Using Proofs by Coinduction to Find “Traditional” Proofs

Clemens Grabmayer

Dept. of Computer Science
Free University (VU) Amsterdam

CALCO 2005
5^{th} of September 2005

Overview

Stepping Stones and Contributions

- A finitary coinduction principle for regular expression equivalence.
- A coinductively motivated proof system $\mathbf{cREG_0}$ for reg.expr.equiv.
- An effective proof-theoretic transformation from the coinductive system $\mathbf{cREG_0}$ to the "traditional" system **REG**.

Used Concepts

Overview

Stepping Stones and Contributions

- A finitary coinduction principle for regular expression equivalence.
- A coinductively motivated proof system $\mathbf{cREG_0}$ for reg.expr.equiv.
- An effective proof-theoretic transformation from the coinductive system $\mathbf{cREG_0}$ to the "traditional" system **REG**.

Used Concepts

– Regular expression equivalence.
– Deterministic automata.
– Language derivatives. A coinduction principle for lang. equality.
– Brzozowski derivatives. A coinduction principle for reg.expr.equiv.
– Salomaa's axiomatisation $\mathbf{F_1}$ and its "reverse form" **REG**.

Regular Expression Equivalence

For $\Sigma = \{a_1, \ldots, a_n\}$. $\mathcal{L}(\Sigma)$: the set of *formal languages over* Σ;
$\mathcal{R}(\Sigma)$, the set of *regular expressions* over alphabet Σ:

$$E \; ::= \; 0 \mid 1 \mid a_1 \mid \ldots \mid a_n \mid E+E \mid E.E \mid E^*$$

$=_L$, regular expression equivalence is defined by

$$E =_L F \quad \Longleftrightarrow_{\text{def}} \quad L(E) = L(F)$$

($L(E)$, $L(F)$ are the *languages represented by* E, F).

Example: $(a+b)^* =_L (a^*.b)^*.a^*$.

Regular Expression Equivalence

For $\Sigma = \{a_1, \ldots, a_n\}$. $\mathcal{L}(\Sigma)$: the set of *formal languages over* Σ;
$\mathcal{R}(\Sigma)$, the set of *regular expressions* over alphabet Σ:

$$E \;::=\; 0 \mid 1 \mid a_1 \mid \ldots \mid a_n \mid E+E \mid E.E \mid E^*$$

$=_L$, regular expression equivalence is defined by

$$E =_L F \quad \Longleftrightarrow_{\text{def}} \quad L(E) = L(F)$$

($L(E)$, $L(F)$ are the *languages represented by* E, F).

Example: $(a+b)^* =_L (a^*.b)^*.a^*$.

We say: $E \in \mathcal{R}(\Sigma)$ has the
empty word property $(ewp(E)) \quad \Longleftrightarrow_{\text{def}} \quad \epsilon \in L(E)$.

The Axiom System REG for $=_L$ (Salomaa’s axiomatisation $\mathbf{F}_1$ reversed)

The *axioms* of **REG**:

(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 **REG**: REFL, SYMM, TRANS, and

$$\frac{E = F}{C[E] = C[F]}\ \text{CTXT} \qquad\qquad \frac{E = F.E + G}{E = F^*.G}\ \text{FIX (if } \neg\, ewp(F))$$

The Axiom System REG (Cont.)

Theorem ($\sim$ Salomaa, Aanderaa (1965/66)). *The axiom system* **REG** *is sound and complete with respect to* $=_L$ *:*

$$(\textit{for all } E, F \in \mathcal{R}(\Sigma)) \left[\;\vdash_{\textsf{REG}} E = F \quad \Longleftrightarrow \quad E =_L F \;\right].$$

The Axiom System REG (Cont.)

Theorem ($\sim$ Salomaa, Aanderaa (1965/66)). *The axiom system* **REG** *is sound and complete with respect to* $=_L$ *:*

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

Sub-Axiom-Systems without FIX that will be used:

ACI: the *axioms* for associativity, commutativity, idempotency;

ACI$^+$: all *axioms* not involving $^* + \{ 1.E = E \} + \{ 0.E = 0 \}$.

$\equiv_{\textbf{ACI}}$, $\equiv_{\textbf{ACI}^+}$: relations of provable equality in **ACI** and **ACI$^+$**.

$\mathcal{R}(\Sigma)_{\textbf{ACI}}$, $\mathcal{R}(\Sigma)_{\textbf{ACI}^+}$: $\equiv_{\textbf{ACI}}$- and $\equiv_{\textbf{ACI}^+}$-equivalence classes of $\mathcal{R}(\Sigma)$.

A Coinductively Motivated Proof System $\mathbf{cREG_0}$

The possible *marked assumptions* in $\mathbf{cREG_0}$:

$$\text{(Assm)}\quad (E = F)^{u}$$

The *inference rules* of $\mathbf{cREG_0}$: (Given $\Sigma = \{a_1, \ldots, a_n\}$).

$$\frac{\begin{matrix}\mathcal{D}_1\\ C[E_1] = F\end{matrix}}{C[E_2] = F}\,\mathsf{App}_l\mathsf{Ax}_{\mathbf{ACI}^+} \qquad \frac{\begin{matrix}\mathcal{D}_1\\ F = C[E_1]\end{matrix}}{F = C[E_2]}\,\mathsf{App}_r\mathsf{Ax}_{\mathbf{ACI}^+}$$

($E_1 = E_2$ or $E_2 = E_1$ is an $\mathbf{ACI}^+$-axiom),

$$\frac{\begin{matrix}\mathcal{D}_1\\ E_{a_1} = F_{a_1}\end{matrix} \qquad \ldots \qquad \begin{matrix}\mathcal{D}_n\\ E_{a_n} = F_{a_n}\end{matrix}}{E = F}\,\text{COMP}$$

(if $ewp(E) \Leftrightarrow ewp(F)$)

A Coinductively Motivated Proof System $\mathbf{cREG_0}$

The possible *marked assumptions* in $\mathbf{cREG_0}$:

$$\text{(Assm)}\quad (E = F)^{u}$$

The *inference rules* of $\mathbf{cREG_0}$: (Given $\Sigma = \{a_1, \ldots, a_n\}$).

$$\dfrac{\begin{array}{c}\mathcal{D}_1\\ C[E_1] = F\end{array}}{C[E_2] = F}\,\text{App}_l\text{Ax}_{\mathbf{ACI}^+} \qquad \dfrac{\begin{array}{c}\mathcal{D}_1\\ F = C[E_1]\end{array}}{F = C[E_2]}\,\text{App}_r\text{Ax}_{\mathbf{ACI}^+}$$

($E_1 = E_2$ or $E_2 = E_1$ is an $\mathbf{ACI}^+$-axiom),

$$\dfrac{\begin{array}{c}[E = F]^{u}\\ \mathcal{D}_1\\ E_{a_1} = F_{a_1}\end{array} \qquad \ldots \qquad \begin{array}{c}[E = F]^{u}\\ \mathcal{D}_n\\ E_{a_n} = F_{a_n}\end{array}}{E = F}\ \text{COMP/FIX}, u$$

(if $ewp(E) \Leftrightarrow ewp(F)$)

The Proof System $\mathbf{cREG_0}$ (Cont.)

- does not possess SYMM and TRANS: these rules are "admissible";
- has an extension **cREG** with SYMM and TRANS that is similar to
  - the coinductive axiomatisation of *recursive type equality* by Brandt and Henglein (1998).
  - an axiomatisation of *bisimilarity of normed recursive BPA-processes* due to Stirling (1994);
- is "normalised": it fulfills a "subformula property";
- is sound and complete w.r.t. $=_L$ (proof sketched later).

Deterministic Automata

A *deterministic automaton* $\mathcal{S} = \langle S, A, o, t \rangle$ consists of

– a set S of *states* (may be infinite),

– an *input alphabet* A (may be infinite),

– an *output function* $o : S \to \{0, 1\}$,

– a *transition function* $t : S \to S^A$.

(No initial state is specified.)

Notation. For states s and s',

$s \sim s'$ means: s and s' are bisimilar;

$s \sim_{\text{fin}} s'$ means: s and s' are related by a *finite bisimulation*.

Differential Calculus for Formal Languages

Using language derivatives

$$L_a =_{\text{def}} \left\{ v \in \Sigma^* \mid a.v \in L \right\} ,$$

$\mathcal{L}(\Sigma)$ can be turned into the automaton $\mathcal{L}(\Sigma) = \langle \mathcal{L}(\Sigma), \Sigma, o_{\mathcal{L}}, t_{\mathcal{L}} \rangle$ by

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

Differential Calculus for Formal Languages

Using language derivatives

$$L_a =_{\text{def}} \{ v \in \Sigma^* \mid a.v \in L \} \ ,$$

$\mathcal{L}(\Sigma)$ can be turned into the automaton $\mathcal{L}(\Sigma) = \langle \mathcal{L}(\Sigma), \Sigma, o_{\mathcal{L}}, t_{\mathcal{L}} \rangle$ by

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

Theorem (Rutten). *For all* $L_1, L_2 \in \mathcal{L}(\Sigma)$*:*

$$L_1 \sim L_2 \ \ \textit{in}\ \mathcal{L}(\Sigma) \quad \Longrightarrow \quad L_1 = L_2 \ .$$

This justifies a *coinduction principle for proving equality of formal languages*.

Differential Calculus for Regular Expressions

Brzozowski derivatives $(\cdot)_a : \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)$ defined by clauses like

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

mimic language derivatives. And $o_{\mathcal{L}}(\cdot)$ can be mimicked by a function $o : \mathcal{R}(\Sigma) \to \{0, 1\}$.

Proposition. $L(E_a) = (L(E))_a , \quad o(E) = o_{\mathcal{L}}(L(E))$.

Automaton $\mathcal{R}(\Sigma) =_{\mathsf{def}} \langle \mathcal{R}(\Sigma), \Sigma, t, o \rangle$: letting $t(E)(a) =_{\mathsf{def}} E_a$.

Differential Calculus for Regular Expressions

Brzozowski derivatives $(\cdot)_a : \mathcal{R}(\Sigma) \rightarrow \mathcal{R}(\Sigma)$ defined by clauses like

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

mimic language derivatives. And $o_{\mathcal{L}}(\cdot)$ can be mimicked by a function $o : \mathcal{R}(\Sigma) \rightarrow \{0,1\}$.

Proposition. $L(E_a) = (L(E))_a , \quad o(E) = o_{\mathcal{L}}(L(E))$.

Automaton $\mathcal{R}(\Sigma) =_{\text{def}} \langle \mathcal{R}(\Sigma), \Sigma, t, o \rangle$: letting $t(E)(a) =_{\text{def}} E_a$.

Theorem (Rutten). *For all* $E_1, E_2 \in \mathcal{R}(\Sigma)$*:*

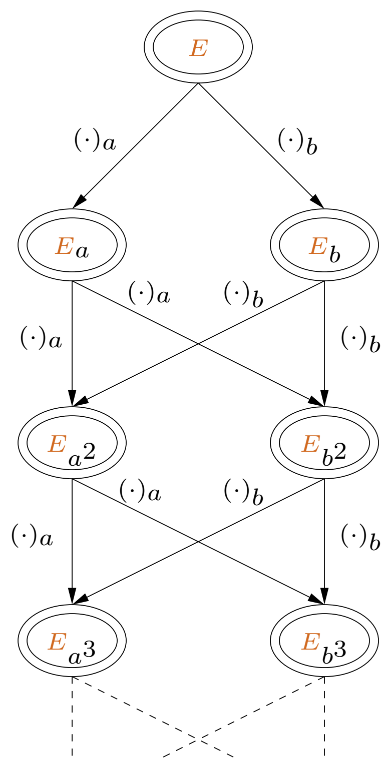
$$E_1 \sim E_2 \ \ \textit{in}\ \mathcal{R}(\Sigma) \quad \Longrightarrow \quad E_1 =_L E_2 \ .$$

(This justifies a *coinduction principle for proving equiv. of reg. expr's*.)

Naive Use of the Coinduction Principle for $=_L$ . . .

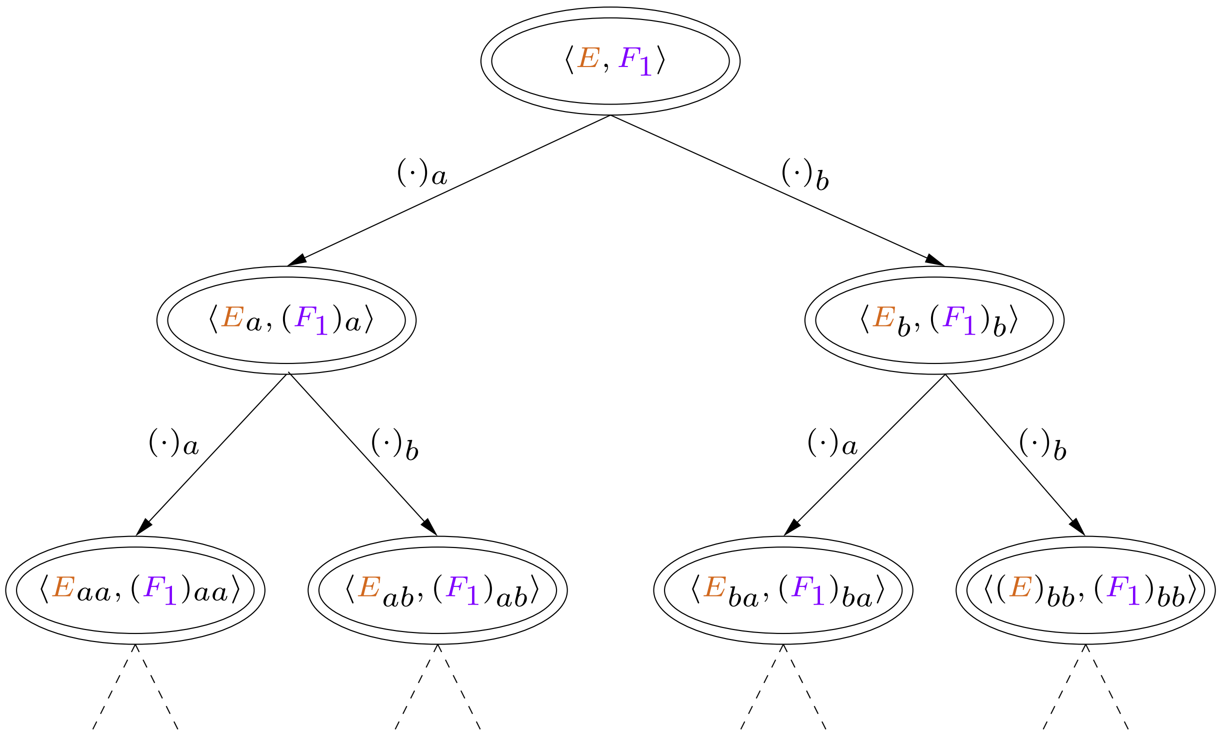
We want to show $E \equiv (a+b)^* =_L (a^*.b)^*.a^* \equiv F_1$ (justifying a simple instance of Conway's axiom scheme (SUMSTAR)).

The subautomaton of E in $\mathcal{R}(\{a,b\})$ is infinite:



. . . is not effective

Therefore a bisimulation between E and F_1 in $\mathcal{R}(\Sigma)$ that starts as



cannot be finite. Hence, used naively, the coinduction principle is not effective (not *realisable*).

Using Identities to Get Less Derivatives

$E \equiv (a+b)^*$ has infinitely many (iter.) derivatives: f.a. $w \in \{a,b\}^*$

$$E_{wa} \equiv \underbrace{(0+0).E \; + \; (\ldots \; + \; ((0+0).E}_{|w| \text{ times}} + (1+0).E))$$

$$E_{wb} \equiv \qquad\qquad \ldots + \; (0+1).E))$$

Not so if simplifying by **ACI**-identities is allowed:

$$E_{wa} \equiv_{\mathbf{ACI}} 0.E + (1+0).E \qquad\qquad E_{wb} \equiv_{\mathbf{ACI}} 0.E + (0+1).E$$

nor if simplifying by **ACI**$^+$-identities is allowed:

$$E_{wa} \equiv_{\mathbf{ACI}^+} E_{wb} \equiv_{\mathbf{ACI}^+} E \; .$$

A Finitary Coinduction Principle for $=_L$

Lemma ($\sim$ Brzozowski). *The set* $\left\{ [E_w]_{\mathsf{ACI}} \,\middle|\, w \in \Sigma^* \right\}$ *is finite f.a.* $E \in \mathcal{R}(\Sigma)$*. Hence also* $\left\{ [E_w]_{\mathsf{ACI}^+} \,\middle|\, w \in \Sigma^* \right\}$ *is finite, f.a.* $E \in \mathcal{R}(\Sigma)$*.*

Factor automaton $\mathcal{R}(\Sigma)_{\mathsf{ACI}^+} = \langle \mathcal{R}(\Sigma)_{\mathsf{ACI}^+}, o_{\mathsf{ACI}^+}, t_{\mathsf{ACI}^+} \rangle$:

letting $t_{\mathsf{ACI}^+}([E]_{\mathsf{ACI}^+})(a) =_{\mathsf{def}} [E_a]_{\mathsf{ACI}^+}$.

A Finitary Coinduction Principle for $=_L$

Lemma ($\sim$ Brzozowski). *The set* $\left\{[E_w]_{\mathsf{ACI}} \;\middle|\; w \in \Sigma^*\right\}$ *is finite f.a.* $E \in \mathcal{R}(\Sigma)$. *Hence also* $\left\{[E_w]_{\mathsf{ACI}^+} \;\middle|\; w \in \Sigma^*\right\}$ *is finite, f.a.* $E \in \mathcal{R}(\Sigma)$.

Factor automaton $\mathcal{R}(\Sigma)_{\mathsf{ACI}^+} = \langle \mathcal{R}(\Sigma)_{\mathsf{ACI}^+}, o_{\mathsf{ACI}^+}, t_{\mathsf{ACI}^+} \rangle$:

letting $t_{\mathsf{ACI}^+}([E]_{\mathsf{ACI}^+})(a) =_{\mathrm{def}} [E_a]_{\mathsf{ACI}^+}$.

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

$$[E]_{\mathsf{ACI}^+} \sim_{\mathit{fin}} [F]_{\mathsf{ACI}^+} \;\text{ in } \mathcal{R}(\Sigma)_{\mathsf{ACI}^+} \quad \Longleftrightarrow \quad E =_L F \;.$$

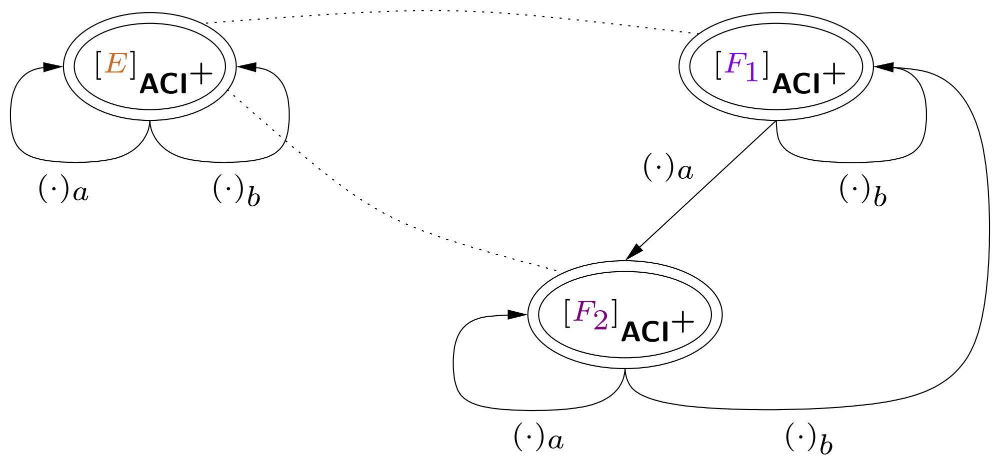
This justifies a *finitary coinduction principle* for proving equality of regular expressions.

Corollary. $=_L$ *on* $\mathcal{R}(\Sigma)$ *can be decided by checking for the existence of finite bisimulations in* $\mathcal{R}(\Sigma)_{\mathsf{ACI}^+}$.

Finitary Coinduction Principle: an Example

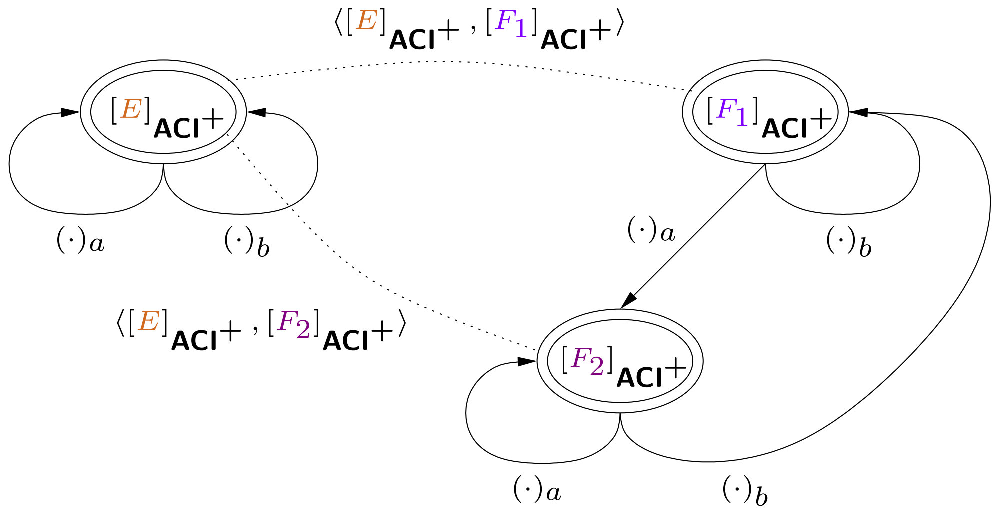
Again, we aim to prove $(a+b)^* =_L (a^*.b)^*.a^*$.

For $E \equiv (a+b)^*$, $F_1 \equiv (a^*.b)^*.a^*$, and $F_2 \equiv ((a^*.b).(a^*.b)^*).a^* + a^*$ it is easy to verify:

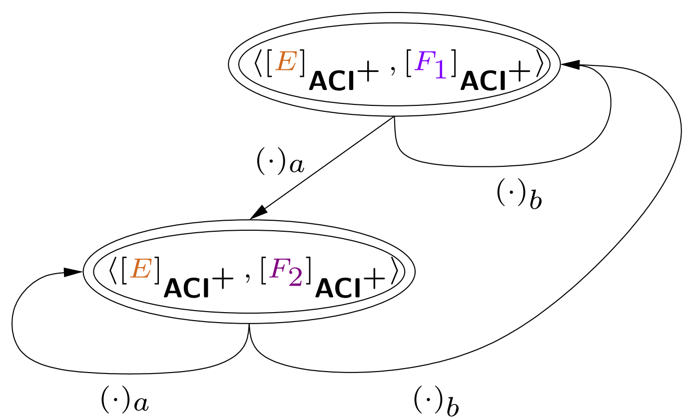


$\left\{ \langle [E]_{\mathsf{ACI}^+}, [F_1]_{\mathsf{ACI}^+} \rangle, \langle [E]_{\mathsf{ACI}^+}, [F_2]_{\mathsf{ACI}^+} \rangle \right\}$ is a finite bisimulation.
By the (finitary) coinduction principle $(a+b)^* =_L (a^*.b)^*.a^*$ follows.

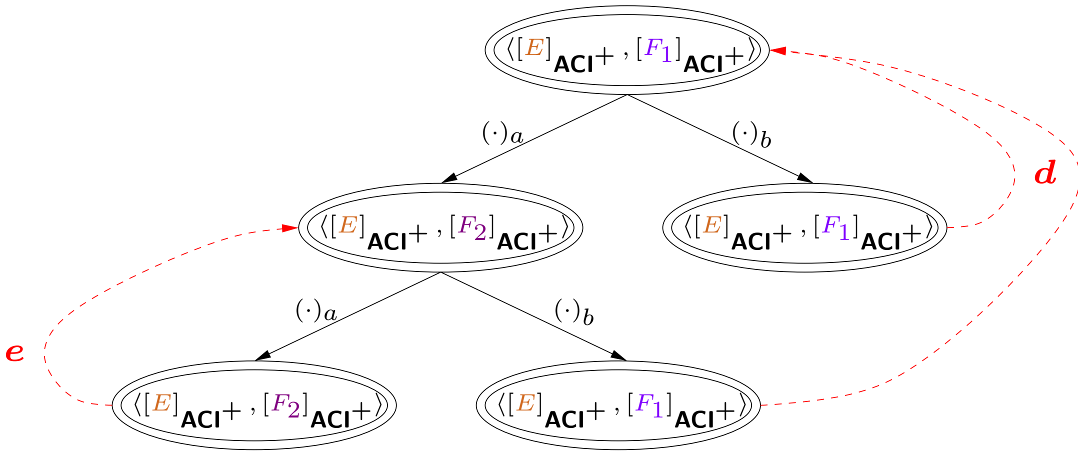
Relation with cREG$_0$: Finite Bisimulations . . .



This bisimulation defines the following automaton in $\mathcal{R}(\Sigma)_{\mathsf{ACI}^+}$:



. . . correspond to . . . derivations in cREG$_0$



is an "unwinding" of the bisimulation between $[E]_{\mathbf{ACI}^+}$ and $[F_2]_{\mathbf{ACI}^+}$ which corresponds to the **cREG$_0$**-derivation

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

Soundness and Completeness of $\mathbf{cREG_0}$

Theorem. $\mathbf{cREG_0}$ *is sound and complete with respect to* $=_L$ *:*

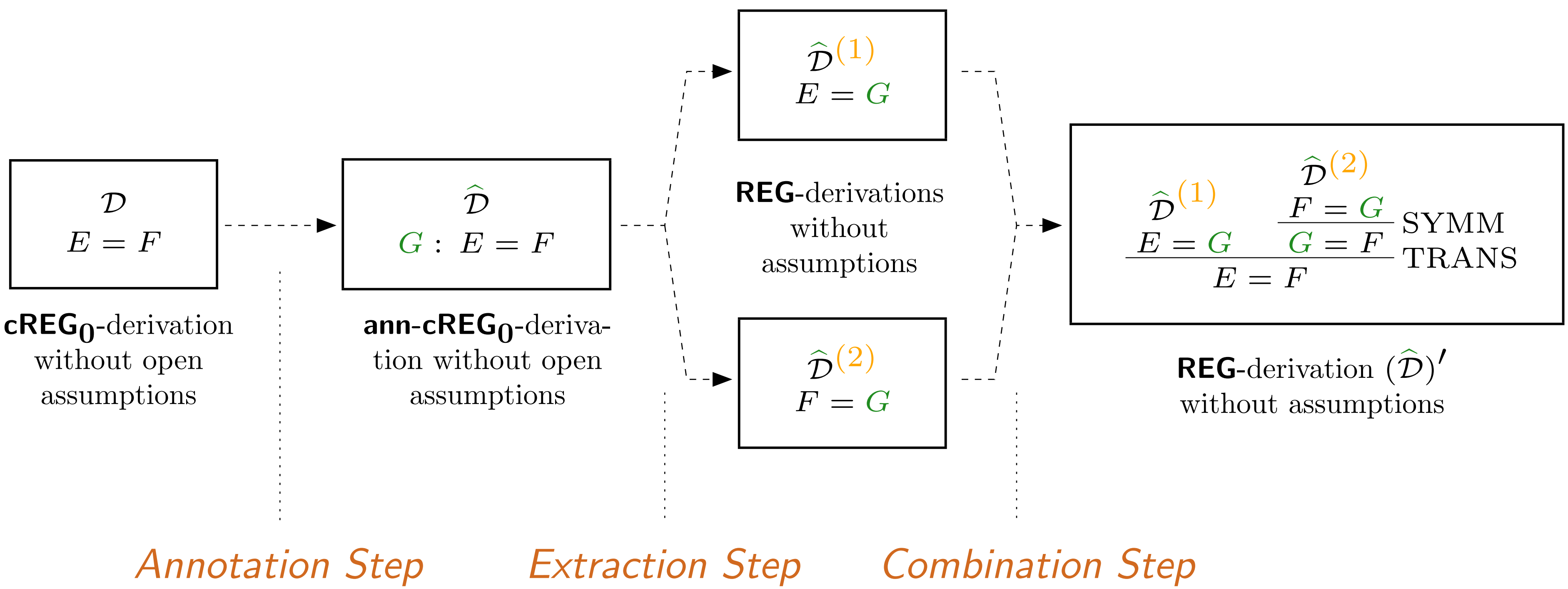
$$\textit{for all } E, F \in \mathcal{R}(\Sigma) : \left[\vdash_{\mathbf{cREG_0}} E = F \iff E =_L F \right].$$

Hint at the Proof.

"$\Leftarrow$": argue as just explained for the example.

"$\Rightarrow$": Let $\mathcal{D}$ a derivation in $\mathbf{cREG_0}$ with conclusion $E = F$.
Then $\{\langle [G]_{\mathbf{ACI}^+}, [H]_{\mathbf{ACI}^+} \rangle \mid G = H \text{ occurs in } \mathcal{D}\}$ is a finite bisimulation between $[E]_{\mathbf{ACI}^+}$ and $[F]_{\mathbf{ACI}^+}$ in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$.
By the finitary coinduction principle, $E =_L F$ follows.

A Transformation from cREG_0 to REG



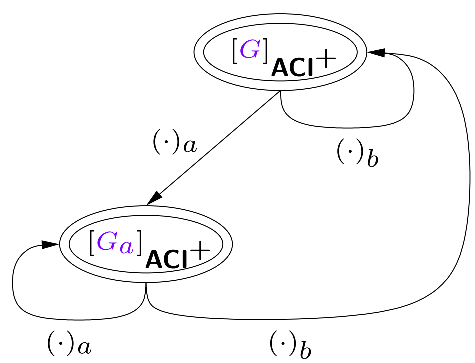
The Annotation Step (in our example)

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

The Annotation Step (in our example)

$$
\text{COMP/FIX},\ \boldsymbol{d}\ \dfrac{\dfrac{\text{COMP/FIX},\ \boldsymbol{e}\ \dfrac{\dfrac{(1.e:\ E = F_2)^{\boldsymbol{e}}}{1.e:\ E_a = (F_2)_a} \qquad \dfrac{(1.d:\ E = F_1)^{\boldsymbol{d}}}{1.d:\ E_b = (F_2)_b}}{a^* + a^*b.d:\ E = F_2}}{a^* + a^*b.d:\ E_a = (F_1)_a} \qquad \dfrac{(1.d:\ E = F_1)^{\boldsymbol{d}}}{1.d:\ E_b = (F_1)_b}}{\underbrace{(aa^*b + b)^*(1 + aa^*)}_{\equiv G}:\ E = F_1}
$$

The annotation G in this **ann-cREG$_0$**-deriv. describes the bisimulation betw. E and F_1; it has the following gen. subautomaton in $\mathcal{R}(\Sigma)_{\mathbf{ACI}^+}$:



The Extraction Step

$\widehat{\mathcal{D}}$

$G : E = F$

ann-cREG$_0$-derivation without open assumptions

$\widehat{\mathcal{D}}^{(1)}$

$E = G$

$\widehat{\mathcal{D}}^{(2)}$

$F = G$

REG-derivations without assumptions

The Extraction Step (the deriv. $\widehat{\mathcal{D}}^{(1)}$ in our example)

$$
\begin{prooftree}
\AxiomC{$\mathcal{D}^{(E)}$}
\noLine
\UnaryInfC{$E = 1 + a.E_a + b.E_b$}
\AxiomC{}
\LeftLabel{REFL, $\mathsf{App}_r\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E = 1.E$}
\LeftLabel{$\mathsf{App}_l\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E_a = E$}
\LeftLabel{CTXT}
\UnaryInfC{$a.E_a = a.E$}
\AxiomC{}
\RightLabel{REFL, $\mathsf{App}_r\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E = 1.E$}
\RightLabel{$\mathsf{App}_l\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E_b = E$}
\RightLabel{CTXT}
\UnaryInfC{$b.E_b = b.E$}
\RightLabel{$+$}
\BinaryInfC{$a.E_a + b.E_b = a.E + b.E$}
\RightLabel{CTXT}
\UnaryInfC{$1 + a.E_a + b.E_b = 1 + a.E + b.E$}
\RightLabel{TRANS}
\BinaryInfC{$E = 1 + a.E + b.E$}
\RightLabel{$\mathsf{App}_r\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E = a.E + (1 + b.E)$}
\RightLabel{FIX}
\UnaryInfC{$E = a^*(1 + b.E)$}
\end{prooftree}
$$

$$
\begin{prooftree}
\AxiomC{$\mathcal{D}^{(E)}$}
\noLine
\UnaryInfC{$E = 1 + a.E_a + b.E_b$}
\AxiomC{$E = a^*(1 + b.E)$}
\RightLabel{$\mathsf{App}_{l/r}\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E_a = a^* + a^*b.E$}
\RightLabel{CTXT}
\UnaryInfC{$a.E_a = a.(a^* + a^*b.E)$}
\AxiomC{}
\RightLabel{REFL, $\mathsf{App}_r\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E = 1.E$}
\RightLabel{$\mathsf{App}_l\mathsf{Ax}_{\mathbf{ACI}}+$}
\doubleLine
\UnaryInfC{$E_b = E$}
\RightLabel{CTXT}
\UnaryInfC{$b.E_b = b.E$}
\RightLabel{$+$}
\BinaryInfC{$a.E_a + b.E_b = a.(a^* + a^*b.E) + b.E$}
\RightLabel{CTXT}
\UnaryInfC{$1 + a.E_a + b.E_b = 1 + a.(a^* + a^*b.E) + b.E$}
\RightLabel{$\mathsf{App}_r\mathsf{Ax}_{\mathbf{ACI}}+$}
\UnaryInfC{$1 + a.E_a + b.E_b = (aa^*b + b).E + (1 + aa^*)$}
\RightLabel{TRANS}
\BinaryInfC{$E = (aa^*b + b).E + (1 + aa^*)$}
\RightLabel{FIX}
\UnaryInfC{$E = (aa^*b + b)^*(1 + aa^*)$}
\end{prooftree}
$$

The Combination Step

$$\begin{array}{c}\widehat{\mathcal{D}}^{(1)}\\ E = G\end{array}$$

REG-derivations
without
assumptions

$$\begin{array}{c}\widehat{\mathcal{D}}^{(2)}\\ F = G\end{array}$$

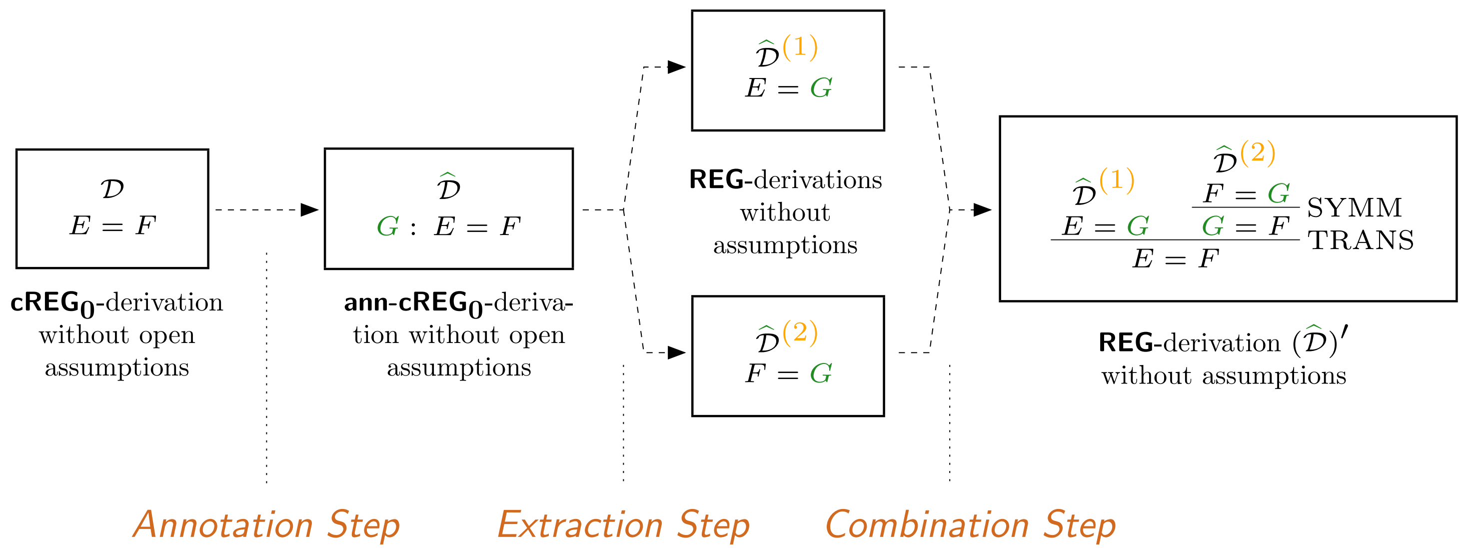
$$\dfrac{\begin{array}{c}\widehat{\mathcal{D}}^{(1)}\\ E = G\end{array} \qquad \dfrac{\begin{array}{c}\widehat{\mathcal{D}}^{(2)}\\ F = G\end{array}}{G = F}\,\mathrm{SYMM}}{E = F}\,\mathrm{TRANS}$$

REG-derivation $(\widehat{\mathcal{D}})'$
without assumptions

A Transformation from $\mathbf{cREG_0}$ to REG

Theorem. *Every derivation $\mathcal{D}$ in* **cREG$_0$** *without open assumptions can effectively be transformed into a derivation $\mathcal{D}'$ in* **REG** *with the same conclusion as $\mathcal{D}$.*

Proof.



Corollary. *The system* **REG** *is complete with respect to* $=_L$.

Summary

In the paper I have

– restated Rutten’s coinduction principle for regular expression equivalence $=_L$ as a finitary coinduction principle (that can be used to decide $=_L$);

– introduced a coinductively motivated, complete proof system **cREG**$_0$ for $=_L$;

– described an effective proof-theoretic transformation from **cREG**$_0$ to **REG**, the “reversed” form of Salomaa’s axiomatisation **F**$_1$;

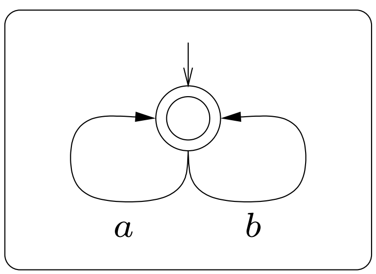
– thereby provided a coinductive completeness proof for **REG** (which can be “redone” for **F**$_1$).

Related Work. A Related Problem?

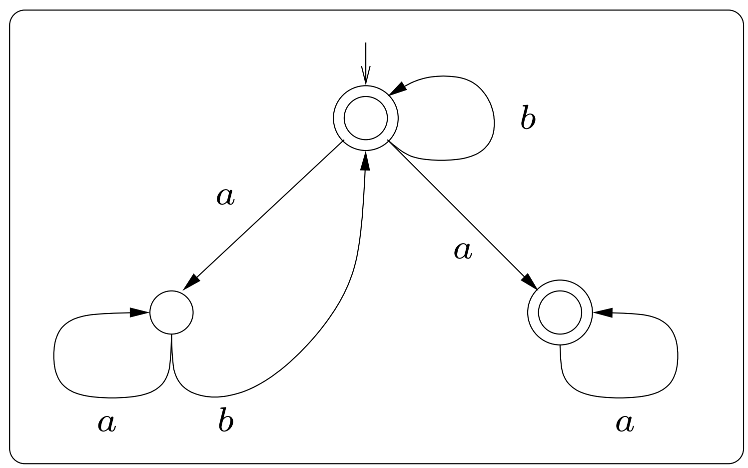
- Proof systems for *recursive type equality*: a transformation from the coinductive axiomatisation by Brandt-Henglein into the “traditional” axiomatisation by Amadio-Cardelli (in my thesis).

- Milner’s problem (1984): Find a system that is weaker than **REG**, but complete for *star behaviours*? (Is $\mathrm{BPA}^*_{\delta,\epsilon}$ complete?)

$$(a+b)^* \quad \neq_{\mathrm{BPA}^*_{\delta,\epsilon}} \quad (a^*b)^*a^*$$



$\not\Leftrightarrow$



Thanks for your attention!

References

[1] Amadio, R.M., Cardelli, L.: "Subtyping Recursive Types", *ACM Transactions on Programming Languages and Systems 15 (4)* pp. 575–631, 1993.

[2] Aanderaa, S.: On the algebra of regular expressions, *Applied Mathematics*, Harvard University (January 1965) 1–18.

[3] Brandt, M., Henglein, F.: "Coinductive axiomatization of recursive type equality and subtyping", *Fundamenta Informaticae* **33** (1998) 1–30.

[4] Brzozowski, J.A.: "Derivatives of regular expressions", *Journal of the ACM* **11** (1964) 481–494.

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

[6] Hüttel, H., Stirling, C.: "Actions Speak Louder Than Words: Proving Bisimilarity for Context-Free Processes", *Journal of Logic and Computation* **8**:4 (1998) 485–509.

[7] Grabmayer, C.: *Relating Proof Systems for Recursive Types*, PhD thesis, Vrije Universiteit Amsterdam (2005).

[8] Milner, R.: "A Complete Inference System for a Class of Regular Behaviours", *Journal of Computer and System Sciences 28*, pp. 439–466, 1984.

[9] Rutten, J.J.M.M.: Automata and Coinduction (an Exercise

in Coinduction), *Proceedings of CONCUR ’98*, LNCS 1466, Springer (1998) 194–218.

[10] Salomaa, A.: Two complete axiom systems for the algebra of regular events, *Journal of the ACM* **13**:1 (1966) 158–169.

[11] Troelstra, A.S., Schwichtenberg, H.: *Basic Proof Theory*, Cambridge University Press (1996, 2000).

Salomaa's Axiomatization $\mathbf{F}_1$ of $=_L$

The *axioms* of $\mathbf{F}_1$:

A_1	$E + (F + G) = (E + F) + G$	A_7	$1.E = E$
A_2	$E.(F.G) = (E.F).G$	A_8	$0.E = 0$
A_3	$E + F = F + E$	A_9	$E + 0 = E$
A_4	$E.(F + G) = E.F + E.G$	A_{10}	$E^* = 1 + E^*.E$
A_5	$(E + F).G = E.G + F.G$	A_{11}	$E^* = (1 + E)^*$
A_6	$E + E = E$		

The *inference rules* of $\mathbf{F}_1$:

$$\frac{E = F \qquad\qquad C[E] = G}{C[F] = C[E], \ C[F] = G} \qquad\qquad \frac{E = E.F + G}{E = G.F^*} \text{(if } o(F) = 0)$$

Brzozowski Derivatives

Brzozowski derivatives $(\cdot)_a : \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)$ are defined by

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

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

mimic language derivatives. Also, a function $o : \mathcal{R}(\Sigma) \to \mathcal{R}(\Sigma)$ can be defined that mimics the function $o_{\mathcal{L}}$:

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

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

The Annotation Step

$$\boxed{\begin{array}{c} \mathcal{D} \\ E = F \end{array}} \dashrightarrow \boxed{\begin{array}{c} \widehat{\mathcal{D}} \\ G : E = F \end{array}}$$

cREG$_0$-derivation without open assumptions

ann-cREG$_0$-derivation without open assumptions

Justifying the Annotation Step

Lemma. $\mathbf{cREG_0}$ *and* $\mathbf{ann\text{-}cREG_0}$ *are linked by an annotating transformation* $\hat{(\cdot)}$ *and an annotating-deleting transf.* $\check{(\cdot)}$ *:*

$$\underbrace{\boxed{\begin{array}{c}\{[E_i = F_i]^{d_i}\}_{i=1,\ldots,m} \\ \mathcal{D} \\ E = F\end{array}}}_{=\check{(\tilde{\mathcal{D}})}} \quad \begin{array}{c}\xmapsto{\hat{(\cdot)}} \\ \xleftarrow[\check{(\cdot)}]{} \end{array} \quad \underbrace{\boxed{\begin{array}{c}\{[1.d_i :\; E_i = F_i]^{d_i}\}_{i=1,\ldots,m} \\ \tilde{\mathcal{D}} \\ (G_0+)\sum_{j=1}^{m} G_j.d_j : E = F\end{array}}}_{=\hat{\mathcal{D}}}$$

$\mathbf{cREG_0}$*-derivation* $\mathcal{D}$ $\qquad$ $\mathbf{ann\text{-}cREG_0}$*-derivation* $\tilde{\mathcal{D}}$

(Each derivation in $\mathbf{ann\text{-}cREG_0}$ *can be written in the form of the prooftree on the right.)*

The Annotated Version ann-cREG$_0$ of cREG$_0$ (I)

The *axioms* and possible *marked assumptions* in **ann-cREG$_0$**(Σ, Δ) :

$$\text{(REFL)} \ \overline{E : E = E} \qquad \text{(Assm)} \ (d : E = F)^{d} \quad (\text{with } d \in \Delta) \ .$$

The *inference rules* of **ann-cREG$_0$**(Σ, Δ) :

$$\frac{\begin{array}{c}\mathcal{D}_1 \\ G : C[E_1] = F\end{array}}{G : C[E_2] = F}\,\mathsf{App}_l\mathsf{Ax}_{\mathbf{REG}} \qquad \frac{\begin{array}{c}\mathcal{D}_1 \\ G : F = C[E_1]\end{array}}{G : F = C[E_2]}\,\mathsf{App}_r\mathsf{Ax}_{\mathbf{REG}}$$

(if $E_1 = E_2$ or $E_2 = E_1$ is an axiom of **REG**),

And annotated versions of the rules COMP and COMP/FIX (on the two following slides).

The Annotated Version ann-cREG$_0$ of cREG$_0$ (II)

Annotated version of the rule COMP:

$$\frac{\begin{array}{c}\mathcal{D}_1 \\ G_{10} + \sum_{j=1}^{m} G_{1j}.d_j \,:\, E_{a_1} = F_{a_1}\end{array} \quad \ldots \quad \begin{array}{c}\mathcal{D}_n \\ G_{n0} + \sum_{j=1}^{m} G_{nj}.d_j \,:\, E_{a_n} = F_{a_n}\end{array}}{\left(o(E) + \sum_{i=1}^{n} G_{i0}\right) + \sum_{j=1}^{m}\left(\sum_{i=1}^{n} a_i.G_{ij}\right).d_j \,:\, E = F}$$

(if $o(E) = o(F)$).

(Here we have assumed $\Sigma = \{a_1, \ldots, a_n\}$).

The Annotated Version ann-cREG$_0$ of cREG$_0$ (III)

Annotated version of the rule COMP/FIX:

$$
\dfrac{\begin{array}{ccc} [d_k : E = F]^{d_k} & & [d_k : E = F]^{d_k} \\ \mathcal{D}_1 & & \mathcal{D}_n \\ G_{10} + \sum_{j=1}^{m} G_{1j}.d_j : E_{a_1} = F_{a_1} & \ldots & G_{n0} + \sum_{j=1}^{m} G_{nj}.d_j : E_{a_n} = F_{a_n} \end{array}}{\begin{array}{l} \left(\sum_{i=1}^{n} a_i.G_{ik}\right)^{*}.\left(o(E) + \sum_{i=1}^{n} G_{i0}\right) + \\ \quad + \sum_{j=1, j \neq k}^{m} \left(\sum_{i=1}^{n} a_i.G_{ik}\right)^{*}.\left(\sum_{i=1}^{n} a_i.G_{ij}\right).d_j : E = F \end{array}} \, d_k
$$

(if $o(E) = o(F)$).

The Extraction Step (the deriv. $\widehat{\mathcal{D}}^{(2)}$ in our example)

$$\dfrac{}{F_2 = 1.F_2}\ \text{REFL, App}_r\text{Ax}_{\textbf{ACI}+} \qquad \dfrac{}{F_1 = 1.F_1}\ \text{REFL, App}_r\text{Ax}_{\textbf{ACI}+}$$

$$\dfrac{F_2 = 1.F_2}{(F_2)_a = F_2}\ \text{App}_l\text{Ax}_{\textbf{ACI}+} \qquad \dfrac{F_1 = 1.F_1}{(F_1)_b = F_1}\ \text{App}_l\text{Ax}_{\textbf{ACI}+}$$

$$\dfrac{(F_2)_a = F_2}{a.(F_2)_a = a.F_2}\ \text{CTXT} \qquad \dfrac{(F_1)_b = F_1}{b.(F_1)_b = b.F_1}\ \text{CTXT}$$

$$\dfrac{a.(F_2)_a = a.F_2 \qquad b.(F_1)_b = b.F_1}{a.(F_2)_a + b.(F_2)_b = a.F_2 + b.F_1}\ +$$

$$\dfrac{a.(F_2)_a + b.(F_2)_b = a.F_2 + b.F_1}{1 + a.(F_2)_a + b.(F_2)_b = 1 + a.F_2 + b.F_1}\ \text{CTXT}$$

$$\dfrac{\overset{\mathcal{D}^{(F_2)}}{F_2 = 1 + a.(F_2)_a + b.(F_2)_b} \qquad 1 + a.(F_2)_a + b.(F_2)_b = 1 + a.F_2 + b.F_1}{F_2 = 1 + a.F_2 + b.F_1}\ \text{TRANS}$$

$$\dfrac{F_2 = 1 + a.F_2 + b.F_1}{F_2 = a.F_2 + (1 + b.F_1)}\ \text{App}_r\text{Ax}_{\textbf{ACI}+}$$

$$\dfrac{F_2 = a.F_2 + (1 + b.F_1)}{F_2 = a^*(1 + b.F_1)}\ \text{FIX}$$

$$\dfrac{F_2 = a^*(1 + b.F_1)}{(F_1)_a = a^* + a^*b.F_1}\ \text{App}_{l/r}\text{Ax}_{\textbf{ACI}+} \qquad \dfrac{}{F_1 = 1.F_1}\ \text{REFL, App}_r\text{Ax}_{\textbf{ACI}+}$$

$$\dfrac{F_1 = 1.F_1}{(F_1)_b = F_1}\ \text{App}_l\text{Ax}_{\textbf{ACI}+}$$

$$\dfrac{(F_1)_a = a^* + a^*b.F_1}{a.(F_1)_a = a.(a^* + a^*b.F_1)}\ \text{CTXT} \qquad \dfrac{(F_1)_b = F_1}{b.(F_1)_b = b.F_1}\ \text{CTXT}$$

$$\dfrac{a.(F_1)_a = a.(a^* + a^*b.F_1) \qquad b.(F_1)_b = b.F_1}{a.(F_1)_a + b.(F_1)_b = a.(a^* + a^*b.F_1) + b.F_1}\ +$$

$$\dfrac{a.(F_1)_a + b.(F_1)_b = a.(a^* + a^*b.F_1) + b.F_1}{1 + a.(F_1)_a + b.(F_1)_b = 1 + a.(a^* + a^*b.F_1) + b.F_1}\ \text{CTXT}$$

$$\dfrac{\overset{\mathcal{D}^{(F_1)}}{F_1 = 1 + a.(F_1)_a + b.(F_1)_b} \qquad 1 + a.(F_1)_a + b.(F_1)_b = 1 + a.(a^* + a^*b.F_1) + b.F_1}{F_1 = 1 + a.(a^* + a^*b.F_1) + b.F_1}\ \text{TRANS}$$

$$\dfrac{F_1 = 1 + a.(a^* + a^*b.F_1) + b.F_1}{F_1 = (aa^*b + b).F_1 + (1 + aa^*)}\ \text{App}_r\text{Ax}_{\textbf{ACI}+}$$

$$\dfrac{F_1 = (aa^*b + b).F_1 + (1 + aa^*)}{F_1 = (aa^*b + b)^*(1 + aa^*)}\ \text{FIX}$$

"Fundamental Theorem of Formal Languages"

Lemma. *For all* $E \in \mathcal{R}(\Sigma)$,

$$E =_L o(E) + \sum_{i=1}^{n} a_i.E_{a_i}$$

holds, and a derivation in **REG**$^-$ *with this conclusion ("$=$" i.p.o. "$=_L$") can effectively be constructed.*

Proof. By induction on the syntactical structure of regular expressions.

Justifying the Extraction Step

Lemma. *For every derivation*

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

in **ann-cREG$_0$** *it is possible to extract effectively derivations*

$$\begin{array}{c} \tilde{\mathcal{D}}^{(1)} \\ E = (G_0+)\sum_{j=1}^{m} G_j.E_j \end{array} \quad \text{and} \quad \begin{array}{c} \tilde{\mathcal{D}}^{(2)} \\ F = (G_0+)\sum_{j=1}^{m} G_j.F_j \end{array}$$

in **REG**.

The Process Algebra $\mathrm{BPA}^*_{\delta,\epsilon}$

The *axioms* of $\mathrm{BPA}^*_{\delta,\epsilon}$:

(A1)	$x + y = y + x$	(A6)	$x + \delta = x$
(A2)	$x + (y + z) = (x + y) + z$	(A7)	$\epsilon.x = x$
(A3)	$x + x = x$	(A8)	$\delta.x = \delta$
(A4)	$(x + y).z = x.z + y.z$	(A9)	$x.\epsilon = x$
(A5)	$x.(y.z) = (x.y).z$	(KS2)	$x^* = \epsilon + x.x^*$
(KS3)	$(x + y)^* = x^*.(y.(x + y)^* + \epsilon)$	(KS1)	$x^* = (\epsilon + x)^*$

Possible *inference rules* for $\mathrm{BPA}^*_{\delta,\epsilon}$: REFL, SYMM, TRANS, and

$$\frac{x = y}{C[x] = C[y]}\,\text{CTXT} \qquad\qquad \frac{x = f.x + z}{x = f^*.z}\,\text{FIX (if } o(f) = 0)$$