincides with the hnewp but for the third condition where we require that it must hold for the second term only.

Some examples are now in order. Regular expressions $(1+1)\bullet a$ and $a\bullet(b^*+c^*)\bullet d$ enjoy the hnewp and a^*, $((1+1)\bullet a)^*$, $(a\bullet(b^*+c^*)\bullet d)^*$ are bounded. On the other hand, regular expressions $a\bullet(1+1)$ and $a\bullet(b^*+c^*)$ do not enjoy the hnewp and $(a^*)^*$, $(a\bullet(b^*+c^*))^*$ are not bounded.

A simple but useful property follows.

LEMMA 3.2
Let E be a regular expression not containing 0 be such that $hnewp(E)$ holds and $E' \in der(E)$. Then, either $hnewp(E')$ holds or $E'=1$. Moreover, $1 \in der(E)$.

PROOF. By induction on the structure of E. ∎

In the rest of the section, we shall use $\mathcal{E}$ to denote the set of regular expressions which do not contain 0 and satisfy the boundedness predicate and shall use only terms in $\mathcal{E}$, often

without mentioning the restriction. Thus, all our results will be proved for 0-free regular expressions whose *-terms satisfy hnewp. Without this restriction most of the proofs would not be valid. In Section 4, then, we study the impact on the axiomatization of introducing a 0 term in the language.

The restriction to bounded regular expressions has (at least) two significant consequences:

(i) A term E^*F cannot be a derivative of F.

(ii) In a cycle $E^*F, \ldots, E'\bullet E^*F, \ldots$ derivatives of F cannot be immediate derivatives of terms $E'\bullet E^*F$ (indeed, any E' cannot be of the form $1 + G$). As a consequence of this fact, the root E^*F of a cycle is eventually reached once the cycle itself is undertaken.

Facts (i) and (ii) allow us to prove the completeness result via structural induction over regular expressions in the same vein as in [9]. Indeed, most of the (axioms and) results we need are in common with those reported by Wan Fokkink in [9]. However, since the considered languages are different, the details of the proofs are significantly different. Fokkink uses binary Kleene star (E^*F with the further restriction that both E and F do not satisfy the termination predicate). Thus, for example, the equality $a^* = a^*\bullet a^*$ can be proved within our system case but not within the axiom system of [9].

In the rest of the paper, we shall refer to all axioms in Table 4 as Ax and for any pair of regular expressions E and F in $\mathcal{E}$, we shall write $E =_{Ax} F$ if $E = F$ is derivable via the laws in Ax and the usual laws for equational reasoning.

The correctness result is standard.

THEOREM 3.3
Axioms in Table 4 are sound with respect to bisimulation equivalence.

Before stating our main result we need some further notation and new results. First of all we show that regular expressions can be reduced to a standard form.

DEFINITION 3.4 (Normal forms)
A normal form is a term of the form

$$\sum_{i\in I} a_i + \sum_{j\in J} n_j^* + \sum_{k\in K} a_k\bullet n_k + \sum_{l\in L} n_l^*\bullet n_l' + \sum_{m\in M} E_m$$

where M has at most one element and $E_m = 1$, and $n_j,\ n_k,\ n_l,\ n_l'$ are normal forms different from each other if they are indexed by the same set and different from 1 and we have $bounded(n_j^*)$ and $bounded(n_l^*)$.

LEMMA 3.5 (Reduction to normal forms)
Every regular expression E is provably equal, via the laws of Table 4 but (*1) and (*2′), to a normal form $fnf(E)$.

PROOF. The proof proceeds by induction on the depth of terms, where depth is defined by

$$\begin{aligned} \text{depth}(1) = \text{depth}(a) &= 1 \\ \text{depth}(E + F) &= \max\{\text{depth}(E), \text{depth}(F)\} \\ \text{depth}(E\bullet F) &= \text{depth}(E) + \text{depth}(F) \\ \text{depth}(E^*) &= 1 + \text{depth}(E). \end{aligned}$$

We assume that the claim holds for terms E with $\text{depth}(E) < n$. Hence, we prove it for terms E of depth equal to n by (inner) induction on the syntactic structure of terms. We refer the reader to [7] for more details. ∎

Let us now briefly comments on our normal forms. A regular expression is provably equal to an expression with at most one 1 summand. One can notice also that a regular expression does not have unique normal form and normal forms are obtained by using *-free axioms only.

Regular expressions can also be easily reduced to head normal forms; terms that will also be needed in the sequel.

LEMMA 3.6
Every regular expression E, can be transformed, via the laws of Table 4 but (*2′), into a head normal form

$$hnf(E) \;=\; \sum_{i\in I} a_i + \sum_{j\in J} a_j{\bullet}E_j + \sum_{k\in K} F_k$$

where K has at most one element and $F_k \;=\; 1$, and $E \stackrel{a_j}{\longrightarrow} E_j$ for every $j \in J$.

PROOF. The proof follows similar lines of that of Lemma 3.5. The only critical case to consider is when $E \;=\; S^*$. By axiom (*1) we have $S^* \;=\; 1 + S{\bullet}S^*$ and hence $hnf(S^*) \;=\; 1 + hnf(S{\bullet}S^*)$, then the problem reduces to finding a head normal form for S; note that $bounded(S^*)$ implies that $hnf(S)$ is of the form $(\sum_{i\in I} a_i + \sum_{j\in J} a_j{\bullet}S_j)$ and thus that R^* terms cannot appear at the top level in $hnf(S{\bullet}S^*)$. (C3) is used as a rewriting rule from left to right. ∎

REMARK 3.7
Often, to abbreviate notation and to consider fewer case analysis, it will be useful to consider $P^*{\bullet}Q$, where $Q = 1$, in place of P^* alone.

Once proven that regular expressions can be proven equal to normal forms we can, as in [9], concentrate on the set of basic terms. This is the set $\mathcal{B} = \mathcal{F} \cup \mathcal{H}$, where $\mathcal{F}$ is the set of normal forms and

$$\mathcal{H} = \{P^*{\bullet}Q, P'{\bullet}P^*{\bullet}Q \mid P^*{\bullet}Q \text{ is a normal form and } P' \neq 1 \text{ is such that } P' \in der(P)\}.$$

We say that $P^*{\bullet}Q \cong P'{\bullet}P^*{\bullet}Q$ for any proper derivative P' of P that is different by 1. By the operational semantics in Table 3 terms $P'{\bullet}P^*{\bullet}Q$ have to be intended as $(P'{\bullet}P^*){\bullet}Q$ even if we will often omit the parenthesis.

The set of normal forms $\mathcal{F}$ and the set of $\mathcal{H}$ are not closed under transitions. The following proposition shows that $\mathcal{B}$ is closed. The proof is similar to the corresponding one (Lemma 6) in [9].

LEMMA 3.8
Let $P \in \mathcal{B}$ and $P \stackrel{a}{\longrightarrow} P'$. Then $P' \in \mathcal{B}$.

In the rest of this section we will further restrict attention to the set $\mathcal{B}$ of regular expressions. To prove completeness, we need a well-founded ordering on terms. To this purpose we want a norm that enjoys specific properties and shall need a number of definitions and properties.

DEFINITION 3.9
Given a regular expression E, star(E) denotes the number of different expressions G^* appearing within E.

The following proposition shows that the number of star expressions of a process is greater than or equal to the number of star expressions of any derivative of the process itself.

PROPOSITION 3.10
Let $P \in \mathcal{B}$. Then:

(i) $P \xrightarrow{a} Q$ implies star$(Q) \leq$ star(P);
(ii) $Q \in der(P)$ implies star$(Q) \leq$ star(P).

PROOF. Item (i) follows by induction on the proof of transition $P \xrightarrow{a} Q$. Item (ii) follows immediately by item (i). ∎

PROPOSITION 3.11
The axioms in Table 4 but the last one (axiom (*2′)) preserve the number of different star expressions. Hence, star$(P) =$ star$(nf(P))$, for any $P \in \mathcal{B}$.

PROOF. A simple inspection of the axioms. Note that any G^* expression of a process $P \in \mathcal{B}$ is already a normal form by definition. Thus, stars expressions are considered atoms and unmodified while reducing P into a normal form. ∎

We are now ready to define the desired ordering over $\mathcal{B}$ terms.

DEFINITION 3.12
Let us define a relation on terms $P < Q$ if

$$P \in der(Q) \text{ and } Q \notin der(P) \qquad \text{or} \qquad \text{star}(P) < \text{star}(Q).$$

LEMMA 3.13
$<$ is a well-founded ordering on regular expressions.

PROOF. Suppose that $<$ is not well-founded, so there exists an infinite chain of elementary steps $\ldots < P_2 < P_1 < P_0$, where $P_{i+1} < P_i$ means that P_{i+1} is a proper derivative of P_i but not vice versa or star$(P_{i+1}) <$ star(P_i). Since by Proposition 3.10, star$(P_{i+1}) \leq$ star(P_i), we have that there exists N such that star$(P_i) =$ star(P_N) for any $i \geq N$. Then, for $i > N$, $P_{i+1} < P_i$ means that P_{i+1} is a proper derivative of P_i but not vice versa. This contradicts the fact that the chain is infinite because the number of different derivatives of a regular expression is finite. ∎

On top of the well-founded partial order we define a partial ordering on regular expressions.

DEFINITION 3.14
Let us define a partial ordering on regular expressions $P \leq Q$ if either $P < Q$ or $P \in der(Q)$ and $Q \in der(P)$ and

PROPOSITION 3.15
Let P, Q, R be $\mathcal{B}$ regular expressions. Then:

(i) $P < Q$ and $Q \leq R$ imply $P < R$;
(ii) $P \leq Q$ and $Q < R$ imply $P < R$.

PROOF. Let us prove item (i). Consider $P < Q$ and $Q \leq R$ and distinguish different cases depending on $P < Q$ and $Q \leq R$.

- Assume $P < Q$ because $P \in der(Q)$ and $Q \notin der(P)$ and $Q \leq R$ because $Q \in der(R)$ and $R \notin der(Q)$. From $P \in der(Q)$ and $Q \in der(R)$ we immediately have $P \in der(R)$. We prove that $R \notin der(P)$. Indeed, by contradiction, if $R \in der(P)$ then $R \in der(P)$ and $P \in der(Q)$ prove $R \in der(Q)$, by contradicting the hypothesis.

- Assume $P < Q$ because $P \in der(Q)$ and $Q \notin der(P)$ and $Q \leq R$ because star$(Q) <$ star(R). Then $P < R$ because star$(P) <$ star(R). Indeed, by Proposition 3.10(ii), star$(P) \leq$ star(Q). Moreover, star$(Q) <$ star(R).
- Assume $P < Q$ because $P \in der(Q)$ and $Q \notin der(P)$ and $Q \leq R$ because $Q \in der(R)$ and $R \in der(Q)$. From $P \in der(Q)$ and $Q \in der(R)$ we immediately have $P \in der(R)$. Assume $R \in der(P)$. This together with $Q \in der(R)$ proves $Q \in der(P)$ which contradicts the hypothesis.
- If $P < Q$ because star$(P) <$ star(Q) then the proof proceeds as the previous cases.

Item (ii) can be proved similarly. ∎

A crucial property we need states that in any regular expression $P^*{\bullet}Q$, $P' < P^*{\bullet}Q$ for any $P' \in der(P^*)$.

PROPOSITION 3.16
For any $P' \in der(P)$, $P' < P^*{\bullet}Q$.

PROOF. By Proposition 3.10, star$(P') \leq$ star(P). Moreover, star$(P) <$ star(P^*). Finally, star$(P^*) \leq$ star$(P^*{\bullet}Q)$. Hence, star$(P') <$ star$(P^*{\bullet}Q)$. ∎

Another crucial property we need, states that each derivative S' of S, cannot derive R^*S. For regular expressions E and F we say that E appears within F if either E is a strict subterm of F or $E = F$.

PROPOSITION 3.17
For any $S' \in der(S)$, $S' < R^*S$.

PROOF. Of course $S' \in der(R^*S)$. We prove that $R^*S \notin der(S')$. To this aim we prove the following more general statement:

$$\nexists S_1 \in der(S) \text{ and } \nexists S_2 \text{ such that } S \text{ appears within } S_2 \text{ and } S_1 = R^*{\bullet}S_2$$

by induction on the structure of S.

- $S = a$ and $S = 1$ are trivial.
- $S = E_1 + E_2$. By induction hypothesis the statement holds for $S = E_1$ and $S = E_2$. Assume that $\exists S_1 \in der(E_1 + E_2)$ $\exists S_2$ such that $E_1 + E_2$ appears within S_2 and $S_1 = R^*{\bullet}S_2$. Assume $S_1 \in der(E_1)$ (case $S_1 \in der(E_2)$ is completely similar). Then we have that $\exists S_1 \in der(E_1)$ and $\exists S_2$ such that $E_1 + E_2$ appears within S_2 and $S_1 = R^*{\bullet}S_2$. In particular $\exists S_1 \in der(E_1)$ and $\exists S_2$ such that E_1 appears within S_2 and $S_1 = R^*{\bullet}S_2$. This contradicts induction hypothesis.
- $S = E_1{\bullet}E_2$. Assume $\exists S_1 \in der(E_1{\bullet}E_2)$, $\exists S_2$ such that $E_1{\bullet}E_2$ appears within S_2 and $S_1 = R^*{\bullet}S_2$.
  We have three different cases to distinguish for S_1.
  (i) $S_1 = E_1'{\bullet}E_2$, where $E_1' \in der(E_1)$. By $S_1 = R^*{\bullet}S_2$ we have $E_1'{\bullet}E_2 = R^*{\bullet}S_2$. Of course it cannot be $E_1' = R^*$ and $E_2 = S_2$ since, by hypothesis, $E_1{\bullet}E_2$ appears within S_2.
  (ii) $S_1 = E_2$. Again this case is not possible since E_2 cannot coincide with R^*S_2 in spite of the fact that $E_1{\bullet}E_2$ appears within S_2.

(iii) $S_1 = E_2'$ for a proper derivative $E_2' \in der(E_2)$. Then we have $\exists S_1 \in der(E_2)$, $\exists S_2$ such that E_2 appears within S_2 and $S_1 = R^*{\bullet}S_2$. This contradicts induction hypothesis.

- $S = E^*$. Assume $\exists S_1 \in der(E^*)$, $\exists S_2$ such that E^* appears within S_2 and $S_1 = R^*{\bullet}S_2$. Each derivative of E^* is either of the form $E'{\bullet}E^*$ or E^* or 1. The latter case is of course not possible. Thus we can simply consider the former ones.

(i) $S_1 = E^*$. This case is not possible since E^* does not coincide with $R^*{\bullet}S_2$.

(ii) $S_1 = E'{\bullet}E^*$. Then we have $E' = R^*$ and $E^* = S_2$. This case is not possible because E' enjoys the hnewp while R^* does not.

∎

Another useful property is a right-cancellative property for sequential composition; namely, whenever $E{\bullet}F \sim G{\bullet}F$ then $E \sim G$. To prove this result we need a norm. It can easily seen that this norm corresponds to computing the shortest sequence of derivations of a given regular expression.

DEFINITION 3.18
Let E be a regular expression. Define $|E|$ by structural induction as follows:

$$\begin{array}{rcl} |1| & = & 0 \\ |a| & = & 1 \\ |E+F| & = & min\{|E|,|F|\} \\ |E{\bullet}F| & = & |E|+|F| \\ |E^*| & = & 0. \end{array}$$

This norm is preserved by bisimulation equivalence.

PROPOSITION 3.19
Let E and F be regular expressions. Then: $E \sim F$ implies $|E| = |F|$.

PROOF. By contradiction assume $E \sim F$ but $|E| \neq |F|$. W.l.g assume $|E| < |F|$. Then there exists a sequence of derivations out of E, $E = E_0 \stackrel{a_1}{\longrightarrow} E_1 \ldots E_{n-1} \stackrel{a_{n-1}}{\longrightarrow} E_n = 1$ where $n \geq 0$ and $n = |E|$. Since $|E| < |F|$, any sequence out of F has length greater than n. This contradicts assumption $E \sim F$. ∎

Thus our decomposition statement can be proven. This is analogous to Lemma 5 in [9] proven for regular expressions containing 1s.

LEMMA 3.20
Let E and F be regular expressions that enjoy the hnewp. Then $E{\bullet}G \sim F{\bullet}G$ implies $E \sim F$.

PROOF. Prove that relation

$$\Re = \{(E,F) \mid hnewp(E), hnewp(F) \text{ and } E{\bullet}G \sim F{\bullet}G\} \cup id,$$

where id is the identity relation, is a bisimulation relation.

Consider $(E,F) \in \Re$ and assume:

(i) $E\surd$. This case is not possible because E enjoys the hnewp. Similarly for F.

(ii) $E \stackrel{a}{\longrightarrow} E'$. Then we distinguish two possible cases.

(1) $E' = 1$. Thus also $E \bullet G \xrightarrow{a} 1 \bullet G$. By $E \bullet G \sim F \bullet G$ we have $|E \bullet G| = |F \bullet G|$. Thus it must be the case that $F \bullet G \xrightarrow{a} 1 \bullet G$ and $F \xrightarrow{a} 1$ (otherwise, $F \bullet G \xrightarrow{a} F' \bullet G$ would imply $|G| < |F' \bullet G|$ since F enjoys the hnewp). Of course $(1, 1) \in \Re$.

(2) $E' \neq 1$. Thus, by the hnewp assumption, E' enjoys the hnewp and $E \bullet G \xrightarrow{a} E' \bullet G$. By $E \bullet G \sim F \bullet G$ we also have $F \bullet G \xrightarrow{a} H$. Since E' enjoys the hnewp $|E'| > 0$. Hence H must be of the form $F' \bullet G$ for some F' derivative of F such that $|F'| > 0$. Hence, $F \bullet G \xrightarrow{a} F' \bullet G$, $F \xrightarrow{a} F'$ and $(E', F') \in \Re$.

∎

REMARK 3.21
Lemma 3.20 does not hold if E and F do not enjoy the hnewp regular expressions. As a simple counter-example note that $(1 + E) \bullet E^* \sim 1 \bullet E^*$ but $(1 + E) \not\sim 1$.

We need a lemma analogous to Lemma 9 in [9].

LEMMA 3.22
$P \in \mathcal{B}$ and $P \xrightarrow{a} P'$ implies either $P' < P$ or$P, P' \in \mathcal{H}$ and $P' \cong P$.

PROOF. The proof relies on the definition of a new norm over regular expressions. Intuitively, it gives the maximum size of a regular expression and is defined by the following inference rules:

$$\begin{aligned} [1] &= 0 \\ [a] &= 1 \\ [E + F] &= max\{[E], [F]\} \\ [E \bullet F] &= [E] + [F] \\ [E^*] &= 0. \end{aligned}$$

Then prove the following two facts A and B.

A. $P \in \mathcal{B}$ and $P' \notin \mathcal{H}$ and $P \xrightarrow{a} P'$ implies $[P'] < [P]$.
Proof: By induction on the structure of P. If $P \in \mathcal{H} - \mathcal{F}$ then it follows that $P' \in \mathcal{H}$ and we are done. Then we may assume $P \in \mathcal{F}$:

$$P = \sum_{i \in I} a_i + \sum_{j \in J} n_j^* + \sum_{k \in K} a_k \bullet n_k + \sum_{l \in L} n_l^* \bullet n_l' + \sum_{m \in M} E_m.$$

Since $P \xrightarrow{a} P'$ then P' is of one of the following terms:
- $P' = n_k$, for some n_k. In such a case we are done because $[n_k] < [a_k \bullet n_k] \leq [P]$.
- $P' = n_j^*$, $P' = n_l^* \bullet n_l'$ or $P' = n_l'' \bullet n_l^* \bullet n_l'$, for some indexes j and l, are not possible because they contradict the assumption that $P' \notin \mathcal{H}$.
- $n_l' \xrightarrow{a} P'$, for some j. In this case induction yields that $[P'] < [n_l'] = [n_l^* \bullet n_l'] \leq [P]$.
- $P' = 1$ if $a_i \xrightarrow{a} 1$, for some i. Of course $[1] = 0 < [P]$.

B. $P \in \mathcal{H}$ and $P \xrightarrow{a} P'$ implies either $P' < P$ or $P' \in \mathcal{H}$ and $P \cong P'$.
Proof: Since $P \in \mathcal{H}$ either $P = Q' \bullet Q^* \bullet R$ or $P = Q^* \bullet R$, for some $Q, Q' \in der(Q)$ and R. Hence, either $P' = Q' \bullet Q^* \bullet R$ or $P' = Q^* \bullet R$ or $P' = R'$, for a proper derivative R' of R. In the first two cases $P' \in \mathcal{H}$ and $P \cong P'$. In the latter case $R' < P$ if $P = Q^* \bullet R$ or $P = Q' \bullet Q^* \bullet R$. Indeed, $R' < Q^* \bullet R$ by Proposition 3.17 and $R' < Q' \bullet Q^* \bullet R$ because $R' < Q^* \bullet R$ and $Q^* \bullet R \leq Q' \bullet Q^* \bullet R$ (thus apply Proposition 3.15).

We are now ready to prove the main statement. Let $P \xrightarrow{a} P'$ with $P' \not< P$; we prove that $P, P' \in \mathcal{H}$ and $P \cong P'$. Since P' is a derivative of P and $P' \not< P$ then P is a derivative of P' (otherwise it would be $P' < P$ by definition of $<$).

Thus there exists a sequence of derivations $P_0 \xrightarrow{a_1} P_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} P_n = P_0$, where $n \geq 1$, $P_0 = P$ and $P_1 = P'$.

Suppose that $P_k \notin \mathcal{H}$ for all k. According to fact A, we have $[P_k] > [P_{k+1}]$. Thus $P_n = P_0$ has smaller maximum size than P_0 that is of course a contradiction. Hence $P_l \in \mathcal{H}$ for some l. Since P_k is a derivative of each $P_{k'}$ and $P_n = P_0$ by Proposition 3.10 we have star$(P_k) \leq$ star$(P_{k'})$. The fact that P_k is a derivative of each $P_{k'}$ then implies that it cannot be the case that $P_k < P_{k+1}$. Fact B together with $P_l \in \mathcal{H}$ imply $P_k \in \mathcal{H}$ for all k and $P_0 \cong P_1 \cong \ldots \cong P_n$. ∎

LEMMA 3.23
If R^* appears within T then there exists $T' \in der(T)$ such that $|T'| \geq |R'|$, where $R' \in der(R)$, $R' \neq 1$ and $|R'| = max\{|R''| \mid R'' \in der(R)\}$.

PROOF. By induction on the structure of T.

- Cases $T = 1$ or $T = a$ are not possible.
- $T = T_1 + T_2$. Then either R^* appears within T_1 or R^* appears within T_2. By induction hypothesis there exists $T' \in der(T_1)$ or $T' \in der(T_2)$ (hence $T' \in der(T_1 + T_2)$) such that the statement holds.
- $T = T_1 \bullet T_2$. We distinguish two cases:
  (i) R^* appears within T_1. By induction hypothesis there exists $T' \in der(T_1)$ such that $|T'| \geq |R'|$. Then also $T' \bullet T_2 \in der(T_1 \bullet T_2)$ and $|T' \bullet T_2| = |T'| + |T_2| \geq |R'| + |T_2| \geq |R'|$.
  (ii) R^* appears within T_2. By induction hypothesis there exists $T' \in der(T_2)$ such that $|T'| \geq |R'|$. Since $T' \in der(T_1 \bullet T_2)$, $|T'| \geq |R'|$ follows by induction hypothesis.
- $T = T_1^*$. We still have two cases to consider:
  (i) $R^* = T_1^*$. Then take $R' \bullet R^*$ where $R' \in der(R)$, $R' \neq 1$ and $|R'| = max\{|R''| \mid R'' \in der(R)\}$. Hence $|T'| = |R'| + |R^*| = |R'|$.
  (ii) R^* appears within T_1. By induction hypothesis there exists $T' \in der(T_1)$ such that $|T'| \geq |R'|$. Hence $T' \bullet T_1^* \in der(T_1^*)$ and $|T' \bullet T_1^*| = |T'| \geq |R'|$ by induction hypothesis.

∎

LEMMA 3.24
Let E and F be regular expressions in normal form such that $E \sim_S F$. Then proving $E =_{Ax} F$ reduces to proving a finite number of equations of the following kinds, where we let $E' \in der(E)$, $F' \in der(F)$ and $E' < E$ or $F' < F$

$$
\begin{array}{lrcll}
E_1 & 1 & = & 1 & \\
E_2 & a & = & a & \\
E_3 & a \bullet E' & = & a \bullet F' & \\
E_4 & R^* \bullet S & = & T^* \bullet U & \\
E_5 & R' \bullet R^* \bullet S & = & T^* \bullet U & R' \in der(R) \\
E_6 & R' \bullet R^* \bullet S & = & T' \bullet T^* \bullet U & R' \in der(R) \text{ and } T' \in der(T).
\end{array}
$$

PROOF. Consider the head normal forms of E and F respectively, namely $hnf(E)$ and $hnf(F)$. Split them according to their $\sim$-equivalent summands. Then we have that the two summands appearing in the equations can be as follows:

- like in cases E_1, E_2 or E_3
  or
- of the form $a{\bullet}E' = a{\bullet}F'$ where $E' \in der(E)$, $F' \in der(F)$ and neither $E' < E$ nor $F' < F$.

The only problematic case is the last one. By Lemma 3.22, we have that, when $a{\bullet}E' = a{\bullet}F'$ reduces to $E' = F'$, only cases E_4, E_5, and E_6 are possible because terms will necessarily derive each other. ∎

We are now ready to prove our main statement. Its proof is again very similar to both the one given in [9] for a smaller set of regular expressions and the one in [7] for the same set (but of 0 for the moment), but in the case of tree equivalence. The different ordering relation we are going to define on $\mathcal{E} \times \mathcal{E}$ plays a crucial rôle when using induction.

DEFINITION 3.25
Let star(E, F) denote the number of different subterms of the form G^* within E and F. Then, $<$ is the relation over $\mathcal{E} \times \mathcal{E}$ defined as follows: $(E, E') < (F, F')$ if

- $E < F$ and $E' \leq F'$ or $E < F$ and star$(E, E') <$ star(F, F')
  or
- $E' < F'$ and $E \leq F$ or $E' < F'$ and star$(E, E') <$ star(F, F').

It can be immediately seen that the relation defined above yields a well-founded partial order on $\mathcal{E} \times \mathcal{E}$. We are now ready to prove our main statement.

THEOREM 3.26
Let E and F be regular expressions such that $E \sim F$. Then $E =_{Ax} F$.

PROOF. The proof is by induction on $\mathcal{E} \times \mathcal{E}$. By Lemma 3.5 we can assume that E and F are in normal form. By Lemma 3.24, we have to consider at most the equations E_1, E_2, E_3, E_4, E_5 and E_6. But E_1 and E_2 are trivial and E_3 follows immediately by induction reasonings. Thus we concentrate on proving E_4, E_5 and E_6.

E_4 Consider equation $R^*{\bullet}S \sim T^*{\bullet}U$. Consider the head normal forms (see Lemma 3.6)

$$R =_{Ax} \sum_{i \in I} R_i \qquad \text{and} \qquad T =_{Ax} \sum_{j \in J} T_j$$

of R and T respectively. By definition of head normal form R_i and T_j are of the form either $a{\bullet}V$ or a where $a \in A$. Since $R^*{\bullet}S \sim T^*{\bullet}U$, each $R_i{\bullet}R^*{\bullet}S$ ($i \in I$) is bisimilar to $T_j{\bullet}T^*{\bullet}U$ ($j \in J$). We distinguish two cases:

(a) $R_i{\bullet}R^*{\bullet}S \sim T_j{\bullet}T^*{\bullet}U$ for a $j \in J$. Then $R_i \sim T_j$ by Lemma 3.20.

(b) $R_i{\bullet}R^*{\bullet}S \sim a{\bullet}U'$ for a derivative U' of U.

Thus I can be divided into the following not necessarily disjoint subsets:

$$I_0 \;=\; \{i \in I \mid \exists j \in J \text{ such that } R_i \sim T_j\}$$

and

$$I_1 \;=\; \{i \in I \mid \exists U' \in der(U), \exists a \in A \text{ such that } R_i{\bullet}R^*{\bullet}S \sim a{\bullet}U'\}.$$

Similarly, J can be divided:

$$J_0 \;=\; \{j \in J \mid \exists i \in I \text{ such that } T_j \sim R_i\}$$

and

$$J_1 \;=\; \{j \in J \mid \exists S' \in der(S), \exists a \in A \text{ such that } T_j{\bullet}T^*{\bullet}U \sim a{\bullet}S'\}.$$

If both I_1 and J_1 are not empty, then $R^*{\bullet}S \sim U''$, for a derivative U'' of U, and $T^*{\bullet}U \sim S''$, for a derivative S'' of S. Thus $U'' \sim S''$. Induction yields $R^*{\bullet}S =_{Ax} U'' =_{Ax} S'' =_{Ax} T^*{\bullet}U$.

Hence, we may assume that either I_1 or J_1 is empty, say $J_1 = \emptyset$. We try to derive equation

$$\sum_{i\in I_1} R_i{\bullet}R^*{\bullet}S \;+\; S \;=_{Ax}\; U\,. \tag{3.1}$$

In order to derive equation (3.1), we show that each summand at the l.h.s. of the equality is provably equal to a summand of U and vice versa.

By definition of I_1, for each $i \in I_1$, there exists a summand $a{\bullet}U'$ of U such that $R_i{\bullet}R^*{\bullet}S \sim a{\bullet}U'$. Since $U' < T^*{\bullet}U$, induction yields $R_i{\bullet}R^*{\bullet}S =_{Ax} a{\bullet}U'$.

Consider a summand $a{\bullet}S'$ of S. Since $R^*{\bullet}S \sim T^*{\bullet}U$ and $J_1 = \emptyset$, it follows that $a{\bullet}S'$ is equivalent to a summand $a{\bullet}U'$ of U so induction gives $a{\bullet}S' =_{Ax} a{\bullet}U'$. Finally summands equal to 1 of S corresponds with analogous summands of U.

By converse arguments it follows that each summand of U is provably equal to a summand at the l.h.s. of the equality.

Since $J_1 = \emptyset$, it follows that $J_0 \neq \emptyset$, so clearly also $I_0 \neq \emptyset$. Put $R_0 =_{Ax} \sum_{i\in I_0} R_i$ and

$$R_0 \;=\; T\,. \tag{3.2}$$

In order to prove this equation, note that by definition of I_0 and $J_0 = J$, each R_i ($i \in I_0$) is equivalent to a T_j ($j \in J$). Since $R_i \leq R < R^* \leq R^*{\bullet}S$, induction yields $R_i =_{Ax} T_j$. Conversely, each T_j ($j \in J$) is provably equal to a R_i ($i \in I_0$). Hence $R_0 =_{Ax} T$. Since $I_0 \cup I_1 \;=\; I$, we have

$$\begin{array}{lll}
R^*{\bullet}S & =_{Ax} & (R_0 + \sum_{i\in I_1} R_i)^*{\bullet}S \\
& =_{Ax} & {R_0}^*{\bullet}(\sum_{i\in I_1} R_i{\bullet}((R_0 + \sum_{i\in I_1} R_i)^*{\bullet}S) + S) \\
& & \qquad\qquad\qquad\qquad\qquad\qquad \text{by axiom (*2}') \text{ and RD)} \\
& =_{Ax} & {R_0}^*{\bullet}(\sum_{i\in I_1} R_i{\bullet}(R^*{\bullet}S) \;+\; S) \\
& =_{Ax} & T^*{\bullet}U.
\end{array}$$

E_5 Consider equation $R'{\bullet}R^*{\bullet}S \sim T^*{\bullet}U$, where $R' \in der(R)$. If $R' = 1$ then, by axiom (S3), the proof proceeds as in the previous case. Assume $R' \neq 1$. By Lemma 3.2, R' enjoys the hnewp. For this reason, it cannot be the case that 1 is a summand of the head normal form of R'. From this fact we can deduce that $|U| > 0$, thus there exists $U' \in der(U)$ such that $R'{\bullet}R^*{\bullet}S \sim U' \sim T^*{\bullet}U$.

First of all assume that R^* is a subterm of T^*. By Lemma 3.23, there exists $T' \in der(T)$ such that $|R''| \leq |T'|$ for every $R'' \in der(R)$.

We can have the following cases:

- $T'\bullet T^*\bullet U \sim S''$, hence $T^*\bullet U \sim S'$ for some $S' \in der(S)$ and we are done by inductive hypothesis.
- $T'\bullet T^*\bullet U \sim R''\bullet R^*\bullet S$. Since $|R''| \leq |T'|$ we can have that $R^*\bullet S$ is either bisimilar to $T^*\bullet U$ or to $T''\bullet T^*\bullet U$. The former case is impossible because $R'\bullet R^*\bullet S$ would be equivalent to $R^*\bullet S$ (note that $|R'\bullet R^*\bullet S| > |R^*\bullet S|$).
  Assume $R^*\bullet S \sim T''\bullet T^*\bullet U$, we have $|S| > 0$, then, for every immediate derivative S'' of S, $S'' \in der(S)$, we have either $S'' \sim T^*\bullet U$ or $S'' \sim T'''\bullet T^*\bullet U$. In both cases there is $S' \in der(S)$ such that $S' \sim T^*\bullet U$ and we are done. Hence we have that there is $U' \in der(U)$ and $S' \in der(S)$ such that $R'\bullet R^*\bullet S \sim U' \sim S' \sim T^*\bullet U$. Inductive hypothesis yields $R'\bullet R^*\bullet S =_{Ax} U'$, $S' =_{Ax} U'$ and $S' =_{Ax} T^*\bullet U$.

If R^* is not a subterm of T^* (notice that axioms reducing a term into a normal form do not change the number of $*$-subterms) the head normal form of U must be of the form:

$$U =_{Ax} \sum_{i\in I} a_i\bullet U_i\,.$$

Since $R'\bullet R^*\bullet S \sim T^*\bullet U$ each U_i is bisimilar to $R^*\bullet S$ or to a term $R''\bullet R^*\bullet S$, where $R'' \in der(R)$.

But $U_i < T^*\bullet U$ and $R'\bullet R^*\bullet S$, $R''\bullet R^*\bullet S$, $R^*\bullet S$ derive each other.

Induction yields $U_i =_{Ax} R^*\bullet S$ or $U_i =_{Ax} R''\bullet R^*\bullet S$ respectively. This holds for all i, so $U =_{Ax} \sum_{i\in I} a_i\bullet U_i =_{Ax} V\bullet(R^*\bullet S)$ for some regular expression V.

Then, $R'\bullet R^*\bullet S \sim (T^*\bullet V)\bullet(R^*\bullet S)$ and hence, by Lemma 3.20, $R' \sim T^*\bullet V$. Notice that each $*$-subterm in R' or in $T^*\bullet V$ is also in $R'\bullet R^*\bullet S$ or in $T^*\bullet U$. But R^* is neither a $*$-subterm in R' nor in V nor in T^* by hypothesis. It follows that we can use inductive hypothesis proving $R' = T^*\bullet V$ and hence $R'\bullet(R^*\bullet S) =_{Ax} (T^*\bullet V)\bullet(R^*\bullet S) =_{Ax} T^*\bullet(V\bullet(R^*\bullet S)) =_{Ax} T^*\bullet U$.

E_6 Consider equation $R'\bullet R^*\bullet S \sim T'\bullet T^*\bullet U$, where $R' \in der(R)$ and $T' \in der(T)$. We have two cases to consider.

(i) Suppose there exists $T'' \in der(T)$ such that $R^*\bullet S \sim T''\bullet T^*\bullet U$ (if there exists $R'' \in der(R)$ such that $R''\bullet R^*\bullet S \sim T^*\bullet U$ the proof is completely similar). Then, by item E_5, $R^*\bullet S =_{Ax} T''\bullet T^*\bullet U$ follows.
Hence, $R'\bullet R^*\bullet S \sim R'\bullet T''\bullet T^*\bullet U \sim T'\bullet T^*\bullet U$. By Lemma 3.20, $R'\bullet T'' \sim T'$. Again we can prove that $(R'\bullet T'', T') < (R'\bullet R^*\bullet S, T'\bullet T^*\bullet U)$. Indeed, either R^* or T^* cannot appear in $R'\bullet T''$ and T'. By induction hypothesis $R'\bullet T'' =_{Ax} T'$.
Finally, $R'\bullet R^*\bullet S =_{Ax} R'\bullet T''\bullet T^*\bullet U =_{Ax} T'\bullet T^*\bullet U$.

(ii) If $R^*\bullet S \sim T^*\bullet U$ then $R^*\bullet S =_{Ax} T^*\bullet U$ follows by case E_4. Moreover, by induction hypothesis we have $R' =_{Ax} T'$ so that $R'\bullet R^*\bullet S =_{Ax} T'\bullet T^*\bullet U$ immediately follows.

∎

4 Axioms for 0

In this section we add 0 term to the syntax of regular expressions. In the literature, researchers have considered several kinds of zeros. We will restrict to the two zeros that are interpreted either as a null process or as a deadlock. The first one is the null operator by Milner [14]; the second one was originally proposed by Baeten and Bergstra [3] and is in line with the classical interpretation of regular expressions. We consider the two alternatives separately and show

that the chosen interpretation has a critical impact on the completeness of the axiomatization. Indeed, we have that in the first case the system is no longer finitely axiomatizable.

First, we shall concentrate on Milner's zero that satisfies the laws in Table 5.

TABLE 5. Axioms for Milner's zero term

$X + 0$	$=$	X	(C4)
$0 \bullet X$	$=$	0	(S4)

An interesting result [19] shows that the addition of this term to the syntax of regular expressions renders bisimulation not finitely axiomatizable. Sewell has proved that any finite axiomatization of bisimulation equivalence cannot derive the equation:

$$a^* = (\underbrace{a \bullet \ldots \bullet a}_{\text{n times}})^*$$

where n is a prime number.

Since the two regular expressions are bounded, the counterexample applies also within our framework. We can thus conclude that bisimulation equivalence cannot have a finite axiomatization over bounded regular expressions when the 0 is interpreted as proposed by Milner.

REMARK 4.1
Lemma 3.20 does not hold anymore in the presence of Milner's zero. As a simple counterexample consider $a^* \bullet 0$ and $(a \bullet a)^* \bullet 0$ that are bisimulation equivalent while a^* and $(a \bullet a)^*$ are not.

Let us now consider the zero term studied in [3] and axiomatized also in [7]. The analogy with language theory suggests considering trees as the description of the local behaviour of a process just like a language describes only the (local) behaviour of an automaton moving from a state s to another state s'. Then if one considers introducing final states, and distinguishes between successful states, e.g. the states containing 1s, and deadlocked states, e.g. the states corresponding to 0, we have that $X \bullet 0 = 0$ is an expected property. Indeed, if a deadlock is encountered a final state will never be reached.

The new 0 satisfies both laws in Table 5 and $X + 0 = 0$. The new axioms, for what we call *deadlocking zero* are reported in Table 6.

TABLE 6. Axioms for deadlocking zero

$X + 0$	$=$	X	(C4)
$0 \bullet X$	$=$	0	(S4)
$X \bullet 0$	$=$	0	(S5)

In order to add the deadlocking zero to the language and the theory considered in the previous section, we need to make a small addition to the operational semantics of regular expressions. Namely, we need to add the following condition

$F \xrightarrow{b} F'$ for some regular expression F' and $b \in A$ to the premisses of rules Seq_1 and Seq_2 in Table 3. It will enable us to detect 0 processes and thus to avoid performing actions leading to a deadlock and meet the requirement of axiom (S5).

Within this setting, if we concentrate on the set of bounded regular expressions, we can obtain a completeness proof also when the deadlocking zero is taken into account. We can indeed reduce every bounded regular expression into a normal form as defined in Lemma 3.5. This means that all 0s can be 'statically' removed: every regular expression containing some 0s can be compiled into a regular expression without 0s unless it can be proven equal to 0. Please notice that the above mentioned 'compilation' cannot be performed when considering the null zero; this, to some extent, can only be detected 'dynamically', during evaluation.

After all deadlocking 0s have been removed, the completeness proof would proceed exactly as that for the regular expressions without 0's presented in the previous section.

5 Conclusions

We have provided a finite equational axiomatization for bisimulation equivalence over the set of regular expressions that enjoy the hereditary non-empty word property (hnewp); a strengthening of Salomaa's non-empty word property. Unfortunately, differently from the situation of Salomaa's axiomatization based on trace semantics, it turns out that in our case *hnewp* is essential because critical terms cannot be eliminated. It must, however, be said that the induced subset properly includes all the subsets of regular expressions for which a complete bisimulation-based axiomatization has been proposed. We strongly conjecture that, if the hnewp is removed, no finite equational axiomatization for the regular expressions can be defined.

In the framework of the classical trace interpretation *hnewp* would instead be inessential, just like *newp*. For any regular expression there exists a (trace) equivalent one that enjoys *hnewp*. Given this, one could then aim at obtaining a finite equational axiomatization for our sublanguage. Unfortunately, Redko's counterexamples can easily be reproduced also for the restricted language. Thus, we can deduce that the set of regular expression that admits a finite equational axiomatization under bisimulation semantics is larger than the one that can be finitely equationally axiomatized under trace semantics.

Our axiomatization would not be complete (even for the sublanguage without 0), if we required newp instead of hnewp. The equation $(a+b)^* = a^*(ba^*)^*$, although valid, cannot be proved in our axiomatic system. Indeed, if one can prove, via our axioms, that $E = F$ and E contains a subterm X^* such that $X = X_1 + X_2$, with $X_1 \neq X_2$, then F contains a subterm Y^* such that $Y = Y_1 + Y_2 + Y_3$ with $X_1 = Y_1$ and $X_2 = Y_2$. This means that our axioms are not able to remove sums under iteration. If we abandon the hnewp then this property does not hold anymore. Regular expressions $(a+b)^*$ and $a^* \bullet (b \bullet a^*)^*$ are bisimilar but $(a+b)^*$ does not appear within the latter term.

The work presented here is strictly related to [6, 7]. In [6] we have proposed and studied *resource equivalence*, a new bisimulation equivalence which counts the instances of specific actions that regular expression, interpreted as nondeterministic process, can perform and thus considers as different (the 'resources' available in) X and $X + X$. Resource equivalence is strictly finer than Milner's bisimulation. In [7] we equip resource equivalence with a finite equational axiomatization over the subset of regular expressions which enjoy the hnewp. In this paper we concentrate on Milner's bisimulation instead of resource equivalence and give a finite equational axiomatization for it. Again, the axiomatization is limited to the subset of

regular expressions which enjoy the hnewp. At a first glance, the proof looks very similar to that in [7] but the details are considerably different. The most important difference is in the norm that is used for defining the well-founded ordering for induction. The norm used in [7] is not suitable for Milner's bisimulation because this equivalence does not count the number of different alternatives a regular expression has to perform a specific action. Also the norm in [9] does not work in our setting due to the presence of 1s. Our new norm has required new proofs also for those statements that are present also in [7]. Other minor differencies between [7] and this paper are due to the different models: we had a denotational model (based on labelled trees) in [7] and have an operation model here. Due to this, the statement of Lemma 3.20 is stricter than that of Lemma 8 in [7]. The latter would not be valid in the setting of Milner's bisimulation equivalence over the set of regular expressions with 1s.

While concluding, we would like to remark that the axioms in Table 4 plus those in Table 6 are sound and complete also for the class of trees (denotations of regular expressions) considered in [6] quotiented modulo bisimulation equivalence [8]. This proves a coincidence result between our 'observational' view and the 'denotational' one.

Acknowledgements

Research supported by Murst projects 'Saladin' and 'Tosca'.

References

[1] L. Aceto and W. Fokkink. An equational axiomatization for multi-exit iteration. *Information and Computation*, **134**, 121–158, 1997.

[2] L. Aceto, W. Fokkink, R. van Glabbeek and A. Ingólfsdóttir. Axiomatizing prefix iteration with silent steps. *Information and Computation*, **127**, 26–40, 1996.

[3] J. C. M. Baeten and J. A. Bergstra. Process algebra with a zero object. In *Proceedings of Concur'90*, LNCS **458**, pp. 83–98, Springer Verlag, Berlin, 1990.

[4] J. A. Bergstra and J. W. Klop. Process theory based on bisimulation semantics. LNCS **354**, pp. 50–122, Springer Verlag, Berlin, 1989.

[5] S. L. Bloom and Z. Ésik. Iteration algebras of finite state process behaviours. Manuscript.

[6] F. Corradini, R. De Nicola and A. Labella. Models of nondeterministic regular expressions. *Journal of Computer and System Sciences* **59**, 412–449, 1999.

[7] F. Corradini, R. De Nicola and A. Labella. A finite axiomatization of nondeterministic regular expressions. *Theoretical Informatics and Applications*, **33**, 447–465, 1999.

[8] R. De Nicola and A. Labella. Tree Morphisms and Bisimulations. In *Proceedings of MFCS'98 Workshop on Concurrency*. Electronic Notes of TCS **18**, 1998.

[9] W. Fokkink. On the completeness of the equations for the Kleene star in bisimulation. In *Proceedings of AMAST'96*, LNCS **1101**, pp. 180–194, Springer Verlag, Berlin, 1996.

[10] W. Fokkink. Axiomatizations for the perpetual loop in process algebras. In *Proceedings of ICALP'97*, LNCS **1256**, pp. 571–581, Springer Verlag, Berlin, 1997.

[11] W. Fokkink and H. Zantema. Basic process algebra with iteration: completeness of its equational axioms. *Computer Journal* **37**, 259–267, 1994.

[12] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1989.

[13] S. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata Studies*, Shannon and McCarthy ed. pp. 3–41, Princeton University Press, 1956.

[14] R. Milner. A complete inference system for a class of regular behaviors. *Journal of Computer and System Sciences*, **28**, 439–466, 1984.

[15] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[16] R. N. Moll, M. A. Arbib and A. J. Kfoury. *An Introduction to Formal Language Theory*. Springer-Verlag, Berlin, 1987.

[17] V. N. Redko. On defining relations for the algebra of regular events (Russian). In *Ukrainskii Matematicheskii Zhurnal*, **16**, 120–126, 1964.

[18] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of ACM*, **13**, 158–169, 1966.

[19] P. Sewell. Nonaxiomatisability of equivalences over finite state processes. *Annals of Pure and Applied Logic*, **90**, 163–191, 1997.

[20] D. R. Troeger. Step bisimulation is pomset equivalence on a parallel language without explicit internal choice. *Mathematical Structures in Computer Science*, **3**, 25–62, 1993.

Received September 2000