FLAVIO CORRADINI
ROCCO DE NICOLA
ANNA LABELLA

# A finite axiomatization of nondeterministic regular expressions

*Informatique théorique et applications*, tome 33, nº 4-5 (1999), p. 447-465
<http://www.numdam.org/item?id=ITA_1999__33_4-5_447_0>

**Theoretical Informatics and Applications**
Theoret. Informatics Appl. **33** (1999) 447–465

# A FINITE AXIOMATIZATION OF NONDETERMINISTIC REGULAR EXPRESSIONS[*]

FLAVIO CORRADINI[1], ROCCO DE NICOLA[2] AND ANNA LABELLA[3]

**Abstract.** An alternative (tree-based) semantics for a class of regular expressions is proposed that assigns a central rôle to the + operator and thus to nondeterminism and nondeterministic choice. For the new semantics a consistent and complete axiomatization is obtained from the original axiomatization of regular expressions by Salomaa and by Kozen by dropping the idempotence law for + and the distribution law of $^\bullet$ over +.

**AMS Subject Classification.** 68Q55, 68Q68, 08B05.

## 1. Introduction

The theory of regular languages was first studied by Kleene [19] and then axiomatized by Salomaa [32] to obtain so called Kleene algebras. These are algebraic structures with $+$, $\bullet$, $^*$, 0 and 1 operators satisfying certain properties that have been fruitfully used also in many areas of computer science. A version of the axiomatization proposed by Salomaa is reported in Table 1. There, the additional condition is imposed that the law (*R) only holds for those expressions that satisfy the non-empty word property (*i.e.*, the axiomatization applies only to terms without 1 summands in *-contexts). Recently a new axiomatization has been proposed by Kozen [21] (see also [5]), that relies on the original axiomatization of Salomaa for finite (*-free) terms, and on few new laws for dealing with infinite expressions.

*Keywords and phrases:* Semantics, regular expressions, axiomatizations, bisimulation.

[*] *This work has been partially funded by CNR, the Italian Research Council.*

[1] Università dell'Aquila, Dipartimento di Matematica Pura ed Applicata, Via Vetoio, Loc. Coppito, I-67100 L'Aquila, Italy; e-mail: `flavio@univaq.it`

[2] Università di Firenze, Dipartimento di Sistemi e Informatica, Via C. Lombroso 6/17, 50134 Firenze, Italy; e-mail: `denicola@dsi.unifi.it`

[3] Università di Roma "La Sapienza", Dipartimento di Scienze dell'Informazione, Via Salaria 113, 00198 Roma, Italy; e-mail: `labella@dsi.uniroma1.it`

TABLE 1. Complete axiomatization of regular expressions.

$$
\begin{array}{rcll}
X+Y & = & Y+X & \text{(C1)} \\
(X+Y)+Z & = & X+(Y+Z) & \text{(C2)} \\
X+0 & = & X & \text{(C3)} \\
X+X & = & X & \text{(C4)} \\
\\
(X{\bullet}Y){\bullet}Z & = & X{\bullet}(Y{\bullet}Z) & \text{(S1)} \\
X{\bullet}1 & = & X & \text{(S2)} \\
1{\bullet}X & = & X & \text{(S3)} \\
X{\bullet}0 & = & 0 & \text{(S4)} \\
0{\bullet}X & = & 0 & \text{(S5)} \\
\\
(X+Y){\bullet}Z & = & (X{\bullet}Z)+(Y{\bullet}Z) & \text{(RD)} \\
X{\bullet}(Y+Z) & = & (X{\bullet}Y)+(X{\bullet}Z) & \text{(LD)} \\
\\
1+X{\bullet}X^* & = & X^* & \text{(*1)} \\
(1+X)^* & = & X^* & \text{(*2)} \\
Y+Z{\bullet}X=Z & \Longrightarrow & Y{\bullet}X^*=Z & \text{(*R)}
\end{array}
$$

Regular expressions and Kleene algebras have also been a direct inspiration for many of the constructs and axiomatizations of concurrency models such as CCS, CSP and ACP, generally referred to as *process algebras*. The main differences between the axiomatization of finite regular expressions and those for process algebras are essentially due to the different stress that they put on nondeterminism. Indeed, the possible structure induced by the $+$ operator is ignored by the traditional interpretation (as sets of strings) of regular expressions, while in the framework of process algebras nondeterminism has been assigned a central role.

After studying process algebras, and specifically CCS and its observational equivalence based on the notion of bisimulation [24], Milner considered an alternative semantics based on bisimulation for a calculus of expressions without the $^*$-operator and with an explicit recursion operator; these expressions have been called $\mu$-expressions. In [25], a complete inference system is defined for $\mu$-expressions and it is shown that, even when restricting recursion to the $^*$-operator, most of the axioms used by Salomaa's could be proved sound also for bisimulation. In [6] Milner's axiomatization for $\mu$-expressions was refined, by showing that finite state process behaviours have a finite implicational axiomatization, both for bisimulation and for a so called *tree equivalence* that considers as equivalent those $\mu$-expressions that give rise to isomorphic unfoldings. This improves on Milner's result, because the unique fixed point rule is not even implicational (this is due to the guardedness condition corresponding to the non-empty word property).

However, neither [25] nor [6] provide exact axiomatic characterizations of $^*$-expressions and the problem of finding a complete axiomatization for the Kleene star modulo bisimulation is left open. Indeed, after the seminal work by Milner, many researchers have unsuccessfully attempted to solve the problem. Interesting results have been obtained for restricted variants: prefix iteration [2], multi-exit iteration [1], binary star [17], perpetual loops [16], ... , but the problem of axiomatizing nondeterministic behaviour of ($^*$-)regular expressions is still open.

Here, we consider a nondeterministic tree-based semantics of ($^*$-)regular expressions. Our trees are very similar to those of [7]. By relying on the tree semantics of $^*$-expressions, we provide a complete axiomatization for a large set of regular expressions. The restriction we impose on the general syntax of regular expression with the classical $^*$ operator requires that terms in $^*$-contexts do not have derivatives of the form $1 + Q$ for some $Q \neq 0$. This restriction, defined inductively over the syntax of the terms, can be seen as a strengthening of the non-empty word property used in [32]. The additional constraint is needed for the different stress that our semantics put on choices and thus on branching. Indeed, inner 1 summands would compromise the control of the branches' growth.

For our restricted set of regular expressions we shall present a consistent and complete axiomatization that is in full agreement with a natural tree based interpretation. We shall rely on the axiomatization that first appeared in [10] (see also [11]) for the full language and on that proposed in [15] for the restricted language without 0 and 1 and with the binary $^*$-operator (instead of the unary $^*$).

In [10], the same tree based semantics of regular expression with unary $^*$-operator is considered and an axiomatization is obtained from that of Salomaa in Table 1 by *removing* the laws stating idempotence of $+$ (C4), the distribution of $\bullet$ over $+$ (LD), and rules $^*2$ and $^*R$ and *adding* the infinitary rule:

$$\forall n \in N, \quad X_n \bullet Z \leq Y \quad \text{implies} \quad X^* \bullet Z \leq Y \qquad (\omega\text{-R})\,.$$

This rule relies on the correspondence between approximants of terms and those terms that are built by unfolding via the rewriting rule

$$P^* \longrightarrow 1 + P \bullet P^*.$$

Here we show that a large class of regular expressions can be axiomatized by the finitary axioms of [10] including the law $^*1$ of Table 1 plus the variant of a law called BKS3 in [15] (firstly proposed in [35]) reported below:

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

The use of tree isomorphism as basic equivalence instead of the coarser bisimulation, is motivated by two facts: it naturally arises from the algebraic-categorical context and permits discriminating processes also according to the number of different computations that they can perform for reaching specific states. For this reason, we have called it *resource equivalence*. As shown in [12], well known bisimulation-based equivalences can be obtained as quotients of this finer one.

In [15], for the restricted set of terms (without 0 and 1 and with a binary star) a finitary axiomatization is provided that relies on the classical axioms for strong bisimulation and has three additional laws for *-terms. This work has been very important for obtaining our results. Although our language is strictly more expressive, we have imported many proof techniques to establish our completeness result. This can be explained if one retraces the main options one can take for the actual proof.

As a matter of fact, one would hope to prove the completeness result by structural induction, but this is impossible, because, due to the fixed point equation, a term can be equal to one of its proper subterms. Alternatively one could try with induction on proper derivatives, but, once again, one could have proper derivatives that are bigger than the original term. To avoid this situations, the basic idea of [15] is that of defining a norm on terms that decreases when considering proper subterms of a given term and then using it to define a well founded ordering. As a matter of fact a good norm must satisfy the conditions explicitated in Proposition 5. This seems to be impossible in general and one tries to restrict the class of terms under consideration. Fokkink [15] solved the problem by not considering 1 and by using binary Kleene star, which allowed him to ignore the terms representing "successful termination". In presence of 1's, Fokkink's norm cannot be used. Indeed, we have kept 1's and unary star but have imposed the "more generous" condition about the non-empty word property that enable us to use a new, more effective, norm.

The distinguishing feature of our approach, when compared to that of Fokkink, is that our norm is based on the notion of branching and this is not easily adaptable to calculi with the absorption law (C4) that is implied by the bisimulation based semantics of [15]. After the crucial new definition of the norm, our completeness follows the same lines of [15].

We would like to thank L. Aceto, R. Backhouse, W. Fokkink, and A. Ingólfsdóttir for many discussions on previous attempts at the problem discussed in the paper and an anonymous referee for his/her detailed comments and suggestions. Special thanks are due to Z. Ésik for his encouragement at continuing when we were discouraged by our failures.

## 2. A Tree-based interpretation of regular expressions

In this section we present a denotational semantics of regular expressions by interpreting them over a category of labelled trees. The reader is referred to [10] for additional details about the definitions and statements presented here.

Regular expressions on an alphabet $A = \{a, b, c, \dots\}$, are defined in the classical way via the BNF below.

$$E ::= 0 \mid 1 \mid a \mid E+E \mid E\bullet E \mid E^* \qquad \text{where } a \text{ is in } A.$$

For the full understanding of the rest of this section a basic knowledge of a few notions of category theory is required that can be found in any introductory book; see *e.g.* [29] and references therein.

Our category of trees, (see [12, 20] and [13]) will be named $\mathbf{T}$. A single tree will be modelled by listing all of its runs (or paths) and then saying where they agree. Thus, the tree that describes a choice between the two sequences of actions $a.b$ and $a.c$ (usually denoted by the term $a^\bullet b + a^\bullet c$ [26]) will be modelled *via* two runs, $x$ and $y$, labelled by $ab$ and $ac$ respectively, and stating that $x$ and $y$ do not agree at all. In contrast, the tree denoted by $a^\bullet(b + c)$ and representing the possibility of executing an $a$ and then performing the choice between $b$ and $c$ will be modelled via two runs $x$ and $y$ labelled by $ab$ and $ac$, but, in this case, it will be stated that $x$ and $y$ agree on the initial $a$. Runs are used to describe computations from one state to another, exactly like strings of actions within automata theory. Additional structure is introduced by agreements.

We start by introducing a structure to deal with the labels. Below $A^\star$ denotes the set of finite strings over the set $A$.

**Definition 1.** Let $\mathcal{A} = (A^\star, \leq, \wedge, \epsilon)$ be the meet semilattice where

i. $A^\star$ is the set of words on $A$,
ii. $\leq$ is the prefix order of words,
iii. $\wedge$ is the largest common prefix operation on words,
iv. $\epsilon$ is the empty word.

**Definition 2.** An $\mathcal{A}$-*tree*, that often will be called simply *tree*, $t = (X, \alpha, \beta)$ consists of:

i. a set $X$ of *runs*;
ii. a map $\alpha : X \to A^\star$,
the *extent map*, giving the computation $\alpha(x)$ performed on a run $x$;
iii. a map $\beta : X \times X \to A^\star$,
the *agreement map*, saying to what extent two computations *agree*.
For the *agreement map* it is required, that for any $x, y, z$ in $X$,
  a. $\beta(x, x) = \alpha(x)$
  (a run agrees with itself along all its length)
  b. $\beta(x, y) \leq \alpha(x) \wedge \alpha(y)$
  (the agreement between runs is not more than their largest common prefix)
  c. $\beta(x, y) \wedge \beta(y, z) \leq \beta(x, z)$
  (the agreement between $x$, $y$ and $z$ is not more than that between $x$ and z)
  d. $\beta(x, y) = \beta(y, x)$
  (it does not matter in what order agreement is specified).

We will write $t$, $t_1$ and $t_2$ for denoting typical trees, with components $t = (X, \alpha, \beta)$, $t_1 = (X_1, \alpha_1, \beta_1)$ and $t_2 = (X_2, \alpha_2, \beta_2)$. We now present an appropriate notion for comparing trees. A tree morphism from a tree $t_1$ to a tree $t_2$ is a map from the set of runs of $t_1$ to the set of runs of $t_2$ preserving extent while allowing the agreement to increase.

**Definition 3.** A *tree morphism* $f : t_1 \to t_2$ is a map $f : X_1 \to X_2$ satisfying

i. $\alpha_2(f(x)) = \alpha_1(x)$ ($f$ does not change extent)
ii. $\beta_2(f(x), f(y)) \geq \beta_1(x, y)$ ($f$ does not decrease agreement).

We are now set to define a category of $\mathcal{A}$-trees and shall denote by $\mathbf{T}$ the category whose

i. *objects* are trees ($t = (X, \alpha, \beta)$);
ii. *arrows* are tree morphisms;
iii. *identities*, ($id_t = id_X$) are defined in terms of identities over set of runs;
iv. *composition*, ($g \circ f$), is given by function composition.

Some properties of our category immediately follow.

**Proposition 1.** $\mathbf{T}$ *has initial object, given by the empty tree* $\mathbf{0} = (\emptyset,\ \emptyset,\ \emptyset)$*, and has coproducts* $\oplus$*, given by disjoint unions.*

In the next definition we introduce a concatenation operator between trees and then we prove that it is a tensor product, *i.e.* an associative binary functor with unity.

**Definition 4.** Given two trees, $t_1 = (X_1, \alpha_1, \beta_1)$ and $t_2 = (X_2, \alpha_2, \beta_2)$ sequential composition $\otimes$ is defined as follows (here . is used to denote string concatenation): $t_1 \otimes t_2 =< X, \alpha, \beta >$ , where

- $X = X_1 \times X_2$
  (a run in t is a run of $t_1$ followed by a run of $t_2$)
- $\alpha(< x_1, x_2 >) = \alpha_1(x_1).\alpha_2(x_2)$
  (the labels of runs in t are obtained by concatenating those of the arguments);
- $\beta(< x_1, x_2 >, < y_1, y_2 >) = \begin{cases} \beta_1(x_1, y_1).\beta_2(x_2, y_2) & \text{if } x_1 = y_1 \\ \beta_1(x_1, y_1) & \text{otherwise} \end{cases}$
  (the agreement between the second components of two composite runs is considered only if the runs have a common initial part).

**Proposition 2.** *Sequential composition* $\otimes$ *is a tensor product with object unit tree* $\mathbf{1} = (\{x\},\ \alpha(x) = \epsilon,\ \beta(x, x) = \epsilon)$ *and* $\mathbf{T}$ *is monoidal w.r.t.* $\otimes$.

An "iteration" operator over trees is now given.

**Definition 5.** Given a tree $t = (X, \alpha, \beta)$ over the alphabet $A$, we can define $t^\infty = (X^\infty, \alpha^\infty, \beta^\infty)$:

1. $X^\infty = \{< x_1, x_2, ..., x_n > \mid n \in N \text{ and } x_i \in X\}$
2. $\alpha^\infty(< x_1, x_2, ..., x_n >) = \alpha(x_1)\alpha(x_2)...\alpha(x_n)$
3. $\beta^\infty(< x_1, x_2, ..., x_n >, < y_1, y_2, ..., y_m >) =$

$$= \begin{cases} \alpha(x_1)\alpha(x_2)...\alpha(x_k)\beta(x_{k+1}, y_{k+1}) & \text{if } x_i = y_i,\ \forall i,\ 0 \leq i \leq k \\ & \text{and } x_{i+1} \neq y_{i+1} \\ \alpha(x_1)\alpha(x_2)...\alpha(x_k) & \text{if } x_i = y_i\ \forall i,\ 0 \leq i \leq k \\ & \text{and } n = k \text{ or } m = k. \end{cases}$$

If in the definition above we stop the construction at a given level $i$, *i.e.* $n \leq i$ we obtain the definition of the $i$-th approximant of $t^\infty$. We have the following statement (see [10] for details).

**Proposition 3.** $t^\infty = (X^\infty, \alpha^\infty, \beta^\infty)$ *is the colimit of the chain made out of its approximants.*

Regular expressions can be interpreted as trees in the category $\mathbf{T}$ by means of a function $\mathcal{T}$ defined by induction on the structure of terms.

**Definition 6.** (*Denotational Semantics*)
An algebraic interpretation of regular expressions is obtained by associating to them a tree in $\mathbf{T}$ *via* function $\mathcal{T}$:

- $\mathcal{T}[\![0]\!] = \mathbf{0}$,
- $\mathcal{T}[\![1]\!] = \mathbf{1}$,
- $\mathcal{T}[\![a]\!] = (x, \alpha(x) = a, \beta(x,x) = a)$,
- $\mathcal{T}[\![E + F]\!] = \mathcal{T}[\![E]\!] \oplus \mathcal{T}[\![F]\!]$,
- $\mathcal{T}[\![E \bullet F]\!] = \mathcal{T}[\![E]\!] \otimes \mathcal{T}[\![F]\!]$,
- $\mathcal{T}[\![E^*]\!] = \mathcal{T}[\![E]\!]^\infty$.

The above tree-based interpretation of regular expressions gives rise to a natural equivalence relation over them. Two regular expressions $E$ and $F$ are resource equivalent if and only if they give rise, according to function $\mathcal{T}[\![\_]\!]$, to isomorphic trees.

**Definition 7.** We say that two regular expressions $E$ and $F$ are resource equivalent, $E \sim F$, if and only if $\mathcal{T}[\![E]\!]$ and $\mathcal{T}[\![F]\!]$ are isomorphic.

## 3. The completeness proof

In this section we prove that the axioms in Table 2 form a complete axiomatization for

$$(\mathcal{T}[\![\mathcal{E}]\!],\ \oplus,\ \otimes, \mathbf{0}, \mathbf{1}, (-)^\infty).$$

We improve, hence, previous results that rely on an infinitary inference rule: the $\omega$-induction rule. To avoid considering terms leading to infinitely branching trees, we should restrict attention to terms without iterations of 1 within $^*$-contexts. Indeed, we shall further restrict this condition and consider only terms within $^*$-contexts that do not have derivatives of the form $1 + Q$ for some $Q \neq 0$. This will enable us to use structural induction in our completeness proof. The wanted set of terms is determined by the boundedness predicate defined below. The restriction will allow us to assume that terms of the form $G^*H$ cannot be derived from derivatives of their subterm $H$. For a more explicit syntactical and semantical characterization of the predicate the reader is referred to Lemma 2 and Proposition 4, respectively.

**Definition 8.** (*Well-formed property and Boundedness Predicate*)
Let $wf$ and *bounded* be the least predicates over regular expressions that satisfy

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

Some examples of (non-) well-formed and bounded terms are now in order. Regular expressions $(1+1){\bullet}a$ and $a{\bullet}(b^*+c^*){\bullet}d$ are well-formed 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 $a{\bullet}(b^*+c^*)$ are not well-formed and $(a^*)^*$, $(a{\bullet}(b^*+c^*))^*$ are not bounded.

Let $\mathcal{E}$ denote the set of regular expressions which satisfy the boundedness predicate. In the rest of the paper we will concentrate on terms in $\mathcal{E}$ which will be still called regular expressions.

TABLE 2. Axioms for nondeterministic regular expressions.

| | | | |
|---|---|---|---|
| $X+Y$ | $=$ | $Y+X$ | (C1) |
| $(X+Y)+Z$ | $=$ | $X+(Y+Z)$ | (C2) |
| $X+0$ | $=$ | $X$ | (C3) |
| $(X{\bullet}Y){\bullet}Z$ | $=$ | $X{\bullet}(Y{\bullet}Z)$ | (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) |
| $X^*$ | $=$ | $1+X{\bullet}X^*$ | (*1) |
| $X^*{\bullet}(Y{\bullet}(X+Y)^*+1)$ | $=$ | $(X+Y)^*$ | (*2′) |

Moreover, all terms shall be considered modulo associativity, commutativity and absorption of 1 and 0, *i.e.* up to axioms (C1), (C2), (C3) and (S2), (S3) of Table 2. We shall refer to all axioms there in 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 from the laws in $Ax$ and the usual laws for equational reasoning. Notice that in Table 2 axiom (*1) could be replaced by the simpler axiom $0^* = 1$.

The correctness result is standard. Apart from (*2′), all the axioms in Table 2 have been proven sound in [10]. Soundness of (*2′) can be established similarly.

**Theorem 1.** *Axioms in Table 2 are valid in* $(\mathcal{T}[\![\mathcal{E}]\!], \oplus, \otimes, \mathbf{0}, \mathbf{1}, (-)^\infty)$.

*Proof.* We prove correctness of (*1) and (*2′) only. We observe that both $\mathcal{T}[\![X^*]\!]$ and $\mathcal{T}[\![1 + X{\bullet}X^*]\!]$ are colimit of the same chain of approximants, therefore they are isomorphic. More elementarily, they are both obtained as all possible finite concatenations of runs in $\mathcal{T}[\![X]\!]$, each of them taken once, labelled *via* concatenation in $A^\star$ and glued as far as possible. Analogously, $\mathcal{T}[\![X^*{\bullet}(Y{\bullet}(X+Y)^* + 1)]\!]$ and $\mathcal{T}[\![(X+Y)^*]\!]$ are both obtained as all possible finite concatenations of finite blocks of runs in $\mathcal{T}[\![X]\!]$ and $\mathcal{T}[\![Y]\!]$, every concatenation taken once, labelled using concatenation in $A^\star$ and glued as far as possible. □

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

**Definition 9.** (*Normal forms*)
A normal form is either 0 or 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 $E_m = 1$ for all $m$ and $n_j$, $n_k$, $n_l$, $n_l'$ are normal forms different from 0 and 1 and we have *bounded*$(n_j^*)$ and *bounded*$(n_l^*)$.

**Lemma 1.** (*Reduction to normal forms*)
*Every regular expression $E$ is provably equal, via the laws of Table 2, to a normal form $fnf(E)$.*

*Proof.* The proof proceeds by induction on the depth of terms, defined by

$$\begin{array}{rcl}
\text{depth}(0) & = & 0\\
\text{depth}(1) = \text{depth}(a) & = & 1\\
\text{depth}(E+F) & = & \max\{\text{depth}(E), \text{depth}(F)\}\\
\text{depth}(E{\bullet}F) & = & \text{depth}(E) + \text{depth}(F)\\
\text{depth}(E^*) & = & 1 + \text{depth}(E).
\end{array}$$

We assume that the claim holds for terms $E$ with depth$(E) < n$. Hence, we prove it for terms $E$ of depth equal to $n$ by (inner) induction on the syntactic structure of terms. □

Let us now briefly comments on our normal forms. A regular expression is either provably equal to 0 or to an expression not containing 0's, with a number of 1 summands. This multiplicity is due to the fact the classical axiom $X + X = X$ does not hold within our framework. Its absence has been fully motivated in [11], by resorting to parallels with resource based interpretation and fault tolerance. One can notice is probably equal to 0 or to an expression not containing 0. A regular expression does not have unique normal form and a normal form can be obtained by using *-free axioms only.

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. To give an intuition, this norm will correspond to the maximal sum of branchings along a minimal run reached by a derivative of the given term.

The next definition introduces the set of derivatives of a regular expression $E$.

**Definition 10.** Let $E$ be a regular expression not containing 0, $\mathcal{I}$ be a set of regular expressions, and $\mathcal{I}{\bullet}E$ be defined by $\{F{\bullet}E | F \in \mathcal{I}\}$,

1. the set of immediate derivatives of $E$, *ider*$(E)$, is inductively defined by:

$$\begin{array}{rcl} ider(1) & = & \emptyset \\ ider(a) & = & \{1\} \\ ider(E+F) & = & ider(E) \cup ider(F) \\ ider(E{\bullet}F) & = & \begin{cases} ider(E){\bullet}F & \text{if } ider(E) \neq \emptyset \\ ider(F) & \text{otherwise} \end{cases} \\ ider(E^*) & = & ider(E){\bullet}E^*. \end{array}$$

   If $E \in ider(F)$, we shall write $F \succ E$.

2. The set of derivatives of $E$, *der*$(E)$ (the transitive closure of $\succ$), is inductively defined by:

$$\begin{array}{rcl} der(1) & = & \emptyset \\ der(a) & = & \{1\} \\ der(E+F) & = & der(E) \cup der(F) \\ der(E{\bullet}F) & = & der(E){\bullet}F \cup der(F) \\ der(E^*) & = & der(E){\bullet}E^*. \end{array}$$

**Remark 1.** Two derivatives of an expression are considered as different if they are syntactically different and remain such also after application of (C1), (C2), (C3) and of (S2), (S3).

**Lemma 2.** *Let $E$ not containing 0 be such that $wf(E)$ holds and $F \in der(E)$. Then, either $wf(F)$ holds or $F$ is 1. Moreover, $1 \in der(E)$.*

*Proof.* The proof goes by induction on the structure of $E$. □

A notion of head normal form will also be needed in the sequel, and crucial for establishing correspondence results between subterms and their semantic interpretation.

**Definition 11.** (*Head normal forms*)
A head normal form is either 0 or a term of the form

$$\sum_{i \in I} a_i + \sum_{j \in J} a_j{\bullet}E_j + \sum_{k \in K} F_k$$

where for all $k$, $F_k = 1$ and every $E_j$ is an expression different from 0 and 1.

**Lemma 3.** *Every term $E$, can be transformed, via the laws of Table 2, into the head normal form* 0 *or 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 $E_j \in ider(E)$ for every $j \in J$.*

*Proof.* The proof follows similar lines of that of Lemma 1. 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 $hnf(S \bullet S^*)$. □

**Remark 2.** $hnf(E)$ is unique up to axioms (C1), (C2), (C3), (S2), (S3). The cardinality of $K$, *i.e.* the number of 1 summands in $hnf(E)$ corresponds to the number of trivial runs of the tree produced by the semantic interpretation.

Now we want to see the impact of the definitions above on the model. Hence we will describe derivatives semantically. The same will be done in the sequel *via* suitable remarks.

The tree corresponding to the $\mathcal{T}$-interpretation of a derivative of a regular expression $E$ can be obtained from the tree corresponding to the $\mathcal{T}$-interpretation of $E$ as follows.

Let $E$ be a regular expression and $t = (X, \alpha, \beta)$ be the tree corresponding to $E$ according to interpreting function ($\mathcal{T}[\![E]\!] = t$). Consider $x$ be a run in $t$ and $s$ be such that $\epsilon < s \leq e(x)$. If we let $s - s'$ stand for the result of erasing the prefix $s'$ from $s$, the derivative $t' = (X', \alpha', \beta')$ of $t$ along $x$ after $s$, written $t[x, s>$, is defined by:

$$\begin{array}{rcl} X' & = & \{y \in X \mid \beta(x,y) \geq s\} \\ \alpha'(y) & = & \alpha(y) - s \text{ for every } y \in X' \\ \beta' & = & \beta(y,z) - s \text{ for every } y, z \in X'. \end{array}$$

**Lemma 4.** *Let $t$ be the tree interpreting the term $E$. Then every derivative $t'$ of $t$ along $x$ after $s$, $t[x, s>$, is the interpretation of a derivative of $E$.*

*Proof.* The proof goes by induction on the length of $s$. Let such length be 1. Since $t$ is an interpretation of $hnf(E)$ as well, we have that $t[x, a>$, defined as above, is either 1 or $E_j$ depending on whether $a$ is one of the $a_i$ or one of the $a_j$. If we assume the assert for $s$ of length $n$, then every derivative of $t$ for $s$ of length $n+1$ corresponds to an immediate derivative of a derivative of $E$. □

We need also the notion of initial branching of a term (intuitively corresponding to the number of initial branches of the tree obtained by interpreting the term).

**Definition 12.** Given a term $E$ not containing 0 such that its head normal form is:

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

The initial branching of $E$, $br(E)$, is defined below; there $n$ is used to denote the cardinality of $K$:

$$\begin{aligned} br(1) &= 1 \\ br(a) &= 1 \\ br(E+F) &= br(E) + br(F) \\ br(E \bullet F) &= br(E) + n \times br(F) \\ br(E^*) &= 1 + br(E). \end{aligned}$$

**Definition 13.** A complete chain of derivatives of $F$ is a sequence of terms $F_1 \succ F_2 \ldots \succ F_n$, with $F \succ F_1$, such that the head normal form of each $F_i$ does not have 1-summands while the head normal form of $F_n$ does.

Every term (different from 0 and 1), gives rise to a finite complete chain of derivatives. This can be easily seen by structural induction.

First we define the *size* of a term $E$ along one of the *shortest* complete chains $c$:

$$s(E, c) = br(E) + \sum_{E_j \in c} br(E_j).$$

Then define the norm of $E$ as

$$|E| = \max_{(i,c)} s(E'_i, c)$$

where $c$ is one of the shortest complete chains of derivatives $E'_i \in der(E)$. Intuitively, $s(E, c)$ is the sum of branchings along a minimal run; therefore $|E|$ is the maximal size reachable by one of its derivatives.

**Remark 3.** The notions of branching and size are invariant with respect to resource equivalence.

As shown in [10] for the weaker notion of non-empty word property in the sense of Salomaa, we have the following statement:

**Proposition 4.** *A tree corresponding to an expression satisfying the boundedness predicate is finitely branching. Moreover, a tree corresponding to a $wf$ expression does not have two different runs $x_1$, $x_2$ such that $\beta(< x_1, x_2 >) = \alpha(x_1)$.*

Let us show now some critical properties of our norm on processes. The first one states that the subterms of a regular expression have a norm that is smaller or equal to the one of the expression itself. The relevant property is that $|E| < |E^*|$.

**Proposition 5.** *Let $E$ be a regular expression. Then the following inequalities hold:*

$$\begin{array}{llllllll} (1) & |E| & \leq & |E+F| & (2) & |F| & \leq & |E+F| \\ (3) & |E| & \leq & |E{\bullet}F| & (4) & |F| & \leq & |E{\bullet}F| \\ (5) & |E| & < & |E^*|. & & & & \end{array}$$

*Proof.* The only critical case is (5); this, we are going to prove. The other cases are simpler. Let us remark that, since $wf(E)$ holds, the initial branching of derivatives of $E$ and in $E^*$ along a minimal complete chain do coincide apart from the last derivative. This is 1 for $E$ and $1 + br(E)$ for $E^*$. Indeed, if $E' \in der(E)$ then $E'{\bullet}E^* \in der(E^*)$ and, for the boundedness property, they have the same initial branching, if $E'$ is not 1 (see Lem. 2). □

We can now prove that the norm of a derivative of a regular expression is smaller or equal to the one of the expression itself and that norm is preserved by resource equivalence.

**Proposition 6.** *Let $E$ be a regular expression, $E' \in der(E)$ and $\sim$ denote resource equivalence. Then:*
(1) $|E'| \leq |E|$,
(2) $E \sim F$ *implies* $|E| = |F|$.

*Proof.* (1) The two norms are the maximum with respect to two sets such that one is included in the other.
(2) Semantically equivalent terms are interpreted by isomorphic trees, where norm corresponds to sum of branching along a minimal non trivial run. □

The norm and the derivative of a process are used to define an ordering over processes.

**Definition 14.** We say $E < F$ if and only if

$$|E| < |F| \qquad \text{or} \qquad E \in der(F) \text{ and } F \notin der(E).$$

Strong preorder $<$ over regular expressions induces a well-founded partial order on classes of terms (denoted with the same symbol), where two terms are in the same class if they can derive each other. The proof of this fact is borrowed from [15] –Lemma 7–. We report the proof for completeness.

**Lemma 5.** *$<$ is a well-founded ordering on regular expressions.*

*Proof.* If $E \in der(F)$ then every derivative of $E$ is also a derivative of $F$. Thus $|E| \leq |F|$. Hence, $E < F$ implies $|E| \leq |F|$.

Suppose that $<$ is not well-founded, so there exists an infinite chain ... $< E_2 < E_1 < E_0$. For all $n$ we have $|E_{n+1}| \leq |E_n|$. Then there must exist an $N$ such that $|E_N| = |E_n|$ for all $n > N$. By the definition of $<$, $E < F$ and $|E| = |F|$ imply $E \in der(F)$ (and $F \notin der(E)$). But each regular expression has only finitely many derivatives so, in our infinite chain, there are $m, n > N$ with $m < n$ and $E_m = E_n$. Then $E_n \not< E_m$ contradicting the hypothesis. □

Another crucial property we need, states that each derivative $F'$ of $F$, cannot derive $E^*F$. This lemma is adapted from [15] –Lemma 8–.

**Lemma 6.** *If $F' \in der(F)$ then $F' < E^*F$.*

*Proof.* We need a function $g$ defined, for terms not containing 0, as:

$$\begin{array}{rcl} g(1) & = & 0 \\ g(a) & = & 0 \\ g(E+F) & = & \max\{g(E), g(F)\} \\ g(E \bullet F) & = & \max\{g(E), g(F)\} \text{ if } E \text{ is not a } (\_)^*\text{-term} \\ g(E^* \bullet F) & = & \max\{g(E), g(F)+1\} \cdot \end{array}$$

By structural induction one can prove that $F' \in der(F)$ implies $g(F') \leq g(F)$ for every regular expression $F$. Boundedness hypothesis is crucial to prove that $g(G' \bullet G^* \bullet H) = g(G^* \bullet H)$, where $G' \in der(G)$. Indeed, by the boundedness predicate, a derivative of a term $G$ in $G^* \bullet H$ cannot ends with a star term. Hence, by the definition of function $g$, $g(G' \bullet G^* \bullet H) = \max\{g(G'), g(G^* \bullet H)\} = g(G^* \bullet H)$.

Therefore, $g(F') \leq g(F) < g(E^* \bullet F)$, so $E^* \bullet F$ cannot be a *derivative* of $F'$ while $F' \in der(E^* \bullet F)$. Then $F' < E^*F$. □

We need a lemma analogous to Lemma 9 in [15]. We shall call recursive a term of the form $G'G^*H$, with $G' \in der(G)$.

**Lemma 7.** *Let $E$ be a recursive term or in normal form and let $E'$ be such that $E \succ E'$. Then either $E' < E$ or $E$ and $E'$ are both recursive and they derive each other.*

*Proof.* Let us first remark that, if $E'$ is a derivative of $E$ then $|E'| \leq |E|$, therefore we have two mutually excluding cases:

- $E' < E$ and, in this case, $E'$ cannot derive $E$
- $|E'| = |E|$ and the two terms derive each other.

Then proving the lemma essentially reduces to prove the following statement: if two terms derive each other, they are recursive. By Lemma 6, in a finite chain of immediate derivatives such that the first term and the last one are equal (we will call it a loop), if a term is recursive, then all terms are recursive too. In fact norm cannot decrease along the loop, therefore the immediate derivative of a recursive term, say $G'G^*H$, cannot be a derivative of $H$ and it will be recursive; then all terms are recursive because we are dealing with a loop. In particular the lemma is true for recursive terms. Hence we are left to prove that a non recursive term $E$ in normal form cannot be derived by a derivative of the same type, *i.e.* it does not exist a loop of immediate derivatives for it. This last claim can be proved by structural induction. Suppose

$$E = \sum_{i \in I} a_i \bullet n_i + \sum_{j \in J} R_j^* \bullet S_j + \sum_{m \in M} E_m$$

where $E_m = 1$ for all $m$ and $n_i$, $S_j$ are either 1 or normal forms different from 0 and 1 and $E$ is not just a summand consisting of a recursive form.

For the base cases, namely $E = 0$ and $E = \sum_{m \in M} E_m$ of (non recursive) normal forms the assert is true. Suppose now by contradiction that for a given more complex $E$ there is a loop beginning with $E'$. Since $E \succ E'$ and it is a not recursive normal form, we have that the only possibility is: $E' = n_i$ for some $i$ and $n_i$ not recursive.

The supposed loop, would provide a loop also for $E'$, but, as a proper subterm of $E$, the claim is true for $E'$, hence we have a contradiction. □

Another useful property is a right-cancellative property for sequential composition; namely, whenever $E \bullet F \sim G \bullet F$ then $E \sim G$. Let us notice that trees of our model are finite branching and, therefore, they cannot have infinitely many runs of the same length. Cancellative property for coproduct is valid for the same reason.

**Lemma 8.** *If $E \bullet F \sim G \bullet F$ with none of $E, G, F$ resource equivalent to 0, then $E \sim G$.*

*Proof.* Let us call depth of a tree the length of one of its minimal runs. $E$ and $G$ must have the same depth. If this depth is 0, then they have a certain number of trivial runs, but, by the cancellative property for coproduct, they have the same amount of trivial runs, so that we can ignore them and restrict our reasoning to trees where depth is greater than 0. If $t = \mathcal{T}[\![E \bullet F]\!]$ and $t' = \mathcal{T}[\![G \bullet F]\!]$ are isomorphic, being finite branching, we have that they are made out of finitely many derivatives of the form $t[x, a_i >$ and $t'[x', a_i >$ pairwise isomorphic. Since their depth is greater than 0, we can assume that $x = (y, z)$ with $y$ a non trivial run in $\mathcal{T}[\![E]\!]$, and, analogously $x' = (y', z')$ with $y'$ a non trivial run in $\mathcal{T}[\![G]\!]$. Let us take such a non trivial run $y$ of length $n$, by repeatedly applying the reasoning above, after $n$ steps, we can have a semantical derivative where $y$ has become trivial and it will correspond in the induced isomorphism to a run obtained from $\mathcal{T}[\![G]\!]$ in the same way and trivial as well. We have only finitely many of such runs, so that we can inductively define a bijective correspondence between them extending to an isomorphism between $\mathcal{T}[\![E]\!]$ and $\mathcal{T}[\![G]\!]$. □

**Lemma 9.** *Let $E$ and $F$ be regular expressions in normal form such that $E \sim F$. 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 $F' < F$*

| | | | | |
|---|---|---|---|---|
| $E_0$ | $0$ | $=$ | $0$ | |
| $E_1$ | $1$ | $=$ | $1$ | |
| $E_2$ | $a$ | $=$ | $a$ | |
| $E_3$ | $a \bullet E'$ | $=$ | $a \bullet F'$ | |
| $E_4$ | $R^* \bullet S$ | $=$ | $T^* \bullet U$ | |
| $E_5$ | $R' \bullet R^* \bullet S$ | $=$ | $T^* \bullet U$ | $R' \in der(R)$ |
| $E_6$ | $R' \bullet R^* \bullet S$ | $=$ | $T' \bullet T^* \bullet U$ | $R' \in der(R)$ *and* $T' \in der(T)$. |

*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:

- as in cases $E_0$, $E_1$, $E_2$ or
- as in case $E_3$ 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 7, 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 be necessarily recursive. □

We are now ready to prove our main statement.

**Theorem 2.** *Let $E$ and $F$ be regular expressions such that $E \sim F$. Then $E =_{Ax} F$.*

*Proof.* The proof is by induction on the product ordering over $\mathcal{E} \times \mathcal{E}$. By Lemma 1 we can assume that $E$ and $F$ are in normal form. By Lemma 9, we have to consider at most the equations $E_0$, $E_1$, $E_2$, $E_3$, $E_4$, $E_5$ and $E_6$. But $E_0$, $E_1$ 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 Lem. 3)

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

of $R^*{\bullet}S$ and $T^*{\bullet}U$ 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 resource equivalent 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 Proposition 8.
(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 subsets:

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

and

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

Similarly, $J$ can be divided:

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

and

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

If $I_1$ 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

$$\sum_{i\in I_1} R_i{\bullet}R^*{\bullet}S + S =_{Ax} U . \tag{1}$$

In order to derive equation (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

$$R_0 = T . \tag{2}$$

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

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

$E_5$ 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 2, $R'$ is well-formed. For this reason, it cannot be the case that 1 is a summand of the head normal form of $R'$. Thus 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 resource equivalent 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 $|R^* \bullet S| = |R' \bullet R^* \bullet S|$ or $|R^* \bullet S| = |R'' \bullet R^* \bullet S|$ (because $R^* \bullet S$, $R' \bullet R^* \bullet S$ and $R'' \bullet R^* \bullet S$ have the same derivatives). 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, $R' \bullet R^* \bullet S =_{Ax} (T^* \bullet V) \bullet (R^* \bullet S)$ and hence, by Proposition 8, $R' \sim T^* \bullet V$. Since $|R'| < |R' \bullet (R^* \bullet S)|$ and $|T^* \bullet V| \leq |T^* \bullet U|$ induction yields $R' =_{Ax} T^* \bullet V$. 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 Proposition 8, $R' \bullet T'' \sim T'$. Moreover, $|R' \bullet T''| \leq |R' \bullet T'' \bullet T^* \bullet U| = |T' \bullet T^* \bullet U| = |R' \bullet R^* \bullet S|$ so that $|R' \bullet T''| \leq |R' \bullet R^* \bullet S|$ and $|T'| < |T' \bullet T^* \bullet U| = |T'' \bullet T^* \bullet U|$. 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) Suppose that both $R^* \bullet S \sim U'$, for some $U' \in der(U)$, and $T^* \bullet U \sim S'$, for some $S' \in der(S)$, hold.
Then, by Lemma 4, there are $U'' \in der(U)$ and $S'' \in der(S)$ such that $R' \bullet R^* \bullet S \sim U''$ and $T' \bullet T^* \bullet U \sim S''$. But $U'' < T' \bullet T^* \bullet U$ and $S'' < R' \bullet R^* \bullet S$. Thus, $R' \bullet R^* \bullet S =_{Ax} U'' =_{Ax} S'' =_{Ax} T' \bullet T^* \bullet U$ immediately follows. □

## References

[1] L. Aceto and W. Fokkink, An Equational Axiomatization for multi-exit iteration. *Inform. and Comput.* **134** (1997) 121-158.

[2] L. Aceto, W. Fokkink, R. van Glabbeek and A. Ingólfsdóttir, Axiomatizing prefix iteration with silent steps. *Inform. and Comput.* **127** (1996) 26-40.

[3] L. Aceto, W. Fokkink and A. Ingólfsdóttir, A managerie of non-finitely based process semantics over BPA*: From ready simulation to completed traces. Research Report, BRICS, RS-96-23 (1996).

[4] J.C.M. Baeten and J.A. Bergstra, Process Algebra with a Zero Object, in *Proc. of Concur'90.* Springer, *Lecture Notes in Comput. Sci.* **458** (1990) 83-98.

[5] L. Bernatsky, S.L. Bloom, Z. Ésik and Gh. Stefanescu, Equational theories of relations and regular sets, in *Proc. of Words, Languages and Combinatorics*, Kyoto, 1992. Word Scientific, (1994) 40-48.

[6] S.L. Bloom and Z. Ésik, Iteration Algebras of Finite State Process Behaviours. Manuscript.

[7] S.L. Bloom, Z. Ésik and D. Taubner, Iteration theories of synchronization trees. *Inform. and Comput.* **102** (1993) 1-55.

[8] M. Boffa, Une remarque sur les systèmes complets d'identités rationelles. *Theoret. Informatics Appl.* **24** (1990) 419-423.

[9] J.H. Conway, *Regular Algebra and Finite Machines.* Chapman and Hall, London (1971).

[10] F. Corradini, R. De Nicola and A. Labella, Fully Abstract Models for Nondeterministic Regular Expressions, in *Proc. of Concur'95*. Springer Verlag, *Lecture Notes in Comput. Sci.* **962** (1995) 130-144.

[11] F. Corradini, R. De Nicola and A. Labella, Models of Nondeterministic Regular Expressions. *J. Comput. System Sci.*, to appear.

[12] R. De Nicola and A. Labella, Tree Morphisms and Bisimulations, in *Proc. of MFCS'98 Workshop on Concurrency*. Elsevier, Amsterdam, *Electron. Notes Theoret. Comput. Sci.* **18** (1998).

[13] R. De Nicola and A. Labella, A Completeness Theorem for Nondeterministic Kleene Algebras, in *Proc. of MFCS'94*. Springer, *Lecture Notes in Comput. Sci.* **841** (1994) 536-545.

[14] W. Fokkink, A complete equational axiomatization for prefix iteration. *Inform. Process. Lett.* **52** (1994) 333-337.

[15] W. Fokkink, On the completeness of the Equations for the Kleene Star in Bisimulation, in *Proc. of AMAST'96*. Springer, *Lecture Notes in Comput. Sci.* **1101** (1996) 180-194.

[16] W. Fokkink, Axiomatizations for the perpetual loop in Process Algebras, in *Proc. of ICALP'97*. Springer, *Lecture Notes in Comput. Sci.* **1256** (1997) 571-581.

[17] W. Fokkink and H. Zantema, Basic process algebra with iteration: Completeness of its equational axioms. *Comput. J.* **37** (1994) 259-267.

[18] C.A.R. Hoare, *Communicating Sequential Processes*. Prentice Hall (1989).

[19] S.C. Kleene, Representation of Events in Nerve Nets and Finite Automata, in *Automata Studies*, Shannon and McCarthy, Ed. Princeton Univ. Pr. (1956) 3-41.

[20] S. Kasangian and A. Labella, Enriched Categorical Semantics for Distributed Calculi. *J. Pure Appl. Algebra* **83** (1992) 295-321.

[21] D. Kozen, A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. *Inform. and Comput.* **110** (1994) 366-390.

[22] D. Krob, Complete systems of B-rational identities. *Theoret. Comput. Sci.* **89** (1991) 207-343.

[23] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*. Springer, Berlin, *Monogr. Theoret. Comput. Sci. EATCS Ser.* **5** (1986).

[24] R. Milner, *A Calculus of Communicating Systems*. Springer-Verlag, Berlin, *Lecture Notes in Comput. Sci.* **94** (1980).

[25] R. Milner, A complete inference system for a class of regular behaviors. *J. Comput. System Sci.* **28** (1984) 439-466.

[26] R. Milner, *Communication and Concurrency*. Prentice Hall (1989).

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

[28] D. Park, Concurrency and Automata on Infinite sequences, in *Proc. GI*. Springer, *Lecture Notes in Comput. Sci.* **104** (1981) 167-183.

[29] B.C. Pierce, *Basic Category Theory for Computer Scientists*. The MIT Press (1991).

[30] V.N. Redko, On defining relations for the algebra of regular events (Russian). *Ukrain. Mat. Z.* **16** (1964) 120-126.

[31] J. Sakarovitch, Kleene's theorem revisited, in *Proc. of Trends, techniques, and problems in theoretical computer science*. Springer, *Lecture Notes in Comput. Sci.* **281** (1986) 39-50.

[32] A. Salomaa, Two Complete Axiom Systems for the Algebra of Regular Events. *J. ACM* **13** (1966) 158-169.

[33] P. Sewell, Bisimulation is not finitely (first order) equationally axiomatisable, in *Proc. of LICS'94* (1994).

[34] M.B. Smith and G.D. Plotkin, The category-Theoretic Solution of Recursive Domain Equation. *SIAM J. Comput.* **11** (1982) 762-783.

[35] D.R. Troeger, Step bisimulation is pomset equivalence on a parallel language without explicit internal choice. *Math. Structures Comput. Sci.* **3** (1993) 25-62.

Received November 30, 1998. Revised June 10, 1999.