Using Proofs by Coinduction to Find "Traditional" Proofs

Clemens Grabmayer

Department of Computer Science, Vrije Universiteit Amsterdam,
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
clemens@cs.vu.nl
http://www.cs.vu.nl/~clemens

Abstract. In the specific situation of formal reasoning concerned with "regular expression equivalence" we address instances of more general questions such as: how can coinductive argumentation be formalised logically and be applied effectively, as well as how is it linked to traditional forms of proof. For statements expressing that two regular expressions are language equivalent, we demonstrate that proofs by coinduction can be formulated in a proof system based on equational logic, where effective proof-search is possible. And we describe a proof-theoretic method for translating derivations in this proof system into a "traditional" axiom system: namely, into a "reverse form" of the axiomatisation of "regular expression equivalence" due to Salomaa. Hereby we obtain a coinductive completeness proof for the traditional proof system.

1 Introduction

Coalgebraic methods have been applied with much success in many areas of mathematics and computer science, contributing important new concepts as well as introducing fresh viewpoints at established theories. This has frequently led to the discovery of elegant new proofs of known results. Contrasting with the interest in applications, much less attention has been directed to formalising coalgebraic concepts, such as coinduction and corecursion, by using the tools of logic, and to relating these techniques with traditional methods, such as induction and recursion. This concerns also more specific questions such as whether proofs by coinduction that are formalised in an appropriate logical framework can be translated into formalised "conventional" proofs.

In this paper we consider the concrete example of a coinduction principle for proving that two regular expressions are language equivalent. We reformulate such a principle into one that can be used to decide equivalence of regular expressions effectively, and we give a logical formalisation. Furthermore, we describe a method that allows to translate proofs based on the coinduction principle into derivations in a "traditional" axiom system close to the well-known axiomatisation $\mathbf{F_1}$ of "the algebra of regular events" due to Salomaa in [7].

In [6] Rutten formulates a coinduction principle for showing equality of formal languages: to show that two languages L_1 and L_2 are equal, it suffices to prove

J.L. Fiadeiro et al. (Eds.): CALCO 2005, LNCS 3629, pp. 175–193, 2005.

that L_1 and L_2 are bisimilar in the "automaton of formal languages". Based on the differential calculus for regular expressions due to Brzozowski in [2], Rutten applies this principle to give coinductive demonstrations for a number of identities between regular expressions, and stresses the generality of this method. However, Rutten's proofs for exemplary identities use set-theoretical concepts in an essential way and do not lend themselves directly towards a formalisation in a proof system of equational logic comparable to Salomaa's axiomatisations. And to the author of the present paper some details have not been clear about why such a principle does in fact yield a generally applicable decision procedure.

Here we first introduce, by following and refining Rutten's approach, a "finitary" coinduction principle for "regular expression equivalence": to show that two regular expressions E and F are equivalent, prove that they are related by a *finite* bisimulation in a certain automaton on regular expressions whose transition function is based on the "Brzozowski derivative". We show that this principle is effective and lends itself to being mechanised. Next we introduce a natural-deduction style proof system $\mathbf{cREG_0}$ of equational logic with the property that derivations in $\mathbf{cREG_0}$ formalise, and correspond to, arguments by the finitary coinduction principle. It turns out that $\mathbf{cREG_0}$ is sound and complete with respect to regular expression equivalence. Finally, we describe an effective proof-theoretic transformation from derivations in $\mathbf{cREG_0}$ into derivations in a variant system **REG** of Salomaa's $\mathbf{F_1}$, where **REG** is the result of reversing all multiplicative parts of regular expressions in the axioms and in the rules of $\mathbf{F_1}$.

The proof system $\mathbf{cREG_0}$ we introduce is analogous in kind to an axiomatisation of "recursive type equality" introduced by Brandt and Henglein in [1] (together with its coinductive foundations) and to a system for "bisimilarity of normed recursive BPA-processes" due to Stirling given in [4] (without a coinductive motivation). All of these systems (and a number of similar, more recent ones) have in common the presence of inference rules that formalise "cyclic" forms of reasoning. Applications of such rules allow, roughly speaking, to detect that a bisimulation-building process that is formalised by a derivation has reached a subtask which it has already solved before. The transformation between $\mathbf{cREG_0}$-derivations and **REG**-derivations that we develop here was inspired by a transformation given in [5, Ch.8, Sect.8.1], where proof-theoretic relations between proof systems for "recursive type equality" are investigated.

We give a short overview of the paper: In Section 2 we define basic notions concerning regular expressions and finite automata (such as the relation "regular expression equivalence" and the notion of bisimulation). Then in Section 3 we formulate the mentioned variant system **REG** of Salomaa's axiomatisation $\mathbf{F_1}$ and define three weaker systems. In Section 4, we review the most basic notions of the "differential calculus" for formal languages and of that for regular expressions; and we relate the coinduction principle due to Rutten. Subsequently in Section 5, we formulate and prove our "finitary" version of a conduction principle for regular expression equivalence, and argue that it can be used effectively. As a formalisation of this principle, we introduce the proof system $\mathbf{cREG_0}$ in Section 6 and show that it is sound and complete. Finally in Section 7 we de-

scribe an effective proof-theoretic transformation. from $\mathbf{cREG_0}$-derivations into **REG**-derivations. In the Conclusion, Section 8, we summarise our findings and explain how similar results can be obtained that apply directly to Salomaa's $\mathbf{F_1}$.

The proofs in this paper are generally only hinted or sketched, and the methods used are instantiated in supporting examples. However, the most important proofs can be found in a technical appendix that is contained in the electronic version of this paper which is available at `http://www.cs.vu.nl/~clemens/coind2tradproofs.pdf`.

2 Regular Expressions and Deterministic Automata

Let Σ be a finite nonempty set, called *alphabet*; elements of Σ are called *letters*. By Σ^* we denote the set of (finite) *words* over Σ. The *empty word* is designated by ϵ. Concatenation of words w and w' is denoted multiplicatively as $w.w'$. A *language* over Σ is any subset of Σ^*. By $\mathcal{L}(\Sigma)$ we denote the set of languages over Σ. On $\mathcal{L}(\Sigma)$ we define the *regular operators* $+$ (*sum*), $.$ (*product*), and * (*star*), where $+$ and $.$ are binary, and * is unary: for all $L_1, L_2 \in \mathcal{L}(\Sigma)$ we let

$$L_1 + L_2 =_{\text{def}} L_1 \cup L_2\,, \qquad L_1.L_2 =_{\text{def}} \{\, w_1.w_2 \mid w_1 \in L_1,\, w_2 \in L_2 \,\}\,,$$
$$L^* =_{\text{def}} \bigcup_{n\in\omega} L^n\,, \quad \text{where} \quad L^0 =_{\text{def}} \{\epsilon\}\,, \quad \text{and}$$
$$L^{i+1} =_{\text{def}} L.L^i \quad (\text{for all } i \in \omega)$$

(by ω we denote, here and below, the natural numbers including zero).

Let $\Sigma = \{a_1, \ldots, a_n\}$ be an alphabet (from now on, such a description is generally assumed for alphabets Σ). The set $\mathcal{R}(\Sigma)$ of *regular expressions* over Σ is defined as the set of those words over Σ that are generated by the grammar

$$E \;::=\; 0 \mid a_1 \mid \ \ldots\ \mid a_n \mid E + E \mid E.E \mid E^*$$

We designate the regular expression 0^* by the symbol 1. By $\equiv$ we denote the binary relation "syntactical equality" between regular expressions. By $\sum_{i=1}^{n} E_i$ we denote, for all $n \in \omega\backslash\{0\}$ and $E_1, \ldots, E_n \in \mathcal{R}(\Sigma)$, the regular expression $E_1 + (E_2 + \ldots + (E_{n+1} + E_n))$. By a *context* C over $\mathcal{R}(\Sigma)$ we mean the result of replacing a single letter in a regular expression by a hole $[\,]$; by $C[E]$ we denote the result of hole-filling in C with the regular expression E. Every regular expression E denotes a language $L(E)$ via the function $L : \mathcal{R}(\Sigma) \to \mathcal{L}(\Sigma)$ that is inductively defined by

$$L(0) = \emptyset\,, \qquad L(a_i) = \{a_i\} \quad (1 \le i \le n)\,,$$
$$L(E + F) = L(E) \cup L(F)\,, \qquad L(E.F) = L(E).L(F) \qquad L(E^*) = L(E)^*\,.$$

(for all $E, F \in \mathcal{R}(\Sigma)$). Two regular expressions $E, F \in \mathcal{R}(\Sigma)$ are called *equivalent* (denoted by $E =_L F$) if and only if E and F denote the same formal language, i.e. iff $L(E) = L(F)$. In accordance with the notation just stipulated, we define a binary relation $=_L$ on $\mathcal{R}(\Sigma)$, called *regular expression equivalence*, by $=_L =_{\text{def}} \{\langle E, F\rangle \in \mathcal{R}(\Sigma) \times \mathcal{R}(\Sigma) \mid L(E) = L(F)\}$.

Let A be a (possibly) infinite set of *input symbols*. A *(deterministic) automaton* with input alphabet A is a triple $S = \langle S, o, t\rangle$ consisting of a set S of *states*, an *output function* $o : S \to 2$, and a *transition function* $t : S \to S^A$, where S^A denotes the set of all functions from A to S, and $2 = \{0, 1\}$ (in this set 0 and 1 are usually numbers, but for convenience[1] we agree to consider them as regular expressions here). The output function o indicates whether a state s is S is *accepting* (if $o(s) = 1$) or not (if $o(s) = 0$). The transition function t assigns to a state s a function $t(s) : A \to S$ which defines the state $t(s)(a)$ that is reached by S after reading input symbol a. Sometimes we write $s\!\downarrow$ for $o(s) = 1$, $s\!\uparrow$ for $o(s) = 0$, and $s \xrightarrow{a} s'$ for $t(s)(a) = s'$.

Let $S = \langle S, o, t\rangle$ and $S' = \langle S, o', t'\rangle$ be automata. A *homomorphism* between S and S' is a function $f : S \to S'$ such that, for all $s \in S$ and $a \in A$, $o(s) = o'(f(s))$ and $f(t(s)(a)) = t'(f(s))(a)$ holds. A *bisimulation* between S and S' is a nonempty relation $R \subseteq S \times S'$ such that for all $s \in S$, $s' \in S'$, and $a \in A$

$$s\ R\ s' \quad \Longrightarrow \quad o(s) = o'(s) \quad \text{and} \quad t(s)(a)\ R\ t'(s')(a)$$

holds. For $s \in S$ and $s' \in S'$ we write $s \sim s'$ if there exists a bisimulation R with sRs'; if there exists a finite bisimulation R with sRs', we write $s \sim_{\text{fin}} s'$.

3 The Axiom System REG

The first complete axiomatisations of regular expression equivalence were given by Salomaa in [7]. Here, our investigations will be based on Salomaa's first system $\mathbf{F_1}$. However, we introduce a variant system **REG** that arises from $\mathbf{F_1}$ essentially[2] by reversing all multiplicative expressions in axioms and rules. The reason is that, while having analogous properties as $\mathbf{F_1}$, the system **REG** will turn out to lend itself much better to establish a connection with the differential calculus for regular expressions in its usual form (as described in Section 4).

Let Σ be an alphabet. The axiom system $\mathbf{REG}(\Sigma)$ is defined as follows: its *formulas* are equations $E = F$ between regular expression E and F on Σ; its *axioms* are the formulas that belong to one of the schemes (B1)–(B11) listed in Figure 1; and its *inference rules* are the four rules SYMM, TRANS, CTXT, and FIX whose applications are schematically defined in Figure 1 (reflexivity axioms are not used in this definition as they can easily be recognised to be derivable).

Derivations in $\mathbf{REG}(\Sigma)$ are prooftrees, that is, finite upwards-growing labeled trees such that: all nodes are labeled by formulas of $\mathbf{REG}(\Sigma)$, the leaves at the top carry axioms of $\mathbf{REG}(\Sigma)$, and each internal node ν is labeled by a formula that is the conclusion of an application of a $\mathbf{REG}(\Sigma)$-rule with the formula(s) that label(s) the immediate successor(s) of ν as premises; the bottommost formula of a prooftree is called its *conclusion*. For $E, F \in \mathcal{R}(\Sigma)$, we denote by $\vdash_{\mathbf{REG}(\Sigma)} E = F$ the statement that there exists a derivation in $\mathbf{REG}(\Sigma)$

[1] We want to be able to view outcomes of output functions as regular expressions.

[2] A less important change consists in dropping the substitution rule R2 specific to $\mathbf{F_1}$ in favour of the symmetry, transitivity, and context rules of equational logic.

The *axioms* of $\mathbf{REG}(\Sigma)$:

(B1)	$E+(F+G)=(E+F)+G$	(B7)	$E.1=E$
(B2)	$(E.F).G=E.(F.G)$	(B8)	$E.0=0$
(B3)	$E+F=F+E$	(B9)	$E+0=E$
(B4)	$(E+F).G=E.G+F.G$	(B10)	$E^*=1+E.E^*$
(B5)	$E.(F+G)=E.F+E.G$	(B11)	$E^*=(1+E)^*$
(B6)	$E+E=E$		

The *inference rules* of $\mathbf{REG}(\Sigma)$:

$$\frac{E=F}{F=E}\ \text{SYMM} \qquad \frac{E=G \qquad G=F}{E=F}\ \text{TRANS}$$

$$\frac{E=F}{C[E]=C[F]}\ \text{CTXT} \qquad \frac{E=F.E+G}{E=F^*.G}\ \text{FIX}\ \ (\text{if } o(F)=0 \text{ [cf. Sect. 4]})$$

Fig. 1. The axiom system $\mathbf{REG}(\Sigma)$ for regular expression equivalence, which results from Salomaa's system $\mathbf{F_1}$ by reversing multiplicative expressions

with conclusion $E=F$. (We sometimes write **REG** in place of $\mathbf{REG}(\Sigma)$.)

The following theorem can be proved analogously to Salomaa's result for $\mathbf{F_1}$.

Theorem 1. *The system* $\mathbf{REG}(\Sigma)$ *is sound and complete with respect to regular expression equivalence. More formally, it holds:*

$$\textit{for all } E,F\in\mathcal{R}(\Sigma): \quad \big[\ \vdash_{\mathbf{REG}(\Sigma)} E=F \quad\Longleftrightarrow\quad E=_L F\ \big]\,. \tag{1}$$

For later use, we define three systems that are weaker than $\mathbf{REG}(\Sigma)$, but closely related: by $\mathbf{REG}^-(\Sigma)$ we designate the axiom system that results from $\mathbf{REG}(\Sigma)$ by excluding the rule FIX; by $\mathbf{ACI}(\Sigma)$ we denote the subsystem of $\mathbf{REG}^-(\Sigma)$ that contains only the axioms (B1), (B3), and (B6) for associativity, commutativity, and idempotency of $+$; and by $\mathbf{ACI}^+(\Sigma)$ we denote the extension of $\mathbf{ACI}(\Sigma)$ that contains of all the axioms (B1)–(B9) and furthermore

$$(\text{B7})^R \quad 1.E=E \qquad\qquad \text{and} \qquad\qquad (\text{B8})^R \quad 0.E=0\,,$$

but that does not contain the rule FIX. For each of these three systems, we define binary relations on $\mathcal{R}(\Sigma)$ that denote "equality is derivable" in the respective system: for instance, we stipulate, for all $E,F\in\mathcal{R}(\Sigma)$,

$$E\equiv_{\mathbf{ACI}^+} F \quad\Longleftrightarrow_{\text{def}}\quad \vdash_{\mathbf{ACI}^+(\Sigma)} E=F\,; \tag{2}$$

the relations $\equiv_{\mathbf{ACI}}$ and $\equiv_{\mathbf{REG}^-}$ are defined analogously. It is easy to verify that all three relations are congruence relations on $\mathcal{R}(\Sigma)$. For all $E\in\mathcal{R}(\Sigma)$, we respectively denote by $[E]_{\mathbf{ACI}}$, $[E]_{\mathbf{ACI}^+}$, and $[E]_{\mathbf{REG}^-}$ the $\equiv_{\mathbf{ACI}}$-, $\equiv_{\mathbf{ACI}^+}$- and $\equiv_{\mathbf{REG}^-}$-equivalence classes of E. And by $\mathcal{R}(\Sigma)_{\mathbf{ACI}}$, $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$, and $\mathcal{R}(\Sigma)_{\mathbf{REG}^-}$ we denote by factor sets of $\mathcal{R}(\Sigma)$ with respect to $\equiv_{\mathbf{ACI}}$, $\equiv_{\mathbf{ACI}^+}$, and $\equiv_{\mathbf{REG}^-}$.

4 The Differential Calculus for Regular Expressions

In this section we review the basic notions of a differential calculus for formal languages, as for example described by Conway [3, Ch.5], and for regular expressions, due to Brzozowski in [2]. We also state two coinduction principles.

Let Σ be an alphabet, and $L \in \mathcal{L}(\Sigma)$. For all words $w \in \Sigma^*$, the *w-derivative* of L is $L_w =_{\text{def}} \{\, v \in \Sigma^* \mid w.v \in L \,\}$. In the special case of letters $a \in \Sigma$ the *a-derivative* L_a can be used to turn the set $\mathcal{L}(\Sigma)$ of languages over Σ into an automaton $\langle \mathcal{L}(\Sigma), o_{\mathcal{L}}, t_{\mathcal{L}} \rangle$ by defining, for all $L \in \mathcal{L}(\Sigma)$ and $a \in \Sigma$,

$$o_{\mathcal{L}}(L) =_{\text{def}} \begin{cases} 1 \ldots \epsilon \in L \\ 0 \ldots \epsilon \notin L \end{cases} \qquad \text{and} \quad t_{\mathcal{L}}(L)(a) =_{\text{def}} L_a \,.$$

In [6, Section 4] Rutten shows the *coinduction principle* for proving equality of formal languages that is stated by the following proposition.

Proposition 1. *For all $L_1, L_2 \in \mathcal{L}(\Sigma)$ it holds:*

$$L_1 \sim L_2 \ \text{ in } \mathcal{L}(\Sigma) \quad \Longrightarrow \quad L_1 = L_2 \,. \tag{3}$$

That is: to show $L_1 = L_2$ for two languages L_1 and L_2 over Σ, it suffices to demonstrate that L_1 and L_2 are bisimilar in the automaton $\mathcal{L}(\Sigma)$.

From Proposition 1 a similar proof principle for showing equivalence of regular expressions can be extracted by using the "Brzozowski derivative". This concept, here just called "derivative", allows to mimic language derivatives on regular expressions. Let again Σ be an alphabet. For all $a \in \Sigma$, and $G \in \mathcal{R}(\Sigma)$, the *a-derivative* G_a of a regular expression G over Σ is defined inductively by

$$0_a =_{\text{def}} 0 \,, \qquad (E+F)_a =_{\text{def}} E_a + F_a \,, \qquad (E^*)_a =_{\text{def}} E_a.E^* \,,$$

$$b_a =_{\text{def}} \begin{cases} 1 & \ldots\ b = a \\ 0 & \ldots\ b \neq a \end{cases} \qquad (E.F)_a =_{\text{def}} \begin{cases} E_a.F + F_a & \ldots\ o(E) = 1 \\ E_a.F & \ldots\ o(E) = 0 \end{cases}$$

(for all $b \in \Sigma$ and $E, F \in \mathcal{R}(\Sigma)$). In a similar way, the function $o : \mathcal{R}(\Sigma) \to 2$ is inductively defined by (for all $b \in \Sigma$ and $E, F \in \mathcal{R}(\Sigma)$)

$$o(0) =_{\text{def}} 0 \,, \qquad o(b) =_{\text{def}} 0 \,, \qquad o(E+F) =_{\text{def}} \begin{cases} 0 & \ldots\ o(E) = o(F) = 0 \\ 1 & \ldots\ \text{else} \end{cases}$$

$$o(E.F) =_{\text{def}} \begin{cases} 1 & \ldots\ o(E) = o(F) = 1 \\ 0 & \ldots\ \text{else} \,, \end{cases} \qquad o(E^*) =_{\text{def}} 1 \,.$$

We also define, for all $w \in \Sigma$ and $E \in \mathcal{R}(\Sigma)$, the *w-derivative* of E inductively: we let $E_\epsilon =_{\text{def}} E$, and, for all $w_0 \in \Sigma^*$ and $a \in \Sigma$, $E_{w_0.a} =_{\text{def}} (E_{w_0})_a$.

Now an automaton $\mathcal{R}(\Sigma) = \langle \mathcal{R}(\Sigma), o, t \rangle$ can be formed by letting o as above and $t : \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)^\Sigma$ be defined by $t(E)(a) =_{\text{def}} E_a$ for all $a \in \Sigma$, $E \in \mathcal{R}(\Sigma)$. The function L is a homomorphism from $\mathcal{R}(\Sigma)$ to $\mathcal{L}(\Sigma)$ because, for $E \in \mathcal{R}(\Sigma)$ and $a \in \Sigma$, $L(E_a) = (L(E))_a$ and $o(E) = o_{\mathcal{L}}(L(E))$ hold (as is simple to prove). Due to this, the following statement is an easy consequence of Proposition 1.

Proposition 2. *The following* coinduction principle *holds for proving equivalence of regular expressions: for all* $E, F \in \mathcal{R}(\Sigma)$ *it holds*

$$E \sim F \text{ in } \mathcal{R}(\Sigma) \implies E =_L F \tag{4}$$

Although this principle can often be applied successfully in an informal manner (cf. the examples in [6, Section 6]), it does not itself define a general mechanisable method for deciding whether two regular expressions are equivalent. The reason is that the set of iterated derivatives of a regular expression is frequently infinite[3], and that therefore bisimulations in $\mathcal{R}(\Sigma)$ can be infinite.

5 A Finitary Coinduction Principle for $=_L$

One possible way of adopting the coinduction principle in Proposition 2 for deciding regular expression equivalence consists in refining it into a statement that only refers to finite bisimulations. As mentioned above, Proposition 2 relies on infinite bisimulations in an essential way since the number of derivatives of a regular expression may be infinite. However, it turns out that already "modulo" provability in the system **ACI** the number of derivatives of a regular expression is finite. This is stated by the second of the following two lemmas.

Lemma 1. *Let* Σ *be an alphabet, and let* $\equiv_{\mathcal{S}}$ *be one of the relations* $\equiv_{\mathbf{ACI}}$ *or* $\equiv_{\mathbf{ACI}^+}$ *on* $\mathcal{R}(\Sigma)$*. Then for all* $E, F \in \mathcal{R}(\Sigma)$ *and for all* $a \in \Sigma$ *it holds:*

$$E \equiv_{\mathcal{S}} F \implies \big(o(E) = o(F) \ \&\ E_a \equiv_{\mathcal{S}} F_a\big)\,. \tag{5}$$

Proof (Sketch). In a first step, it can be verified in a straightforward way that (5) holds for all $a \in \Sigma$, and for all $E, F \in \mathcal{R}(\Sigma)$ such that $E = F$ is an axiom of **ACI** or $\mathbf{ACI}^+$. The statement obtained hereby can then be "lifted" to apply to all $E, F \in \mathcal{R}(\Sigma)$ such that $E = F$ is a theorem of **ACI**, or of $\mathbf{ACI}^+$, by using induction on the depth of derivations in **ACI**, or respectively, in $\mathbf{ACI}^+$.

Lemma 2. *For all* $E \in \mathcal{R}(\Sigma)$*, the set* $\big\{[E_w]_{\mathbf{ACI}} \,\big|\, w \in \Sigma^*\big\}$ *is finite. As a consequence, also* $\big\{[E_w]_{\mathbf{ACI}^+} \,\big|\, w \in \Sigma^*\big\}$ *is finite for arbitrary* $E \in \mathcal{R}(\Sigma)$*.*[4]

Proof (Hint). The lemma can be shown by induction on the syntactical structure of regular expressions in $\mathcal{R}(\Sigma)$, using representation statements for w-derivatives of composite expressions like, in the case of an outermost product,

$$(\forall w \in \Sigma^*)\,(\exists V \subseteq \mathit{Suff}(w))\,\Big[\,(F.G)_w \ \equiv_{\mathbf{ACI}}\ F_w.G + \sum_{v \in V} G_v\,\Big]$$

(for all $F, G \in \mathcal{R}(\Sigma)$), where $\mathit{Suff}(w)$ means the set of all *suffixes* of w.

[3] For instance, by starting from a^* and computing the a-derivative repeatedly one is led to $1.a^*,\ 0.a^* + 1.a^*, \ldots,\ 0.a^* + \ldots (0.a^* + 1.a^*), \ldots$.

[4] The part of Lemma 2 referring to **ACI** is comparable to Theorem 5.3 by Brzozowski in [2], which statement, however, is wrong (as Salomaa rightly points out in [7]). But the reason can easily be recognised in the fact that the derivative for multiplicative expressions is defined differently in [2] than in Section 4 here: there, for all $E, F \in \mathcal{R}(\Sigma)$ and $a \in \Sigma$, $(E.F)_a = E_a.F + o(E).F_a$ is stipulated.

We have formulated these lemmas also with respect to $\mathbf{ACI}^+$, on which system we base ourselves from now on, because it seems natural to apply also other identities than those of **ACI** to simplify derivatives.[5] Relying on Lemma 1, we can now define the "factor automaton" $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+} = \langle \mathcal{R}(\Sigma)_{\mathbf{ACI}^+}, o_{\mathbf{ACI}^+}, t_{\mathbf{ACI}^+} \rangle$ of $\mathcal{R}(\Sigma)$ with respect to $\equiv_{\mathbf{ACI}^+}$ by letting

$$o_{\mathbf{ACI}^+} : \mathcal{R}(\Sigma)_{\mathbf{ACI}^+} \to 2, \quad [E]_{\mathbf{ACI}^+} \mapsto o(E) \,, \qquad t_{\mathbf{ACI}^+} : \mathcal{R}(\Sigma)_{\mathbf{ACI}^+} \to (\mathcal{R}(\Sigma)_{\mathbf{ACI}^+})^{\Sigma}, \quad [E]_{\mathbf{ACI}^+} \mapsto (a \mapsto [E_a]_{\mathbf{ACI}^+}) \,.$$

And we are finally able to formulate the following *finitary coinduction principle* for proving or disproving that two given regular expressions are equivalent.

Theorem 2. *For all* $E, F \in \mathcal{R}(\Sigma)$ *it holds:*

$$[E]_{\mathbf{ACI}^+} \sim_{fin} [F]_{\mathbf{ACI}^+} \quad \text{in } \mathcal{R}(\Sigma)_{\mathbf{ACI}^+} \quad \Longleftrightarrow \quad E =_L F \,. \tag{6}$$

Proof (Sketch). Let $E, F \in \mathcal{R}(\Sigma)$. The implication "$\Rightarrow$" in (6) is a consequence of Proposition 1 in view of the fact that the function $L^* : \mathcal{R}(\Sigma)_{\mathbf{ACI}^+} \to \mathcal{L}(\Sigma)$ which is defined by $L^*([G]_{\mathbf{ACI}^+}) \mapsto L(G)$ is a homomorphism. For the implication "$\Leftarrow$" in (6), assume $E =_L F$. Then $\{ \langle [E_w]_{\mathbf{ACI}^+}, [F_w]_{\mathbf{ACI}^+} \rangle \mid w \in \Sigma^* \}$ is a bisimulation between $[E]_{\mathbf{ACI}^+}$ and $[F]_{\mathbf{ACI}^+}$ in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$ (as is not difficult to verify); this bisimulation can easily be recognised to be finite by using Lemma 2.

As running example in this paper we consider $(a+b)^* = (a^*b)^*a^*$, a simple instance of the axiom scheme "sumstar" in a system due to Conway in [3, p.25]. We let $E^* \equiv (a+b)^*$, $F_1 \equiv (a^*b)^*a^*$, and $F_2 \equiv ((a^*b)(a^*b))^*a^* + a^*$. We find

$$(F_1)_a \equiv (((1.a^*).b + 0).(a.b)^*).a^* + 1.a^* \equiv_{\mathbf{ACI}^+} F_2 \,,$$

and in a similar way, the other entries in the following tables can be verified:

	$[(\cdot)_a]_{\mathbf{ACI}^+}$	$[(\cdot)_b]_{\mathbf{ACI}^+}$	$o_{\mathbf{ACI}^+}(\cdot)$
E	$[E]_{\mathbf{ACI}^+}$	$[E]_{\mathbf{ACI}^+}$	$\downarrow$

	$[(\cdot)_a]_{\mathbf{ACI}^+}$	$[(\cdot)_b]_{\mathbf{ACI}^+}$	$o_{\mathbf{ACI}^+}(\cdot)$
F_1	$[F_2]_{\mathbf{ACI}^+}$	$[F_1]_{\mathbf{ACI}^+}$	$\downarrow$
F_2	$[F_2]_{\mathbf{ACI}^+}$	$[F_1]_{\mathbf{ACI}^+}$	$\downarrow$

From this it follows that $R = \{\langle [E]_{\mathbf{ACI}^+}, [F_1]_{\mathbf{ACI}^+} \rangle, \langle [E]_{\mathbf{ACI}^+}, [F_2]_{\mathbf{ACI}^+} \rangle\}$ is a finite bisimulation in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$ between $[E]_{\mathbf{ACI}^+}$ and $[F_1]_{\mathbf{ACI}^+}$. Using Theorem 2, this demonstrates $(a+b)^* =_L (a^*b)^*a^*$.

Based on the next lemma it is possible to extract an effective decision procedure for regular expression equivalence from our finitary coinduction principle.

Lemma 3. *For all alphabets* Σ*, the relation* $\equiv_{\mathbf{ACI}}$ *is decidable in* $\mathcal{R}(\Sigma)$.

[5] There is some arbitrariness in choosing a system of "basic" identities that one wants to have available for simplifying derivations. For instance, $\mathbf{REG}^-$ could be used as well. $\mathbf{ACI}^+$ has been chosen here partly because of the running example we employ.

Proof (Hint). Equations between $\equiv_{\mathbf{ACI}}$-equivalent sums of regular expressions can be decomposed into equations between $\equiv_{\mathbf{ACI}}$-equivalent parts of the sums. For example, for all $E_1, E_2, F_1, F_2, F_3 \in \mathcal{R}(\Sigma)$ that are not additive expressions,

$$\begin{aligned} E_1 + E_2 \equiv_{\mathbf{ACI}} (F_1 + F_2) + F_3 \quad &\Longleftrightarrow \\ \Longleftrightarrow \quad (\exists f : \{1,2\} \to \{1,2,3\})\,(\exists g : \{1,2,3\} \to \{1,2\}) \\ (\forall i \in \{1,2\})\,(\forall j \in \{1,2,3\})\,[\, E_i \equiv_{\mathbf{ACI}} F_{f(i)} \;\;\&\;\; E_{g(j)} \equiv_{\mathbf{ACI}} F_j \,] \end{aligned}$$

holds. An obvious generalisation of this statement can be shown by structural induction on $\mathbf{ACI}(\Sigma)$-derivations, and it can be used to construct an effective (but clearly not efficient) search-algorithm that decides whether or not $E \equiv_{\mathbf{ACI}} F$ holds for given regular expressions $E, F \in \mathcal{R}(\Sigma)$.

Corollary 1. *Let Σ be an alphabet. Regular expression equivalence on $\mathcal{R}(\Sigma)$ can be decided by checking for the existence of finite bisimulations in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$.*

Proof (Sketch). Let $E, F \in \mathcal{R}(\Sigma)$ be arbitrary. It is an easy consequence of Proposition 2 that $E =_L F$ holds iff $R =_{\text{def}} \left\{ \langle [E_w]_{\mathbf{ACI}^+}, [F_w]_{\mathbf{ACI}^+} \rangle \mid w \in \Sigma^* \right\}$ is a bisimulation. Lemma 2 entails that R is always finite and that it can be determined effectively whether R is a bisimulation: for all pairs $\langle E_w, F_w \rangle$ with $w \in \Sigma^*$ and $w = b_1 \ldots b_m$ such that the list $\langle [E]_{\mathbf{ACI}^+}, [F]_{\mathbf{ACI}^+} \rangle$, $\langle [E_{b_1}]_{\mathbf{ACI}^+}, [F_{b_1}]_{\mathbf{ACI}^+} \rangle$, $\langle [E_{b_1 b_2}]_{\mathbf{ACI}^+}, [F_{b_2 b_2}]_{\mathbf{ACI}^+} \rangle$, $\ldots$, $\langle [E_w]_{\mathbf{ACI}^+}, [F_w]_{\mathbf{ACI}^+} \rangle$ does not contain a loop (this can be decided due to Lemma 3) check whether $o_{\mathbf{ACI}^+}([E_w]_{\mathbf{ACI}^+}) = o(E_w) = o(F_w) = o_{\mathbf{ACI}^+}([F_w]_{\mathbf{ACI}^+})$ holds. Because of Lemma 2 and König's lemma namely only finitely many such checks have to be performed. If one such check detects a mismatch, then R is not a bisimulation, and $E \neq_L F$ holds; if no mismatch is found, then R is a (finite) bisimulation and $E =_L F$ follows.

6 The Coinductively Motivated Proof System $\mathbf{cREG_0}$

Now we introduce a natural-deduction style proof system $\mathbf{cREG_0}(\Sigma)$ based on equational logic that allows to formalise arguments using the finitary coinduction principle for regular expression equivalence as finite derivations.

For the definition of $\mathbf{cREG_0}(\Sigma)$, we assume a countably infinite set Δ of assumption markers such that $\Sigma \cap \Delta = \emptyset$, and refer to the schemata listed in Figure 2: the *formulas* of $\mathbf{cREG_0}(\Sigma)$ are the equations between regular expressions over Σ; possible *assumptions* are formulas that have an assumption marker from the set Δ attached to them; and the *rules* of $\mathbf{cREG_0}(\Sigma)$ are the four rules $\mathrm{App}_r\mathrm{Ax}_{\mathbf{ACI}^+}$, $\mathrm{App}_l\mathrm{Ax}_{\mathbf{ACI}^+}$, COMP, and COMP/FIX that are schematically defined in Figure 2. Applications of the rule COMP/FIX have the special feature that at least one inhabited class of open assumptions is *discharged* (the marker of the assumptions belonging to this class is attached to the application).

Displaying this characteristic feature of proofs formalised in the format of natural-deduction systems (cf. the description of "N-systems" in [8]), namely the use of assumptions that may be "closed" (discharged) at a later stage in a deduction, *derivations* in $\mathbf{cREG_0}(\Sigma)$ are prooftrees such that: the leaves at the

Possible *assumptions* in $\mathbf{cREG_0}(\Sigma)$ and the *inference rules of* $\mathbf{cREG_0}(\Sigma)$*:*

$$\text{(Assm)}\ \ (E = F)^d \quad (\text{with}\ d \in \Delta)$$

$$\frac{\begin{array}{c}\mathcal{D}_1\\ C[\tilde{E}] = F\end{array}}{C[\tilde{F}] = F}\ \mathrm{App}_l\mathrm{Ax}_{\mathbf{ACI^+}} \qquad \frac{\begin{array}{c}\mathcal{D}_1\\ E = C[\tilde{E}]\end{array}}{E = C[\tilde{F}]}\ \mathrm{App}_r\mathrm{Ax}_{\mathbf{ACI^+}}$$
$$(\text{if}\ \tilde{E} = \tilde{F}\ \text{ or }\ \tilde{F} = \tilde{E}\ \text{ is an axiom of }\mathbf{ACI^+})$$

$$\frac{\begin{array}{cccc}\mathcal{D}_1 & & \mathcal{D}_n\\ E_{a_1} = F_{a_1} & \ldots & E_{a_n} = F_{a_n}\end{array}}{E = F}\ \text{COMP}\ \ (\text{if } o(E) = o(F))$$

$$\frac{\begin{array}{cccc}[E = F]^d & & [E = F]^d\\ \mathcal{D}_1 & & \mathcal{D}_n\\ E_{a_1} = F_{a_1} & \ldots & E_{a_n} = F_{a_n}\end{array}}{E = F}\ \text{COMP/FIX},\ d\quad (\text{if } o(E) = o(F))$$

Fig. 2. A coinductively motivated, natural-deduction style proof system $\mathbf{cREG_0}(\Sigma)$ for regular expression equivalence, given that $\Sigma = \{a_1, \ldots, a_n\}$

top are labeled by assumptions such that different markers are attached to different formulas[6]; assumptions may be open (undischarged) or closed (discharged); formulas at an internal node ν arise through applications of $\mathbf{cREG_0}(\Sigma)$-rules from the formulas in the immediate successors of ν, whereby in the case of COMP/FIX-applications some open assumptions are discharged; the bottommost formula is called the *conclusion*. Hereby an occurrence of an assumption $(E = F)^d$ at the top of a derivation $\mathcal{D}$ is called *open* iff on the path down to the conclusion of $\mathcal{D}$ there does not exist an application of COMP/FIX at which this assumption is discharged; otherwise the occurrence of $(E = F)^d$ is called *closed*. Assumptions in a derivation that are occurrences of the same formula with the same marker together form an *assumption class*.

For all $E, F \in \mathcal{R}(\Sigma)$, we denote by $\vdash_{\mathbf{cREG_0}(\Sigma)} E = F$ the statement that there exists a derivation $\mathcal{D}$ in $\mathbf{cREG_0}(\Sigma)$ without open assumptions such that $\mathcal{D}$ has conclusion $E = F$ (i.e. that $E = F$ is a *theorem* of $\mathbf{cREG_0}(\Sigma)$). (Sometimes we write $\mathbf{cREG_0}$ instead of $\mathbf{cREG_0}(\Sigma)$.)

Unlike as this is the case for the system **REG**, the basic axioms and rules of equational logic (the reflexivity axioms and the rules SYMM, TRANS, CTXT) are neither present nor in fact derivable in $\mathbf{cREG_0}$.[7] However, it turns out that these additional axioms rules are admissible in $\mathbf{cREG_0}$ in the following sense:

[6] The main reason for this proviso is that it is important for establishing a smooth proof-theoretical relationship (stated by Lemma 4 below) between $\mathbf{cREG_0}(\Sigma)$ and its annotated version $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ defined in Section 7.

[7] In Remark 1, we comment on an extension of $\mathbf{cREG_0}$ with these axioms and rules.

if their use is limited to situations in which subderivations do not contain open assumptions, then no more theorems (than those of $\mathbf{cREG_0}$) become derivable.[8] This is an easy consequence of the fact, which is stated formally below, that the theorems of $\mathbf{cREG_0}$ are precisely the identities of regular expression equivalence.

Theorem 3. *The proof system* $\mathbf{cREG_0}(\Sigma)$ *is sound and complete with respect to regular expression equivalence; more formally, for all* $E, F \in \mathcal{R}(\Sigma)$ *it holds:*

$$\vdash_{\mathbf{cREG_0}(\Sigma)} E = F \quad \Longleftrightarrow \quad E =_L F \,. \tag{7}$$

Proof (Sketch). Let $E, F \in \mathcal{R}(\Sigma)$ be arbitrary. For the direction "$\Rightarrow$" in (7), let $\mathcal{D}$ be a derivation in $\mathbf{cREG_0}(\Sigma)$ without open assumptions and with conclusion $E = F$. Then $\left\{ \langle [\tilde{E}]_{\mathbf{ACI^+}}, [\tilde{F}]_{\mathbf{ACI^+}} \rangle \mid \tilde{E} = \tilde{F} \text{ is formula in } \mathcal{D} \right\}$ is a finite bisimulation between E and F in $\mathcal{R}(\Sigma)_{\mathbf{ACI^+}}$. Hence Theorem 2 entails $E =_L F$. For the direction "$\Leftarrow$" in (7), suppose $E =_L F$. Then, again by Theorem 2, there exists a finite bisimulation between E and F in $\mathcal{R}(\Sigma)_{\mathbf{ACI^+}}$. From such a finite bisimulation a derivation in $\mathbf{cREG_0}(\Sigma)$ without open assumptions and with conclusion $E = F$ can be extracted in a rather straightforward way.

We consider again our running example $E \equiv (a+b)^* =_L (a^*b)^*a \equiv F_1$. From the finite bisimulation given in Section 5, it is easy to extract the following derivation in $\mathbf{cREG_0}(\{a, b\})$ that does not contain open assumptions (double lines indicate multiple successive applications of $\mathrm{App}_l\mathrm{Ax}_{\mathbf{ACI^+}}$ and/or $\mathrm{App}_r\mathrm{Ax}_{\mathbf{ACI^+}}$):

$$\dfrac{\text{COMP/FIX, } d\ \dfrac{\text{COMP/FIX, } e\ \dfrac{\dfrac{(E = F_2)^e}{E_a = (F_2)_a} \quad \dfrac{(E = F_1)^d}{E_b = (F_2)_b}}{E = F_2}}{E_a = (F_1)_a} \qquad \dfrac{(E = F_1)^d}{E_b = (F_1)_b}}{E = F_1} \tag{8}$$

Remark 1. The fact that the system $\mathbf{cREG_0}$ does not contain the characteristic rules of equational logic is not absolutely necessary for showing a soundness and completeness theorem comparable to Theorem 3. In fact, for all alphabets Σ, the extension $\mathbf{cREG}(\Sigma)$ of $\mathbf{cREG_0}(\Sigma)$ by adding reflexivity axioms and the rules SYMM, TRANS, and CTXT is also sound and complete with respect to $=_L$ (but the soundness part requires a rather more involved proof, cf. the comparable situation treated in [1]). However, $\mathbf{cREG}(\Sigma)$ lacks a nice property of the system $\mathbf{cREG_0}(\Sigma)$: derivations in $\mathbf{cREG_0}(\Sigma)$ without open assumptions correspond, as reflected in the proof of Theorem 3, to finite bisimulations in $\mathcal{R}(\Sigma)_{\mathbf{ACI^+}}$; this is not the case for derivations in $\mathbf{cREG}(\Sigma)$ (due to, above all, the presence of the transitivity rule). Hence the system $\mathbf{cREG_0}$ is much more directly related to the finitary coinduction principle than the system $\mathbf{cREG}$. But there is a second

[8] Note that admissibility in this sense does not demonstrate the soundness with respect to $=_L$ of the extension of $\mathbf{cREG_0}$ with the mentioned equational axioms and rules.

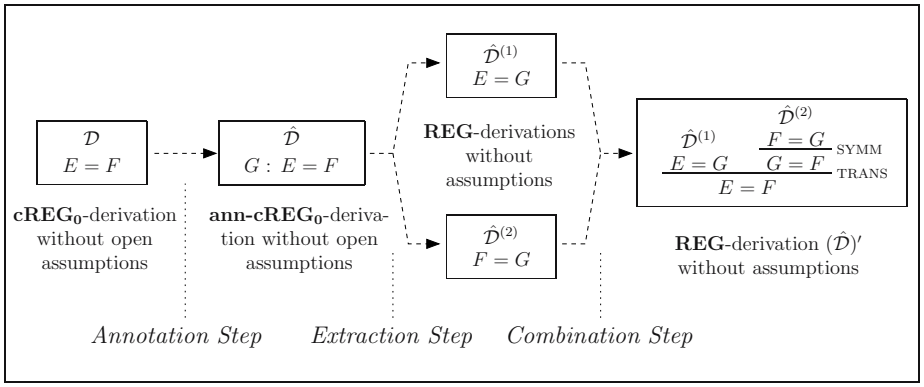

Fig. 3. Illustration of the three main steps in the transformation from an arbitrary derivation $\mathcal{D}$ in $\mathbf{cREG_0}$ without open assumptions into a derivation $(\hat{\mathcal{D}})'$ in **REG** with the same conclusion and without assumptions as $\mathcal{D}$

(although connected) reason for why we base ourselves on the system $\mathbf{cREG_0}$ here: it turns out that $\mathbf{cREG_0}$-derivations lend themselves much better to being transformed into **REG**-derivations than $\mathbf{cREG_0}$-derivations.[9]

7 A Transformation of cREG$_0$- into REG-Derivations

In this section we sketch a proof-theoretic transformation of derivations in the coinductively motivated system $\mathbf{cREG_0}(\Sigma)$ into derivations in the variant system $\mathbf{REG}(\Sigma)$ of Salomaa's axiomatisation $\mathbf{F_1}$. The three steps of this transformation are the annotation step, the extraction step, and the combination step that are illustrated together in Figure 3 and that are described below separately.

7.1 The Annotation Step

In the annotation step, a given derivation $\mathcal{D}$ in $\mathbf{cREG_0}(\Sigma)$ is "analysed" by assigning to each formula a regular expression as an annotation. In this way a derivation $\hat{\mathcal{D}}$ in an annotated version $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ of $\mathbf{cREG_0}(\Sigma)$ is built.

For an alphabet Σ and an infinite set Δ of assumption markers such that $\Sigma \cap \Delta = \emptyset$ holds, the system $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ is defined as follows: the *formulas* of $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ are expressions of the form $G : E = F$ with $E, F \in \mathcal{R}(\Sigma)$ and $G \in \mathcal{R}(\Sigma \cup \Delta)$; possible *assumptions* in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ are of the form $(d : E = F)^d$ with $E, F \in \mathcal{R}(\Sigma)$ and $d \in \Delta$; and the *rules* of $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ are the four rules $\mathrm{App}_r\mathrm{Ax}_{\mathbf{ACI}^+}$, $\mathrm{App}_l\mathrm{Ax}_{\mathbf{ACI}^+}$, COMP, and COMP/FIX that are schematically defined in Figure 4; for both of the applications of COMP and COMP/FIX shown in Figure 4, $\bigcup_{i=1}^{n} J_i = \{1, \ldots, m\}$ is

[9] A possibility for extending the transformation from $\mathbf{cREG_0}$- into **REG**-derivations that is described in the next section to a transformation from **cREG**- into **REG**-derivations consists in the use of an elimination method for basic equational rules similar to one that is developed in [5, Ch.8].

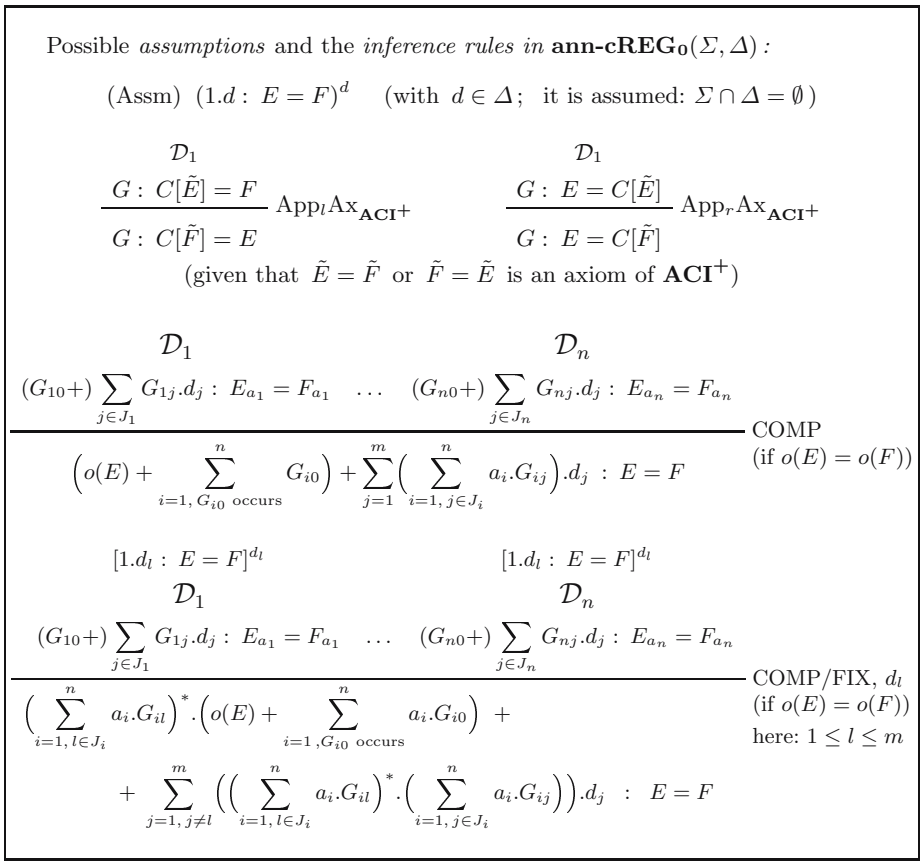

Fig. 4. The annotated version **ann-cREG$_0(\Sigma,\Delta)$** of **cREG$_0(\Sigma)$**

assumed as well as that $G_{ij}\in\mathcal{R}(\Sigma)$ holds (i.e. that the G_{ij} do not contain letters from Δ), for all $i\in\{1,\ldots,n\}$ and $j\in J_i\cup\{0\}$. (We comment on the motivation for the specific way how the annotations have been chosen for the rules of the system **ann-cREG$_0(\Sigma,\Delta)$** in Remark 2 at the end of this subsection.)

As in the system **cREG$_0(\Sigma)$**, every application of the rule COMP/FIX discharges precisely one inhabited class of open assumptions (and the marker of this assumption class is attached to the application). *Derivations* in the system **ann-cREG$_0(\Sigma,\Delta)$** are defined analogously as in **cREG$_0(\Sigma)$**, and a similar proviso on the use of assumption markers is stipulated: in assumptions distinct equations must be annotated by distinct letters from Δ, i.e. if in a derivation the assumptions $(d_1:\ E_1=F_1)^{d_1}$ and $(d_2:\ E_2=F_2)^{d_2}$ occur, then $d_1=d_2$ must entail $E_1\equiv E_2$ and $F_1\equiv F_2$.[10]

The following lemma states the basic proof-theoretic relationship between the systems **cREG$_0(\Sigma)$** and **ann-cREG$_0(\Sigma,\Delta)$**.

[10] This condition is necessary for the extraction step (i.p. for the proof of Lemma 6).

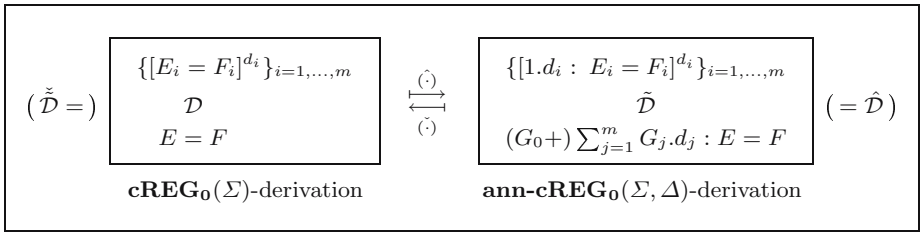

Fig. 5. Proof-theoretic relation betw. $\mathbf{cREG_0}(\Sigma)$ and $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$

Lemma 4. *Every derivation $\mathcal{D}$ in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ is of the form of the right derivation in Figure 5, for some $m \in \omega$, $E, F, G_0 \in \mathcal{R}(\Sigma)$, $E_i, F_i, G_i \in \mathcal{R}(\Sigma)$ for all $i \in \{1, \ldots, m\}$, and distinct $d_1, \ldots, d_n \in \Delta$; the expression at the top of this derivation denotes the family of all open assumption classes in $\mathcal{D}$.*

Every derivation in $\mathbf{cREG_0}(\Sigma)$ that is of the form left in Figure 5 (for some $E, F, E_1, F_1, \ldots, E_m, F_m \in \mathcal{R}(\Sigma)$, $d_1, \ldots, d_m \in \Delta$) can effectively be transformed, by assigning an appropriate annotating regular expression in $\mathcal{R}(\Sigma \cup \Delta)$ to each formula in $\mathcal{D}$, into a derivation $\hat{\mathcal{D}}$ in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ that is, for some $G_0, \ldots, G_m \in \mathcal{R}(\Sigma)$, of the form of the derivation $\tilde{\mathcal{D}}$ on the right in Figure 5.

And vice versa*, every derivation $\tilde{\mathcal{D}}$ in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ that is of the form on the right in Figure 5 can be transformed, by stripping annotated formulas in $\tilde{\mathcal{D}}$ of the annotating regular expressions, into a derivation $\overset{\circ}{\tilde{\mathcal{D}}}$ that is of the form of the left derivation in Figure 5.*

Proof (Hint). All three statements of the lemma can be shown by straightforward induction on the structure (or the depth) of derivations in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$, and respectively, by induction on the structure of derivations in $\mathbf{cREG_0}(\Sigma)$.

It is easy to verify that the result of annotating the $\mathbf{cREG_0}(\{a, b\})$-derivation $\mathcal{D}$ in (8) for our running example is the following $\mathbf{ann\text{-}cREG_0}(\{a, b\}, \Delta)$-derivation $\hat{\mathcal{D}}$ without open assumptions (a number of annotations appear simplified):

$$\dfrac{\dfrac{\dfrac{\dfrac{(1.e :\ E = F_2)^e}{1.e :\ E_a = (F_2)_a} \quad \dfrac{(1.d :\ E = F_1)^d}{1.d :\ E_b = (F_2)_b}}{a^* + a^*b.d :\ E = F_2}\ \text{COMP/FIX}, e \quad \dfrac{(1.d :\ E = F_1)^d}{1.d :\ E_b = (F_1)_b}}{a^* + a^*b.d :\ E_a = (F_1)_a}\ \text{COMP/FIX}, d}{(aa^*b + b)^*(1 + aa^*) :\ E = F_1} \tag{9}$$

Remark 2. Informally, the principal idea underlying the system $\mathbf{ann\text{-}cREG_0}$ and its relation with $\mathbf{cREG_0}$ is the following: annotating a $\mathbf{cREG_0}(\Sigma)$-derivation $\mathcal{D}$ with conclusion $E = F$ and *without* open assumptions into a derivation $\hat{\mathcal{D}}$ in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ with conclusion $G :\ E = F$ amounts to extracting from $\mathcal{D}$ a description as the regular expression G of the bisimulation between $[E]_{\mathbf{ACI^+}}$ and $[F]_{\mathbf{ACI^+}}$ in the automaton $\mathcal{R}(\Sigma)_{\mathbf{ACI^+}}$ that is formalised by $\mathcal{D}$ (cf. the proof of Theorem 3). For this regular expression $G \in \mathcal{R}(\Sigma)$, $[G]_{\mathbf{ACI^+}}$ is bisimilar in $\mathcal{R}(\Sigma)_{\mathbf{ACI^+}}$ to both $[E]_{\mathbf{ACI^+}}$ and $[F]_{\mathbf{ACI^+}}$; moreover, the "generated

subautomaton" of $[G]_{\mathbf{ACI}^+}$ in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$ is a "common unfolding" of the subautomata in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$ that are generated by $[E]_{\mathbf{ACI}^+}$ and $[F]_{\mathbf{ACI}^+}$, respectively. These facts form the deeper reasons for why the extraction step (described in Subsection 7.2) of our transformation from $\mathbf{cREG_0}(\Sigma)$ to $\mathbf{REG}(\Sigma)$ is possible.

However, in the conclusion $G : E = F$ of an $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$-derivation $\tilde{\mathcal{D}}$ *with* open assumptions, the annotation $G \in \mathcal{R}(\Sigma \cup \Delta)$ only describes what could be called a "partial bisimulation" in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$ between $[E]_{\mathbf{ACI}^+}$ and $[F]_{\mathbf{ACI}^+}$. But nevertheless, and slightly apart from this, the annotation G in the conclusion of such a derivation $\tilde{\mathcal{D}}$ also specifies the common structure of a pair of "valid" equations that link the regular expressions on either side of "=" in the conclusion of $\tilde{\mathcal{D}}$ with the regular expressions on respectively the same side of "=" in the open assumptions of $\tilde{\mathcal{D}}$. More precisely, if $\tilde{\mathcal{D}}$ is a derivation in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ of the form

$$\begin{array}{c} \{[1.d_i : \; E_i = F_i]^{d_i}\}_{i=1,\ldots,m} \\ \tilde{\mathcal{D}} \\ (G_0+) \sum_{j=1}^{m} G_j.d_j : E = F \end{array} \tag{10}$$

for some $m \in \omega$, $E, F, E_1, F_1, \ldots, E_m, F_m, G_0, \ldots, G_m \in \mathcal{R}(\Sigma)$, $d_1, \ldots, d_m \in \Delta$, and with the expression at the top denoting the family of all inhabited open assumptions classes of $\tilde{\mathcal{D}}$ (due to Lemma 4 all $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$-derivations can be represented in this way), then the equations $E = (G_0+) \sum_{j=1}^{m} G_j.E_j$ and $F = (G_0+) \sum_{j=1}^{m} G_j.F_j$ are valid with respect to $=_L$, i.e. it holds:

$$E =_L (G_0+) \textstyle\sum_{j=1}^{m} G_j.E_j \qquad \text{and} \qquad F =_L (G_0+) \textstyle\sum_{j=1}^{m} G_j.F_j \; . \tag{11}$$

This property of $\mathbf{ann\text{-}cREG_0}$-derivations is essential for the extraction step. The annotations in the rules of $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ have been chosen accordingly for this purpose, utilising the fundamental relation between regular expressions and their single-letter derivatives as formulated in Lemma 5 below.

7.2 The Extraction Step

In the extraction step, from a given derivation $\tilde{\mathcal{D}}$ in $\mathbf{ann\text{-}cREG_0}$ with conclusion $G : E = F$ two derivations $\tilde{\mathcal{D}}^{(1)}$ and $\tilde{\mathcal{D}}^{(2)}$ are constructed that, in case that $\tilde{\mathcal{D}}$ does not contain open assumptions, demonstrate respectively that E and F are equivalent with the annotating regular expression G. This is justified by Lemma 6 below; the proof of this lemma depends on Lemma 5, a version appropriate for regular expressions of the sometimes so called "fundamental theorem of formal languages" (due to the analogy with the "fundamental theorem of calculus").

Lemma 5. *For all* $E \in \mathcal{R}(\Sigma)$, $E \equiv_{\mathbf{REG}^-} o(E) + \sum_{i=1}^{n} a_i.E_{a_i}$ *holds. What is more, for every given* $E \in \mathcal{R}(\Sigma)$, *a derivation* $\mathcal{D}^{(E)}$ *in* $\mathbf{REG}^-(\Sigma)$ *with conclusion* $E = o(E) + \sum_{i=1}^{n} a_i.E_{a_i}$ *can effectively be constructed.*

Proof (Hint). The lemma can be shown by induction on the syntactical structure of regular expressions in $\mathcal{R}(\Sigma)$. For the treatment of the case $E = F^*$ in the induction step (for $E, F \in \mathcal{R}(\Sigma)$) the axioms (B10), (B11) of $\mathbf{REG}^-$ are needed.

Lemma 6. *From every derivation $\tilde{\mathcal{D}}$ in* $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ *of the form* (10)*, where $m \in \omega$, $d_1, \ldots, d_m \in \Delta$ distinct markers, $E, F, G_0 \in \mathcal{R}(\Sigma)$, and, for all $i \in \{1, \ldots, m\}$, $E_i, F_i, G_i \in \mathcal{R}(\Sigma)$, it is possible to construct effectively derivations in* $\mathbf{REG}(\Sigma)$ *of the respective forms*

$$\frac{\tilde{\mathcal{D}}^{(1)}}{E = (G_0 +) \sum_{j=1}^{m} G_j.E_j} \quad \text{and} \quad \frac{\tilde{\mathcal{D}}^{(2)}}{F = (G_0 +) \sum_{j=1}^{m} G_j.F_j} \tag{12}$$

Proof (Hint). The lemma can be demonstrated by defining an effective extraction procedure of the two derivations $\tilde{\mathcal{D}}^{(1)}$ and $\tilde{\mathcal{D}}^{(2)}$ in $\mathbf{REG}(\Sigma)$ with the respective forms in (12) from an arbitrary derivation $\tilde{\mathcal{D}}$ in $\mathbf{ann\text{-}cREG_0}(\Sigma, \Delta)$ of the form (10), where $m \in \omega$, $E, F, G_0 \in \mathcal{R}(\Sigma)$, $E_i, F_i, G_i \in \mathcal{R}(\Sigma)$ for all $i \in \{1, \ldots, m\}$, and $d_1, \ldots, d_m \in \Delta$ are distinct. Such a procedure can be built by using induction on the structure (or on the depth) of the derivation $\tilde{\mathcal{D}}$.

Let us demonstrate the induction step for the extraction of the derivation $\hat{\mathcal{D}}^{(1)}$ from the annotated derivation $\hat{\mathcal{D}}$ in (9) relating to our running example. $\hat{\mathcal{D}}^{(1)}$ can be written as of the form

$$\frac{\begin{array}{cc}[1.d:\ E = F_1]^d & [1.d:\ E = F_1]^d \\ \hat{\mathcal{D}}_1 & \hat{\mathcal{D}}_2 \\ a^* + a^*b.d:\ E_a = (F_1)_a & 1.d:\ E_b = (F_1)_b\end{array}}{(aa^*b + b)^*(1 + aa^*):\ E = F_1}\ \text{COMP/FIX}$$

with $\hat{\mathcal{D}}_1$ and $\hat{\mathcal{D}}_2$ being the immediate left and right subderivations of $\hat{\mathcal{D}}$; to increase readability in this example, we suppress some "."-signs and brackets. We want to construct a derivation $\hat{\mathcal{D}}^{(1)}$ in $\mathbf{REG}(\{a, b\})$ with conclusion $E = (aa^*b + b)^*(1 + aa^*)$. By the induction hypothesis there exist derivations $\hat{\mathcal{D}}_1^{(1)}$ and $\hat{\mathcal{D}}_2^{(1)}$ in $\mathbf{REG}(\{a, b\})$ with the conclusions $E_a = a^* + a^*b.E$ and $E_b = E$. By temporarily using additional rules, from $\hat{\mathcal{D}}_1^{(1)}$ and $\hat{\mathcal{D}}_2^{(1)}$ the derivation $\hat{\mathcal{D}}_{0,\mathrm{ar}}^{(1)}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\hat{\mathcal{D}}_1^{(1)}}{E_a = a^* + a^*b.E}}{a.E_a = a.(a^* + a^*b.E)}\ \text{CTXT} \quad \dfrac{\hat{\mathcal{D}}_2^{(1)}}{E_b = E}}{b.E_b = b.E}\ \text{CTXT}}{a.E_a + b.E_b = a.(a^* + a^*b.E) + b.E}\ +}{1 + a.E_a + b.E_b = 1 + a.(a^* + a^*b.E) + b.E}\ \text{CTXT}}{1 + a.E_a + b.E_b = (aa^*b + b).E + (1 + aa^*)}\ \text{App}_r\text{Ax}_{\mathbf{ACI}+}$$

can be constructed. This derivation can be extended, by using the fixed-point rule FIX in $\mathbf{REG}(\{a, b\})$ in an essential way, into the derivation $\hat{\mathcal{D}}_{\mathrm{ar}}^{(1)}$

$$\dfrac{\dfrac{\begin{array}{cc}\mathcal{D}_{\mathrm{ar}}^{(E)} & \hat{\mathcal{D}}_{0,\mathrm{ar}}^{(1)} \\ E = o(E) + a.E_a + b.E_b & 1 + a.E_a + b.E_b = (aa^*b + b).E + (1 + aa^*)\end{array}}{E = (aa^*b + b).E + (1 + aa^*)}\ \text{TRANS}}{E = (aa^*b + b)^*(1 + aa^*)}\ \text{FIX}$$

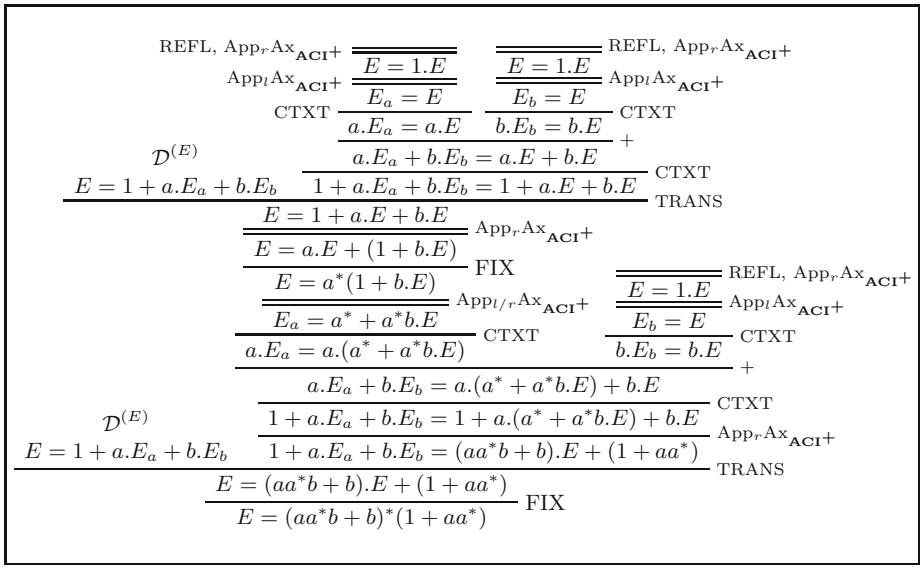

Fig. 6. Abbreviated result $\hat{\mathcal{D}}^{(1)}_{\mathrm{ar}}$ of extracting the $\mathbf{REG}(\{a,b\})$-derivation $\hat{\mathcal{D}}^{(1)}$ from the **ann-cREG$_0$**-deriv. $\hat{\mathcal{D}}$ in (9) (some **REG**-derivable rules are used)

where the derivation $\mathcal{D}^{(E)}_{\mathrm{ar}}$ is guaranteed by Lemma 5 and can be chosen as

$$\dfrac{\dfrac{\dfrac{}{(a+b)^* = 1 + (a+b)(a+b)^*}\,\text{(B10)}}{(a+b)^* = 1 + a(a+b)^* + b(a+b)^*}\,\text{App}_l\text{Ax}_{\mathbf{ACI}+}}{E = 1 + a\underbrace{(1+0)(a+b)^*}_{E_a} + b\underbrace{(0+1)(a+b)^*}_{E_b}}\,\text{App}_l\text{Ax}_{\mathbf{ACI}+}$$

The desired derivation $\hat{\mathcal{D}}^{(1)}$ in $\mathbf{REG}(\{a,b\})$ can then be found as the result of eliminating from $\hat{\mathcal{D}}^{(1)}_{\mathrm{ar}}$ all applications of the additional rules "+", App$_l$Ax$_{\mathbf{ACI}+}$, and App$_r$Ax$_{\mathbf{ACI}+}$, which can easily be recognised to be derivable in $\mathbf{REG}(\{a,b\})$.

The result of the entire extraction process of $\hat{\mathcal{D}}^{(1)}$ from $\hat{\mathcal{D}}$ is displayed in Figure 6 as the derivation $\hat{\mathcal{D}}^{(1)}_{\mathrm{ar}}$ in which applications of additional rules occur and the derivation $\mathcal{D}^{(E)}$ is abbreviated. In an analogous way, also the derivation $\hat{\mathcal{D}}^{(2)}$ in $\mathbf{REG}(\{a,b\})$ with conclusion $F = (aa^*b+b)^*(1+aa^*)$ can be extracted from $\hat{\mathcal{D}}$; similar to $\hat{\mathcal{D}}^{(1)}_{\mathrm{ar}}$, it is given as the abbreviated derivation $\hat{\mathcal{D}}^{(2)}_{\mathrm{ar}}$ in Figure 7.

7.3 The Combination Step

The last step of the transformation is easy and consists in combining the two $\mathbf{REG}(\Sigma)$-derivations $\hat{\mathcal{D}}^{(1)}$ and $\hat{\mathcal{D}}^{(2)}$, which were extracted from the annotated version $\hat{\mathcal{D}}$ of a **cREG$_0$**-derivation $\mathcal{D}$ on which the transformation was started. Building from $\hat{\mathcal{D}}^{(1)}$ and $\hat{\mathcal{D}}^{(2)}$ a $\mathbf{REG}(\Sigma)$-derivation $(\hat{\mathcal{D}})'$ with the same conclusion as $\mathcal{D}$ only requires the use of each an application of SYMM and TRANS, as

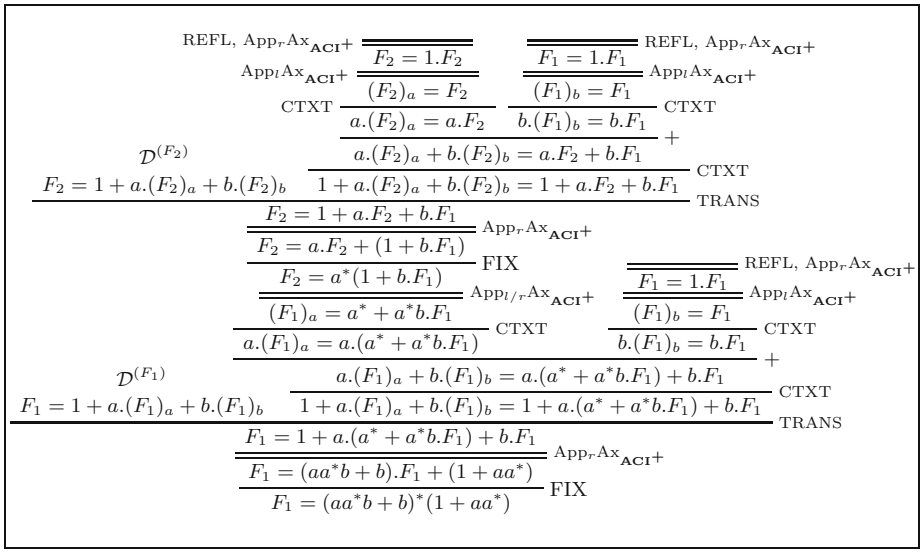

Fig. 7. Abbreviated result $\hat{\mathcal{D}}^{(2)}_{\mathrm{ar}}$ of extracting the $\mathbf{REG}(\{a,b\})$-derivation $\hat{\mathcal{D}}^{(2)}$ from the **ann-cREG$_\mathbf{0}$**-deriv. $\hat{\mathcal{D}}$ in (9) (some **REG**-derivable rules are used)

illustrated in Figure 3. Due to this, Lemma 4 and Lemma 6 together yield the transformation theorem below. In view of Theorem 3, this theorem facilitates an alternative completeness proof for **REG** and therefore entails the subsequent corollary, which is a restatement of the completeness part of Theorem 1.

Theorem 4. *Every derivation $\mathcal{D}$ in $\mathbf{cREG_0}(\Sigma)$ without open assumptions can effectively be transformed into a derivation $\mathcal{D}'$ in $\mathbf{REG}(\Sigma)$ with the same conclusion as $\mathcal{D}$.*

Corollary 2. $\mathbf{REG}(\Sigma)$ *is complete with respect to* $=_L$.

8 Conclusion

Using a coinduction principle for language equality given by Rutten in [6], we introduced a "finitary" coinduction principle for proving equivalence of regular expressions: for showing that two regular expressions E and F are equivalent, prove that, up to applying laws including associativity, commutativity and idempotency of $+$, E and F are bisimilar in an automaton of regular expressions whose transition function is based on the "Brzozowski derivative". We recognised that this principle can be used to decide regular expression equivalence in an effective way, and hence, that it can be implemented in principle (further considerations lead us to the belief that this is indeed a practical possibility).

Subsequently we introduced a proof system **cREG$_\mathbf{0}$** of equational logic that formalises proofs using the finitary coinduction principle as finite derivations:

its soundness and completeness proof directly reflects the fact that a derivation in $\mathbf{cREG_0}$ (without open assumptions) corresponds to a finite bisimulation between the regular expressions in its conclusion. Finally, we showed that derivations in $\mathbf{cREG_0}$ can be transformed into derivations in a variant system **REG** of Salomaa's axiomatisation $\mathbf{F_1}$ in a very straightforward and "natural" way. Hereby we obtained a coinductive completeness proof for the system **REG**.

Our constructions, and in particular the transformation we sketched, can be adapted to yield also a coinductive completeness proof for Salomaa's $\mathbf{F_1}$. This is because an alternative differential calculus for formal languages and regular expressions can be introduced, in which derivatives take away letters from the end of words: one can define, for letters a, the language derivative $(\cdot)'_a : \mathcal{L}(\Sigma) \to \mathcal{L}(\Sigma)$ by $L \mapsto (L)'_a =_{\text{def}} \{\, v \mid v.a \in L \,\}$. Based on corresponding versions of derivatives for regular expressions, one can formulate an effective finitary coinduction principle analogous to Theorem 2, a sound and complete proof system $\mathbf{cREG'_0}$ for $=_L$ analogous to $\mathbf{cREG_0}$, and an effective transformation of $\mathbf{cREG'_0}$-derivations into $\mathbf{F_1}$-derivations analogous to the one described here. An effective completeness proof for $\mathbf{F_1}$ can directly be based on these elements.

Acknowledgement. The proof-transformation described in this paper was conceived after a talk by Jan Rutten in which $1 + a(a+b)^* + (a+b)^*aa(a+b)^* =_L =_L ((b^*a)^*ab)^*$ was shown by coinduction and the "homework" was assigned of giving an alternative proof by a deduction in Salomaa's axiomatisation of $=_L$. Also, I want to thank Jan Rutten for calling it to my attention that Lemma 2 already holds w.r.t. **ACI** (and not only w.r.t. $\mathbf{REG}^-$ as I had used previously). Furthermore, I would like to convey my thanks to the anonymous referees for their remarks, observations, questions, and stimulating comments. Last, but not least thanks are due to Mihály Petreczky for a couple of useful discussions, and to Helle Hansen for suggesting a formulation concerning the name of Lemma 5.

References

1. Brandt, M., Henglein, F.: "Coinductive axiomatization of recursive type equality and subtyping", *Fundamenta Informaticae* **33** (1998) 1–30.
2. Brzozowski, J.A.: "Derivatives of regular expressions", *Journal of the ACM* **11** (1964) 481–494.
3. Conway, J.H.: *Regular Algebra and Finite Machines*, Chapman and Hall (1971).
4. Hüttel, H., Stirling, C.: "Actions Speak Louder Than Words: Proving Bisimilarity for Context-Free Processes", *Journ. of Logic and Computation* **8**:4 (1998) 485–509.
5. Grabmayer, C.: *Relating Proof Systems for Recursive Types*, PhD thesis, Vrije Universiteit Amsterdam (2005) `http://www.cs.vu.nl/~clemens/proefschrift.pdf` .
6. Rutten, J.J.M.M.: "Automata and Coinduction (an Exercise in Coinduction)", *Proceedings of CONCUR '98*, LNCS 1466, Springer (1998) 194–218.
7. Salomaa, A.: "Two complete axiom systems for the algebra of regular events", *Journal of the ACM* **13**:1 (1966) 158–169.
8. Troelstra, A.S., Schwichtenberg, H.: *Basic Proof Theory*, Cambridge University Press (1996, 2000).