IPSJ Online Transactions Vol. 1 53–65 (July 2008)

Regular Paper

Product Derivatives of Regular Expressions

Taro Suzuki[†1] and Satoshi Okui[†2]

We propose a novel extension of Brzozowski's derivative of regular expressions, called product derivatives. It takes two regular expressions and the meaning of its result is stated as follows: a product derivative of R with respect to S is a regular expression obtained by the consumption of all the sequences in the first regular expression by the second, i.e., the result is the product of Brzozowski's derivatives of R with respect to sequences in S. We develop an algorithm for our derivatives in coinductive manner. The termination of the algorithm is shown based on the proof method by Brandt and Henglein. We expect the derivative proposed in this talk is an important step to the development of the derivatives of a regular hedge expression taking a regular hedge expression as input, which is useful for transformation of XML documents with function terms typed with regular hedge expressions. The situation, for example, occurs in the transformation of XML documents with the links to the other Web services, such as Active XML documents.

1. Introduction

The notion of derivative of regular expressions was proposed by Brzozowski to give an algorithm for generating deterministic finite automata from given regular expressions[5]. Given a finite sequence w of symbols and a regular expression S, his algorithm computes a regular expression, denoted by $w^{-1}S$, corresponding a set of sequence $\{v \mid wv \in S\}$. Later, derivatives have been recognized as a quite useful tool. For instance, it is used to present various computational procedures in the algebra of regular expressions[6],[11] and the equivalence of regular expressions[2],[9]. It is also used in the research area on grammatical inference[3],[7]. Recently, researchers on XML have paid an attention to derivatives[13] since XML employs regular expressions as its schema language.

In this paper, we propose an extension of Brzozowski's derivatives called *product derivative*, which have arisen from our work on XML transformation based on regular expression (RE, for short) types[12],[14]. Our original motivation for generalizing derivatives comes from the pattern matching involving RE-typed functions. This naturally arises in the situation where lazy evaluation is unavoidable in the XML transformation that involves infinite data structures, which is often seen in functional programming in Haskell. Moreover, there are situations peculiar to XML transformation where the pattern matching involving RE-typed functions is useful. An extension of XML, called Active XML, is developed in INRIA[1]. Active XML documents contain links to Web services, which are dynamically replaced with XML documents when the information is needed. Such links are considered as functions that never should be evaluated in transformation.

Let us take up an example of transforming contact lists given in an XML format. Here, a contact list means an unsorted sequence of e-mail addresses or phone numbers like the following:

```
<contact>
<email>"taro@mac.com"</email>
<phone>"4391"</phone>
<email>"taro@xp.com"</email>
</contact>
```

We represent the XML documents using term notation introduced in our previous paper[14]. For instance, the above contact list is written as follows.

```
contact[email["taro@mac.com"]
        phone["4391"]
        email["taro@xp.com"]]
```

We assume that the strings (entities enclosed by `"` such as `"taro@mac.com"`) are of type `String`, the terms of the form `email[x]` and `phone[x]`, where `x` of type `String`, are of type `e` and of type `p`, respectively. Letting `x` be a variable of type `(e|p)*e(e|p)*` and `y` of type `(e|p)+`, the function `cut`, which extracts a proper prefix of a given contact list including at least one e-mail address, is described as the transformation rule

`cut(contact[x y])` → `contact[x]`

where `x y` indicates the concatenation of `x` and `y`.

[†1] The University of Aizu
[†2] Chubu University

Now, consider the case in which a contact list is given as a result of inquiries for external Web services as links embedded in XML documents. Suppose that three web services are available: `gc` of type `String` → `(e|p)+` that returns a contact list of a person, `ge` of type `String` → `e+` that returns a list of e-mail addresses of a person, and `gp` of type `String` → `p+` that returns a list of phone numbers of a person. Consider the XML document

`contact[gc("taro") ge("jiro") gp("goro")]`

with links to external Web services. We transform it using the function `cut` without the inquiry of the Web services during the transformation so that the transformed XML document can call Web services embedded.

We would like to perform *type-safe* pattern matching, i.e., the RE-types of the values bound to x and y should be a subtype of the type of x and y respectively. Pattern matching is performed in incremental manner by introducing intermediate variables [12]. First, we know that `x` has a binding `x` $\mapsto$ `gc("taro") x1`, where `x1` is an intermediate variable, since the type of `x` rejects the binding of `x` to the empty sequence `()`. Consider the type of `x1`, denoted by $T_{\mathtt{x1}}$. As `gc("taro")` should not be evaluated during the transformation, we should take account of arbitrary non-empty sequences composed of e-mail addresses and phone numbers as the value of `gc("taro")`. Thus, a necessary condition for type-safe pattern match is that for every sequence $v \in$ `(e|p)+`, $T_{\mathtt{x1}} \subseteq \{w \mid vw \in \mathtt{(e|p)*e(e|p)*}\} = v^{-1}(\mathtt{(e|p)*e(e|p)*})$, where the last expression is the Brzozowski's derivative. It follows that $T_{\mathtt{x1}}$ is the maximal RE-type satisfying $T_{\mathtt{x1}} \subseteq (v^{-1}\mathtt{(e|p)*e(e|p)*})$ for all $v \in$ `(e|p)+`, thereby

$$T_{\mathtt{x1}} = \bigcap_{T \in \mathtt{(e|p)+}} T^{-1}(\mathtt{(e|p)*e(e|p)*}) = \mathtt{(e|p)*e(e|p)*}.$$

Likewise, we know that the only possible binding for `x1` is `x1` $\mapsto$ `ge("jiro") x2` since the type of `x1` does not contain the empty sequence. From the intersection of the derivatives of `(e|p)*e(e|p)*` with respect to the elements of `e+`, we know that the type of `x2` is `(e|p)*`. Because the type of `x2` contains the empty sequence and that of `y` does not, finally we obtain the bindings `x` $\mapsto$ `gc("taro") ge("jiro")` and `y` $\mapsto$ `gp("goro")`. The obtained solution in the pattern match is type safe because the type of the value bound to `x`, i.e., `(e|p)+e+`, is included by `(e|p)*e(e|p)*`, i.e., the type of x, and the type of the value bound to `y`, i.e., `p+`, is included by the type `y`, i.e., `(e|p)+`.

The aim of this paper is to give an algorithm for the computation of product derivatives. This paper is organized as follows. We first recall the notions and the existent results crucial for this paper in Section 2. Next we formally define the notion of product derivatives and it is characterized using the greatest fixed point of a function in Section 3. In Section 4 we consider a general settings: when a function f is defined as a greatest fixed point of F, an algorithm to compute a value $f(x)$ at x is constructed by composing computationally defined functions determined according to F, provided one of them is terminating. We also show a sufficient condition for termination property of the constituent function. In Section 5, we show the sufficient condition holds for the product derivatives and hence an algorithm for generating the product derivatives. Finally, we conclude the paper with some remarks in Section 6.

2. Preliminaries

In this section we recall the notions and properties on Brzozowski's derivative and coinduction.

Regular expressions and their interpretation as sets are defined as usual. As in the literature, we identify a regular expression R and the regular set represented by R, otherwise mentioned. Throughout the paper, we denote an alphabet on which regular expressions are constructed, by Σ, symbols in Σ by $a, b, \ldots$, sequences in Σ^* by u, v, w, and regular expressions by capitals from P to U, primed or subscripted if necessary. The set of regular expressions (built on Σ) is denoted by *Reg*.

In order to know whether a regular expression contains the empty sequence, denoted by ε, we define the function $\delta : \mathit{Reg} \to \mathit{Reg}$ as follows: $\delta(R) = 1$ if $\varepsilon \in R$; $\delta(R) = 0$ otherwise. We say a regular expression is *nullable* if $\delta(R) = 1$.

The following definition of derivatives is given by Brzozowski [5].

Definition 1 *Let R be a regular expression and w a finite sequence. The* derivative *of R with respect to w, denoted by $w^{-1}R$, is defined as $w^{-1}R = \{v \mid wv \in R\}$.*

He also showed that the derivatives can be computed in inductive way.

Theorem 1 (Brzozowski [5]) The derivative of R with respect to a symbol a is inductively computed as follows.

$$\begin{array}{ll} a^{-1}0 = a^{-1}1 = 0 & a^{-1}PQ = (a^{-1}P)Q + \delta(P)a^{-1}Q \\ a^{-1}a = 1 & a^{-1}P+Q = a^{-1}P + a^{-1}Q \\ a^{-1}b = 0 \quad (a \neq b) & a^{-1}P^* = (a^{-1}P)P^* \end{array}$$

The derivivative of R with respect to a sequence is computed as follows: $\varepsilon^{-1}R = R$ and $(aw)^{-1}R = w^{-1}(aR)$.

He also showed the finiteness of derivatives. The regular expressions are *similar* if one can be transformed into another via the following set (ACI+) of identities:

$R + R = R$,
$P + Q = Q + P$,
$(P + Q) + R = P + (Q + R)$.

Theorem 2 (Brzozowski [5]) Every regular expression has only a finite number of dissimilar derivatives.

Next we recall the notions on coinduction. We follow the standard definition of complete lattices and fixed points in the literature. The greatest fixed point of a function f is denoted by νf. Let f be a function from A to A. An element x of A is *f-consistent* if $x \sqsubseteq f(x)$. The following proposition, so-called principle of coinduction, is a consequence of Knaster-Tarski fixed point theorem [15].

Proposition 1 *Let f be a monotonic function from a complete lattice A to itself and $x \in A$. If x is f-consistent then $x \sqsubseteq \nu f$.*

Let $(A, \sqsubseteq_A)$ and $(B, \sqsubseteq_B)$ be complete lattices. A function $f : A \to B$ is *$\sqcap$-continuous* if $\sqcap\{f(x) \mid x \in X\} = f(\sqcap X)$ for any subset X of A such that the elements of X forms a nonincreasing sequence with respect to $\sqsubseteq$. Note that a $\sqcap$-continuous function is monotonic.

Theorem 3 *Let f be a $\sqcap$-continuous function from a complete lattice A to itself. Then $\nu f = \sqcap\{f^n(\top) \mid n \in \mathcal{N}\}$, where $\mathcal{N}$ is a set of non-negative integers and $\top$ the greatest element of A.*

3. Product Derivative

In this section the product derivative is introduced as the intersection of Brzozowski's derivatives. We demonstrate that the product derivaties are characterized by fixed points of a function. Finally we show that the product derivative is the greatest fixed point of a function.

Definition 2 *For any regular expressions R and S, the product derivative of R with respect to S, denoted by $R^{-1}S$, is defined as follows.*

$$R^{-1}S = \bigcap_{w \in R} w^{-1}S$$

For instance, the value of $(aa + b)^{-1}(a^*b^*)$ is b^* from the following computation.

$$\begin{aligned} & (aa + b)^{-1}(a^*b^*) \\ &= (aa)^{-1}(a^*b^*) \cap b^{-1}(a^*b^*) \\ &= a^{-1}(a^{-1}(a^*b^*)) \cap b^* \\ &= a^{-1}(a^*b^*) \cap b^* \\ &= a^*b^* \cap b^* = b^*. \end{aligned}$$

The first equation is derived from Definition 2 and the others from Theorem 1. As an another example, let us compute $(a^*)^{-1}(b^*(ab^*)^*)$. From the above definition,

$$\begin{aligned} & (a^*)^{-1}(b^*(ab^*)^*) \\ &= \varepsilon^{-1}(b^*(ab^*)^*) \cap a^{-1}(b^*(ab^*)^*) \cap (aa)^{-1}(b^*(ab^*)^*) \cap \cdots. \end{aligned}$$

Since $a^{-1}(b^*(ab^*)^*) = b^*(ab^*)^*$ from Theorem 1, we infer that $(a^*)^{-1}(b^*(ab^*)^*) = b^*(ab^*)^*$.

We, however, cannot compute the value of $(a^*)^{-1}(b^*(ab^*)^*)$ in inductive manner:

$$\begin{aligned} & (a^*)^{-1}(b^*(ab^*)^*) \\ &= \varepsilon^{-1}(b^*(ab^*)^*) \cap (aa^*)^{-1}(b^*(ab^*)^*) \\ &= b^*(ab^*)^* \cap (a^*)^{-1}(a^{-1}(b^*(ab^*)^*)) \\ &= b^*(ab^*)^* \cap (a^*)^{-1}(b^*(ab^*)^*). \end{aligned}$$

The recursive equation obtained above implies that the product derivatives are fixed points of a function and we may need to compute an algorithm based on fixed points. In this paper we show that such an algorithm really works. First, we show that a function for product derivatives is characterized as the greatest fixed point of a function.

Definition 3 *We define the function $\mathcal{D} : (Reg \times Reg \to Reg) \to (Reg \times$*

Reg → *Reg*) *as follows.*

$\mathcal{D}(f)(0,S)=\Sigma^*$

$\mathcal{D}(f)(1,S)=S$

$\mathcal{D}(f)(aR,S)=f(R,a^{-1}S)$

$\mathcal{D}(f)((P+Q)R,S)=f(PR,S)\cap f(QR,S)$

$\mathcal{D}(f)(P^*R,S)=f(PP^*R,S)\cap f(R,S)$

Notice that the function $\mathcal{D}$ is $\cap$-continuous. We show the equivalence of $\nu\mathcal{D}(R,S)$ and $R^{-1}S$.

Lemma 1 *Let f be a $\mathcal{D}$-consistent function and $u\in P$, $v\in Q$. If $f(Q,u^{-1}S)\subseteq(uv)^{-1}S$ then $f(PQ,S)\subseteq(uv)^{-1}S$.*

Proof. Since $u\in P$ and $v\in Q$, P and Q are not equal to 0. The proof is performed by the induction on the structure of the regular expression P. If $P=1$ then the result immediately follows.

Suppose P is of the form aP'. Then there exists a $u'\in P'$ such that $u=au'$. Since $f(Q,{u'}^{-1}(a^{-1}S))=f(Q,(au')^{-1}S)\subseteq(au'v)^{-1}S=(u'v)^{-1}(a^{-1}S)$, the induction hypothesis yields $f(P'Q,a^{-1}S)\subseteq(u'v)^{-1}(a^{-1}S)$. Thus it follows $f(aP'Q,S)\subseteq\mathcal{D}(f)(aP'Q,S)=f(P'Q,a^{-1}S)\subseteq(au'v)^{-1}S$ from the $\mathcal{D}$-consistency of f and the definition of $\mathcal{D}$.

Suppose P is of the form $(P_1+P_2)P'$. Then there exists a $w\in P_i$ for some $i=1,2$ and $u'\in P'$ such that $u=wu'$. Since $f(Q,{u'}^{-1}(w^{-1}S))=f(Q,(wu')^{-1}S)\subseteq(wu'v)^{-1}S=(u'v)^{-1}(w^{-1}S)$, the repetitive applications of the induction hypothesis yields $f(P_iP'Q,S)\subseteq(uv)^{-1}S$. Then the result follows from the $\mathcal{D}$-consistency of f and the definition of $\mathcal{D}$.

Finally, suppose P is of the form R^*P'. Then $u\in R^iP'$ for some $i\geq 0$. We show the following claim:

For any $i\geq 0$, w and T, if $w\in R^iP'$ and $f(Q,w^{-1}T)\subseteq(wv)^{-1}T$ then $f(R^*P'Q,T)\subseteq(wv)^{-1}T$ holds.

Since the result immediately follows from this claim, we prove it by the induction on i.

The base case, $i=0$, follows from the first induction hypothesis and the $\mathcal{D}$-consistency of f. Suppose $w=w_1w_2$ where $w_1\in R$ and $w_2\in R^iP'$. If we assume $f(Q,w^{-1}T)\subseteq(wv)^{-1}T$ then the second induction hypothesis yields $f(R^*P'Q,{w_1}^{-1}T)\subseteq(wv)^{-1}T$ and hence, by the first induction hypothesis, $f(RR^*P'Q,T)\subseteq(wv)^{-1}T$. Therefore the claim follows from the $\mathcal{D}$-consistency of f. ∎

Theorem 4 *$R^{-1}S=\nu\mathcal{D}(R,S)$ for any regular expressions R and S.*

Proof. Obviously the product derivative is a fixed point of $\mathcal{D}$. The opposite holds if $\nu\mathcal{D}(R,S)\subseteq w^{-1}S$ for all $w\in R$. Since $\nu\mathcal{D}$ is $\mathcal{D}$-consistent, $\nu\mathcal{D}(1,w^{-1}S)\subseteq\mathcal{D}(\nu\mathcal{D})(1,w^{-1}S)=w^{-1}S$. Lemma 1 implies $\nu\mathcal{D}(R1,S)\subseteq w\varepsilon^{-1}S$ for all $w\in R$, which results in the desired result. ∎

4. Computation of the Greatest Fixed Point

In the previous section, we have shown that the greatest fixed point of $\mathcal{D}$ is a function for product derivatives. But this result is not enough to give an algorithm for generating product derivatives. We would like to have the greatest fixed point as a function defined in a more computational way. In order to obtain such a function, we treat more general settings, inspired from the method for subtype checking proposed by Gapeyev, et al.[8].

Definition 4 *Let A be a set and B a complete lattice with partial order $\sqsubseteq$. Suppose functions $\sigma:A\to fin(A)$ and $\gamma:A\to B$ are given, where $fin(A)$ means the set of the finite subsets of A. The function $\Phi_{\sigma,\gamma}:(A\to B)\to(A\to B)$ (a set of functions determined from σ and γ, precisely) is defined as follows.*

$\Phi_{\sigma,\gamma}(f)(x)=\sqcap\{f(y)\mid y\in\sigma(x)\}\sqcap\gamma(x)$

Notice that the function $\Phi_{\sigma,\gamma}$ is $\sqcap$-continuous. Thus, from Theorem 3 we obtain $\nu\Phi_{\sigma,\gamma}=\sqcap\{\Phi^n_{\sigma,\gamma}(\top)\mid n\in\mathcal{N}\}$, where $\top$ means a function mapping every element of A to the greatest element of B.

Example 1 *Consider $A=\{a,b,c,d,e\}$ and a function σ defined as follows:*

$\sigma(a)=\{b,c\}$,

$\sigma(b)=\{a\}$,

$\sigma(c)=\{d,e\}$,

$\sigma(d)=\sigma(e)=\emptyset$.

Then the greatest fixed point $\nu\Phi_{\sigma,\gamma}$ is

$\nu\Phi_{\sigma,\gamma}(a)=\nu\Phi_{\sigma,\gamma}(b)=\sqcap\{\gamma(a),\gamma(b),\gamma(c),\gamma(d),\gamma(e)\}$,

$\nu\Phi_{\sigma,\gamma}(c)=\gamma(c)\sqcap\gamma(d)\sqcap\gamma(e)$,

$\nu\Phi_{\sigma,\gamma}(d)=\gamma(d)$,

$\nu\Phi_{\sigma,\gamma}(e)=\gamma(e)$.

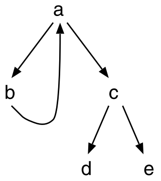

Fig. 1 A graph induced from σ in Example 1.

If σ is represented as a graph such that $x \to y$ iff $y \in \sigma(x)$, illustrated in **Fig. 1**, *then $\nu\Phi_{\sigma,\gamma}(x)$ is the greatest lower bound of the set obtained by applying γ to reachable nodes from x in the graph.*

As illustrated in this example, $\nu\Phi_{\sigma,\gamma}(x)$ is computed as follows: (1) enumerate all elements reachable from x in the graph induced from σ; (2) apply γ to the elements enumerated and compute their greatest lower bound. We introduce a function corresponding to each computational step mentioned above.

Definition 5 *The function $gfp_\sigma \colon fin(A) \to A \rightharpoonup fin(A)$ is defined as follows.*

```
$gfp_\sigma(X)(x) =$
    if $x \in X$ then $X$
    else let $\{x_1, \ldots, x_n\} = \sigma(x)$
    in let $X_0 = X \cup \{x\}$
    in let $X_1 = gfp_\sigma(X_0)(x_1)$
        $\vdots$
    in let $X_n = gfp_\sigma(X_{n-1})(x_n)$
    in $X_n$
```

The function $\Gamma_\gamma : fin(A) \to B$ is defined as $\Gamma_\gamma(X) = \sqcap\{\gamma(x) \mid x \in X\}$.

Note that for some X the function $gfp_\sigma(X)$ is not totally defined because it may be non-terminating at some $x \in A$. (Note that the type of gfp_σ contains $\rightharpoonup$, instead of $\to$, as the second arrow.) We show that if $gfp_\sigma(\emptyset)$ is totally defined then $\Gamma_\gamma \circ gfp_\sigma(\emptyset) = \nu\Phi_{\sigma,\gamma}$ holds.

We extend σ to a function on $fin(A)$: $\sigma(X) = \cup_{x\in X}\sigma(x)$. Note that the extended function σ is monotonic, i.e., if $X \subseteq Y$ then $\sigma(X) \subseteq \sigma(Y)$.

Lemma 2 *If $gfp_\sigma(X)(x)$ terminates then $X \cup \{x\} \subseteq gfp_\sigma(X)(x)$. Moreover, if $x \notin X$ then $gfp_\sigma(X)(x)$ includes $\sigma(x)$.*

Proof. Induction on the run of $gfp_\sigma(X)(x)$. If $x \in X$ then $gfp_\sigma(X)(x) = X$ and hence the result follows. Suppose $x \notin X$. By the induction hypothesis $X_{i-1} \cup \{x_i\} \subseteq gfp_\sigma(X_{i-1})(x_i) = X_i$ for $1 \le i \le n$. Therefore $X \cup \{x\} \cup \sigma(x) = X_0 \cup \sigma(x) \subseteq X_n = gfp_\sigma(X)(x)$. ∎

Lemma 3 *If $gfp_\sigma(X)(x)$ terminates then $\sigma(gfp_\sigma(X)(x)\backslash X) \subseteq gfp_\sigma(X)(x)$.*

Proof. Induction on the run of $gfp_\sigma(X)(x)$. If $x \in X$ then $gfp_\sigma(X)(x) = X$, which results in $\sigma(gfp_\sigma(X)(x)\backslash X) = \emptyset \subseteq gfp_\sigma(X)(x)$. Suppose $x \notin X$. Due to the termination of each $gfp_\sigma(X_{i-1})(x_i)$ for $1 \le i \le n$, the induction hypothesis yields $\sigma(X_i\backslash X_{i-1}) \subseteq gfp_\sigma(X_{i-1})(x_i)$. From Lemma 2 we obtain $X_{i-1} \subseteq gfp_\sigma(X_{i-1})(x_i) = X_i$ for $1 \le i \le n$. Together with $X_0 = X \cup \{x\}$, it follows $X_n\backslash X = (X_n\backslash X_{n-1}) \cup \cdots \cup (X_1\backslash X_0) \cup \{x\}$. Hence

$$\begin{aligned}
&\sigma(gfp_\sigma(X)(x)\backslash X)\\
&\quad = \sigma(X_n\backslash X)\\
&\quad = \sigma(X_n\backslash X_{n-1}) \cup \cdots \cup \sigma(X_1\backslash X_0) \cup \sigma(x)\\
&\quad \subseteq X_n \cup \cdots \cup X_1 \cup \sigma(x)\\
&\quad = X_n = gfp_\sigma(X)(x)
\end{aligned}$$

∎

Lemma 4 *If $gfp_\sigma(X)(x)$ and $gfp_\sigma(Y)(y)$ terminate and $Y \cup \{y\} \subseteq gfp_\sigma(X)(x)$ with $X \subseteq Y$, then $gfp_\sigma(Y)(y) \subseteq gfp_\sigma(X)(x)$.*

Proof. Induction on the run of $gfp_\sigma(Y)(y)$. If $y \in Y$ the result is obvious. Suppose $y \notin Y$. Then, since $X \subseteq Y$, $y \notin X$ also holds and hence, by assumption, we have $y \in gfp_\sigma(X)(x)\backslash X$. Lemma 3 yields $\sigma(y) \subseteq gfp_\sigma(X)(x)$.

Let $\sigma(y) = \{y_1, \ldots, y_n\}$, $Y_0 = Y \cup \{y\}$ and $Y_i = gfp_\sigma(Y_{i-1})(y_i)$ for $1 \le i \le n$. We show the claim $gfp_\sigma(Y_{i-1})(y_i) \subseteq gfp_\sigma(X)(x)$ for $1 \le i \le n$, which derives the desired result since $gfp_\sigma(Y)(y) = gfp_\sigma(Y_{n-1})(y_n)$. The claim is proved by induction on i. If $i = 1$ then we infer $Y_0 \cup \{y_1\} \subseteq gfp_\sigma(X)(x)$ and $X \subseteq Y_0$. Then, by the first induction hypothesis, we have $gfp_\sigma(Y_0)(y_1) \subseteq gfp_\sigma(X)(x)$. Now we show $gfp_\sigma(Y_i)(y_{i+1}) \subseteq gfp_\sigma(X)(x)$ for $i > 0$. From Lemma 2 we have $Y_j \subseteq gfp_\sigma(Y_j)(y_{j+1}) = Y_{j+1}$ for $0 \le j \le i-1$. Hence $X \subseteq Y_i$. From the second induction hypothesis, we obtain $Y_i = gfp_\sigma(Y_{i-1})(y_i) \subseteq gfp_\sigma(X)(x)$. Then the first induction hypothesis yields $gfp_\sigma(Y_i)(y_{i+1}) \subseteq gfp_\sigma(X)(x)$. ∎

Lemma 5 *If $x \notin X$ and $gfp_\sigma(X)(x)$ terminates, $\Gamma_\gamma(gfp_\sigma(X)(x)) \sqsubseteq$*

$\Phi_{\sigma,\gamma}(\Gamma_\gamma \circ gfp_\sigma(X))(x)$.

Proof. Let $\sigma(x) = \{x_1, \ldots, x_n\}$ with $n \geq 0$. By definition $\Phi_{\sigma,\gamma}(\Gamma_\gamma \circ gfp_\sigma(X))(x) = \sqcap\{\Gamma_\gamma(gfp_\sigma(X)(x_i)) \mid 1 \leq i \leq n\} \sqcap \gamma(x)$. From Lemma 2 we have $x \in gfp_\sigma(X)(x)$ and hence $\Gamma_\gamma(gfp_\sigma(X)(x)) \sqsubseteq \gamma(x)$. The rest we have to show is $\Gamma_\gamma(gfp_\sigma(X)(x)) \sqsubseteq \Gamma_\gamma(gfp_\sigma(X)(x_i))$ for every $1 \leq i \leq n$. From Lemma 2 we have $X \cup \sigma(x) \subseteq gfp_\sigma(X)(x)$. Hence Lemma 4 yields $gfp_\sigma(X)(x_i) \subseteq gfp_\sigma(X)(x)$, which results in $\Gamma_\gamma(gfp_\sigma(X)(x)) \sqsubseteq \Gamma_\gamma(gfp_\sigma(X)(x_i))$. ▮

Lemma 6 *If $gfp_\sigma(X)(x)$ terminates then for any fixed point f of $\Phi_{\sigma,\gamma}$ we have $\Gamma_\gamma(X) \sqcap f(x) \sqsubseteq \Gamma_\gamma(gfp_\sigma(X)(x))$.*

Proof. Induction on the run of $gfp_\sigma(X)(x)$. If $x \in X$ then the result follows because $gfp_\sigma(X)(x) = X$. Suppose $x \notin X$. Since f is a fixed point of $\Phi_{\sigma,\gamma}$, we have $f(x) = \Phi_{\sigma,\gamma}(f)(x) = \gamma(x) \sqcap f(x_1) \sqcap \cdots \sqcap f(x_n)$, where $\sigma(x) = \{x_1, \ldots, x_n\}$ $(n \geq 0)$. Now we show the following claim by induction on i:

$\Gamma_\gamma(X) \sqcap \gamma(x) \sqcap f(x_1) \sqcap \cdots \sqcap f(x_i) \sqsubseteq \Gamma_\gamma(X_i)$ for any $0 \leq i \leq n$.

Since $X_0 = X \cup \{x\}$, we have $\Gamma_\gamma(X) \sqcap \gamma(x) \sqsubseteq \Gamma_\gamma(X_0)$. We turn to the induction step. From the second induction hypothesis we obtain $\Gamma_\gamma(X) \sqcap \gamma(x) \sqcap f(x_1) \sqcap \cdots \sqcap f(x_{i+1}) \sqsubseteq \Gamma_\gamma(X_i) \sqcap f(x_{i+1})$. The first induction hypothesis yields $\Gamma_\gamma(X_i) \sqcap f(x_{i+1}) \sqsubseteq \Gamma_\gamma(gfp_\sigma(X_i)(x_{i+1})) = \Gamma_\gamma(X_{i+1})$. ▮

Now we are ready to show the main result in this section.

Theorem 5 *If $gfp_\sigma(\emptyset)$ is totally defined then $\nu\Phi_{\sigma,\gamma} = \Gamma_\gamma \circ gfp_\sigma(\emptyset)$.*

Proof. We first show $\Gamma_\gamma \circ gfp_\sigma(\emptyset) \sqsubseteq \nu\Phi_{\sigma,\gamma}$. By Proposition 1 it is enough to show the $\Phi_{\sigma,\gamma}$-consistency of $\Gamma_\gamma \circ gfp_\sigma(\emptyset)$, i.e., $\Gamma_\gamma(gfp_\sigma(\emptyset)(x)) \sqsubseteq \Phi_{\sigma,\gamma}(\Gamma_\gamma \circ gfp_\sigma(\emptyset))(x)$ for any x in A, which immediately follows from Lemma 5 because $x \notin \emptyset$. Next we show that $\nu\Phi_{\sigma,\gamma}(x) \sqsubseteq \Gamma_\gamma(gfp_\sigma(\emptyset)(x))$. It is immediately obtained from Lemma 6 by taking $A = \emptyset$ and $f = \nu\Phi_{\sigma,\gamma}$. Therefore the result follows. ▮

The above theorem assumes that $gfp_\sigma(\emptyset)$ is totally defined, i.e., $gfp_\sigma(\emptyset)(x)$ terminates for every $x \in A$. We give a sufficient condition for total definedness of $gfp_\sigma(\emptyset)$, which is similar to that by Gapeyev, et al.[8].

Definition 6 *The function* reach_σ *of type $fin(A) \rightarrow fin(A)$ is defined as follows:*

$$\mathrm{reach}_\sigma(X) = \bigcup_{n \geq 0} \sigma^n(X).$$

Note that $X \subseteq \mathrm{reach}_\sigma(X)$. Considering a graph induced from σ illustrated in Example 1, the set $\mathrm{reach}_\sigma(X)$ contains the reachable nodes from the nodes in X.

Lemma 7 $\mathrm{reach}_\sigma(X) \subseteq \mathrm{reach}_\sigma(Y)$ *if $X \subseteq \mathrm{reach}_\sigma(Y)$.*

Proof. Obvious from definition. ▮

Proposition 2 *If $\mathrm{reach}_\sigma(X \cup \{x\})$ is finite then $gfp_\sigma(X)(x)$ terminates and the relation $gfp_\sigma(X)(x) \subseteq \mathrm{reach}_\sigma(X \cup \{x\})$ holds.*

Proof. By induction on the cardinality of $\mathrm{reach}_\sigma(X \cup \{x\}) \backslash X$. We only have to show the case $\mathrm{reach}_\sigma(X \cup \{x\}) \backslash X$ is nonempty and $x \notin X$. Let $\sigma(x) = \{x_1, \ldots, x_n\}$. We inductively define sets $X_0, \ldots, X_n$ in $fin(A)$ as follows: $X_0 = X \cup \{x\}$; $X_{i+1} = gfp_\sigma(X_i)(x_{i+1})$ if $gfp_\sigma(X_i)(x_{i+1})$ terminates, $\emptyset$ otherwise. Now we prove the following claim by induction on i: $gfp_\sigma(X_{i-1})(x_i)$ is terminating and $gfp_\sigma(X_{i-1})(x_i) \subseteq \mathrm{reach}_\sigma(X_0 \cup \{x_1, \ldots, x_i\})$ for each $1 \leq i \leq n$. The finiteness of $\mathrm{reach}_\sigma(X \cup \{x\})$ implies that $\mathrm{reach}_\sigma(X_0 \cup \{x_1, \ldots, x_i\})$ is finite. Since $X_0 = X \cup \{x\}$ and $x_1 \in \mathrm{reach}_\sigma(x)$, Lemma 7 implies $\mathrm{reach}_\sigma(X_0 \cup \{x_1\}) \subseteq \mathrm{reach}_\sigma(X \cup \{x\})$. Hence $\mathrm{reach}_\sigma(X_0 \cup \{x_1\})$ is also finite. By the first induction hypothesis $gfp_\sigma(X_0)(x_1)$ terminates and $gfp_\sigma(X_0)(x_1) \subseteq \mathrm{reach}_\sigma(X_0 \cup \{x_1\})$. We show $gfp_\sigma(X_i)(x_{i+1})$ terminates and $gfp_\sigma(X_i)(x_{i+1}) \subseteq \mathrm{reach}_\sigma(X_0 \cup \{x_1, \ldots, x_{i+1}\})$. Since $gfp_\sigma(X_{i-1})(x_i)$ terminates from the second induction hypothesis, $X_i = gfp_\sigma(X_{i-1})(x_i)$ and hence $X_i \cup \{x_{i+1}\} \subseteq \mathrm{reach}_\sigma(X_0 \cup \{x_1, \ldots, x_{i+1}\})$. Lemma 7 yields $\mathrm{reach}_\sigma(X_i \cup \{x_{i+1}\}) \subseteq \mathrm{reach}_\sigma(X_0 \cup \{x_1, \ldots, x_{i+1}\})$, thus $\mathrm{reach}_\sigma(X_i \cup \{x_{i+1}\})$ is finite. Since the cardinality of $\mathrm{reach}_\sigma(X_{i+1} \cup \{x_{i+1}\}) \backslash X_{i+1}$ is less than that of $\mathrm{reach}_\sigma(X \cup \{x\}) \backslash X$, we can infer the claim holds from the first induction hypothesis. From the above claim we obtain $gfp_\sigma(X)(x) = gfp_\sigma(X_{n-1})(x_n) \subseteq \mathrm{reach}_\sigma(X_0 \cup \{x_1, \ldots, x_n\}) \subseteq \mathrm{reach}_\sigma(X \cup \{x\})$. ▮

Now we obtain a sufficient condition for the total definedness of gfp_σ as a corollary of the above proposition.

Corollary 1 *$gfp_\sigma(\emptyset)$ is totally defined if $\mathrm{reach}_\sigma(\{x\})$ is finite for every $x \in A$.*

5. Termination

In this section we explain that the function $\mathcal{D}$ is considered as an instance of $\Phi_{\sigma,\gamma}$ and that the instantiated $gfp_\sigma(\emptyset)$ is totally defined. The finiteness proof follows the method proposed by Brandt and Henglein[4], which is also used by Gapeyev, et al.[8]: First, we define a *quasi-order* $\preceq$, i.e., a reflexive and transitive relation, that satisfies $x \in \mathrm{reach}(y)$ if $x \preceq y$. Then we show that $\{y \mid y \preceq x\}$ is finite for any x.

The instantiation of σ is based on the definition of $\mathcal{D}$, the first element of its argument should be of the form either 0, 1, aR, $(P+Q)R$ or P^*R, S. This leads us to the introduction of a restricted form of regular expressions called *canonical form*.

Definition 7 *A* canonical form *is a regular expression of the form either 0, 1, aP, $(P+Q)R$, or P^*Q where P, Q, R are canonical forms except 0. A canonical form is called* proper *if it is not 0. Regular expressions of the form either a, $R+S$ or R^*, where R and S proper canonical forms, are called* atom*s. The set of canonical forms are denoted by CF.*

We denote the syntactic equivalence between canonical forms by $\equiv$. Canonical forms are considered as *list* representation of regular expressions: 1 corresponds to the empty list and the construction of canonical forms given above is regarded as *cons* of atoms to lists, which represent canonical forms.

Note that the application of $\cdot$, $+$ or * to the canonical forms does not yield canonical forms. We define the canonical form of regular expressions obtained by the application of these operations to the canonical forms as follows.

Definition 8 *The* canonical form of T, *denoted by* $T\downarrow$, *where T is of the form either VW, $V+W$ or V^* for canonical forms V and W, is defined as follows.*

$(0S)\downarrow \equiv (S0)\downarrow \equiv 0$

$(1S)\downarrow \equiv (S1)\downarrow \equiv (0+S)\downarrow \equiv (S+0)\downarrow \equiv S$

$(0^*)\downarrow \equiv 1$

$(P^*)\downarrow \equiv P^*1$

$(P+Q)\downarrow \equiv (P+Q)1$

$((AP)Q)\downarrow \equiv A(PQ)\downarrow$

Here, A denotes an atom, S a canonical form and P and Q proper canonical forms.

It is easy to see that $T\downarrow$ really stands for a unique canonical form. From the viewpoint of *list* interpretation of canonical forms, the operation $\downarrow$ is mostly considered as an *append* of two lists or a *cons* of atoms to the empty list. Thus, this operation is easily implemented and its complexity is at most linear to the length of top-level lists.

The following proposition ensures that we restrict ourselves to the canonical forms, instead of arbitrary regular expressions.

Proposition 3 *For any regular expression R, there exists a canonical form S such that $R = S$.*

Proof. Structural induction on R. The cases of $R \equiv 0$ and $R \equiv 1$ are obvious. If $R \equiv a$ then the canonical form $a1$ represents the same set as a does. When $R \equiv R_1R_2$, by the induction hypothesis there exist canonical forms R'_i $(i = 1, 2)$ such that $R_i = R'_i$. Obviously $R = (R'_1R'_2)\downarrow$. The rest of the proof follows similarly. ∎

Thanks to the above proposition, we can restrict ourselves to canonical forms without loss of generality. Moreover, its proof gives us a method to generate a unique canonical form from a given regular expression; Given any regular expression R, we can compute *the* canonical form of R. Therefore, we hereafter deal with only the canonical forms. We denote canonical forms by capital letters from P to W and atoms by A and B with primes or subscripts when necessary. We omit $\downarrow$ for readability: for instance, $P^* + Q$ and PQ^* means $(P^*\downarrow +Q)\downarrow$ and $(PQ^*\downarrow)\downarrow$, respectively. Note that although $(P+Q)R$ (or P^*Q) is already a canonical form, it can be regarded as $((P+Q)\downarrow R)\downarrow$ (or $(P^*\downarrow Q)\downarrow$) because $(P+Q)R \equiv ((P+Q)\downarrow R)\downarrow$ (or $P^*Q \equiv (P^*\downarrow Q)\downarrow$) by the definition of $\downarrow$. Moreover, the parentheses enclosing concatenations of canonical forms can be also omitted due to the following proposition.

Proposition 4 $(PQ)R \equiv P(QR)$ *for any canonical forms P, Q and R.*

Proof. If either P, Q or R is 0 then the both hand sides are also 0. Otherwise, the result is obtained by the structural induction on the canonical form P. ∎

Thus, $PQ(R+S)$ stands for either $((PQ)\downarrow (R+S)\downarrow)\downarrow$ or $(P(Q(R+S)\downarrow)\downarrow)\downarrow$, both denote the same canonical form due to Proposition 4.

Now we define instances of σ and γ.

Definition 9 *The function $\mathcal{S} : CF \times Reg/\!\sim_{(ACI+)} \to fin(CF \times Reg/\!\sim_{(ACI+)})$ is defined as follows.*

$\mathcal{S}(0, S) = \emptyset$

$\mathcal{S}(1, S) = \emptyset$

$\mathcal{S}(aR, S) = \{(R, a^{-1}S)\}$

$\mathcal{S}((P+Q)R, S) = \{(PR, S), (QR, S)\}$

```
$gfp_{\mathcal{S}}(X)(R,S) =$
  if $\exists(P,Q) \in X.P \equiv R \wedge P \sim_{(ACI+)} Q$
   then $X$
  else let $X_0 = X \cup \{(R,S)\}$
  in if $R \equiv 0$ or $R \equiv 1$ then $X_0$
  else if $R \equiv aP$ then
      $gfp_{\mathcal{S}}(X_0)(P, a^{-1}S)$
  else if $R \equiv (P_1 + P_2)Q$ then
      let $X_1 = gfp_{\mathcal{S}}(X_0)(P_1 Q, S)$
      in $gfp_{\mathcal{S}}(X_1)(P_2 Q, S)$
  else if $R \equiv P^* Q$ then
      let $X_1 = gfp_{\mathcal{S}}(X_0)(Q, S)$
      in $gfp_{\mathcal{S}}(X_1)(PP^* Q, S)$
```

Fig. 2 The function $gfp_{\mathcal{S}}$.

$$\mathcal{S}(P^*R, S) = \{(R,S), (PP^*R, S)\}$$

Here $Reg/\sim_{(ACI+)}$ is the quotient set of (syntactic) regular expressions by an equivalence relation induced from (ACI+) presented in Section 2.

We define $\mathcal{C} : CF \times Reg/\sim_{(ACI+)} \to CF$ as follows: for any S, $\mathcal{C}(R,S) = S$ if $R \equiv 1$; otherwise $\mathcal{C}(R,S) = \Sigma^*$.

Clearly, from the above definition, the function $\mathcal{D}$ introduced in Section 3 can be regarded as $\Phi_{\mathcal{S},\mathcal{C}}$ due to Proposition 3. Thus, we can formulate an algorithm generating the product derivatives by $\Gamma_{\mathcal{C}} \circ gfp_{\mathcal{S}}(\emptyset)$ provided $gfp_{\mathcal{S}}(\emptyset)$ is totally defined. The first step of our algorithm is described in **Fig. 2**, obtained by unfolding $\mathcal{S}$ in $gfp_{\mathcal{S}}$. We omit the description of the unfolded $\Gamma_{\mathcal{C}}$, which gives the second step of our algorithm. Instead, we only note that in $\Gamma_{\mathcal{C}}$ the intersection $\bigcap$ plays a role of $\sqcap$ in $\Gamma_{\mathcal{C}}$ since we take $\subseteq$ as the partial order.

Consider $(a^*)^{-1}(b^*(ab^*)^*)$ in Section 3 as an example of the computation with the above algorithm. We show a graph that represents the computation in the first step of the algorithm in **Fig. 3**. The function $gfp_{\mathcal{S}}$ returns all nodes occurring in the graph. For the leaf nodes of the graph, the function $\mathcal{C}$ returns the second element of the node and for the other nodes Σ^* are returned. Therefore we obtain the following result (Actually Σ^* occurs four times in the second line below).

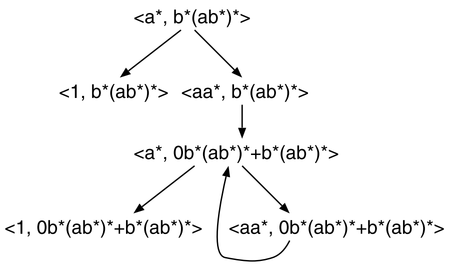

Fig. 3 A graph obtained from $(a^*)^{-1}(b^*(ab^*)^*)$.

$$\text{(ident)}\ \frac{}{S \preceq_1 S} \qquad \text{(pre)}\ \frac{P \preceq_1 Q}{P \preceq_1 RQ}\ \text{if } R \not\equiv 0 \qquad \text{(post)}\ \frac{P \preceq_1 Q}{PR \preceq_1 QR}$$

$$\text{(or1)}\ \frac{P \preceq_1 Q}{P \preceq_1 R + Q} \qquad \text{(or2)}\ \frac{P \preceq_1 Q}{P \preceq_1 Q + R} \qquad \text{(star)}\ \frac{P \preceq_1 Q}{PQ^* \preceq_1 Q^*}$$

Fig. 4 the rules for postfix relation.

$$\Gamma_{\mathcal{C}}(gfp_{\mathcal{S}}(\langle a^*, b^*(ab^*)^* \rangle)$$
$$= b^*(ab^*)^* \cap (0b^*(ab^*)^* + b^*(ab^*)^*) \cap \Sigma^*$$
$$= b^*(ab^*)^*.$$

Now we introduce a relation on the set of canonical forms.

Definition 10 *The* postfix *relation $\preceq_1$ on CF is defined from the set of rules described in* **Fig. 4**.

We say the sequence of formulas of the form $P \preceq_1 Q$ ($\preceq_1$-formulas) obtained by the repetitive applications of the above rules starting from (ident) *is the ($\preceq_1$)-derivation. The Adjacent pairs in a ($\preceq_1$)-derivation is called the steps in π. The last $\preceq_1$-formula of a $\preceq_1$-derivation π is called the conclusion of π, denoted by $conc(\pi)$.*

The relation $\preceq$ is defined with the above relation $\preceq_1$.

Definition 11 *The relation $\preceq_2$ on $Reg/\sim_{(ACI+)}$ is defined as $w^{-1}S \preceq_2 S$ for any sequence $w \in \Sigma^*$ and any canonical form S. The relation $\preceq$ on $CF \times Reg/\sim_{(ACI+)}$ is defined as follows: $(P,Q) \preceq (R,S)$ iff $P \preceq_1 R$ and $Q \preceq_2 S$.*

Together with the definition of $\mathcal{S}$, Theorem 1 reveals that $Q \preceq_2 S$ if $(P,Q) \in \text{reach}_{\mathcal{S}}(R,S)$. Moreover, from Theorem 2 the set $\{R \mid R \preceq_2 S\}$ is finite for any

$S \in Reg/{\sim_{(ACI+)}}$.

Thus, we only have to show that $(P, Q) \in \mathrm{reach}_{\mathcal{S}}(R, S)$ implies $P \preceq_1 R$ and that $\{R \mid R \preceq_1 S\}$ is finite for any $S \in CF$. The first result is a consequence of the following two statements.

Lemma 8 *If $(R, S) \in \mathcal{S}(P, Q)$ then $R \preceq_1 P$.*

Proof. We distinguish the following cases according to the structure of the first argument of $\mathcal{S}$.

aP**:** We show $P \preceq_1 aP$. From (ident) we have $P \preceq_1 P$. Applying (pre) to it, we obtain $P \preceq_1 aP$.

$(P+Q)R$**:** we obtain $PR \preceq_1 (P+Q)R$ by applying (or1) and (post) to $P \preceq_1 P$ in turn. The relation $QR \preceq_1 (P+Q)R$ is obtained likewise.

(P^*Q)**:** We have to show $Q \preceq_1 P^*Q$ and $PP^*Q \preceq_1 P^*Q$. The former is obtained from (pre) to $Q \preceq_1 Q$ since $P^* \not\equiv 0$. The application of (star) to $P \preceq_1 P$ yields $PP^* \preceq_1 P^*$, from which we obtain $PP^*Q \preceq_1 P^*Q$ by (post). ∎

The proof of the following proposition is found in Appendix A.1

Proposition 5 *If $P \preceq_1 Q$ and $Q \preceq_1 R$ then $P \preceq_1 R$.*

The following proposition concludes the first part of the proof.

Proposition 6 *If $(P, Q) \in \mathrm{reach}_{\mathcal{S}}(R, S)$ then $(P, Q) \preceq (R, S)$.*

Proof. An immediate consequence of Lemma 8 and Proposition 5. ∎

We turn to the second part of the proof.

Lemma 9 *The following property holds.*

(1) If $S \preceq_1 0$ then $S \equiv 0$.

(2) If $S \preceq_1 1$ then either $S \equiv 0$ or $S \equiv 1$.

(3) $0 \preceq_1 S$ for any canonical form S.

Proof. (1) and (2) are proved by induction on the structure of derivations. (3) is derived from $0 \preceq_1 0$ by applying (or1). ∎

Definition 12 *The size of a canonical form P, denoted by $|P|$, and of an atom A, denoted by $|A|$, are mutually defined as follows.*

canonical forms

- $|0| = |1| = 0$
- $|AP| = |A| + |P|$

atom

- $|a| = 1$
- $|P + Q| = |P| + |Q|$
- $|P^*| = |P| + 1$

Lemma 10 *If P and Q are proper canonical forms then $|PQ| = |P| + |Q|$.*

Proof. Induction on the structure of P. If $P = 1$ then $PQ = P$. Thus, $|PQ| = |Q| = 0 + |Q| = |P| + |Q|$. If $P = AR$ then the induction hypothesis yields $|RQ| = |R| + |Q|$. Therefore $|(AR)Q| = |A(RQ)| = |A| + |RQ| = |A| + |R| + |Q| = |AR| + |Q|$. ∎

Lemma 11 *For any proper canonical form P the set $\{(Q, R) \mid P \equiv QR\}$ is finite.*

Proof. Structural induction on P. If P is 1 then, from Definition 8, $(1, 1)$ is the only element of the above set. Suppose P is of the form AP'. If $P \equiv 1R$ then the only possible choice is $P \equiv R$. If Q is of the form BQ' then Since $(BQ')R \equiv B(Q'R)$, B should be A. By the induction hypothesis, there are a finite number of the possible choices of Q' and R satisfying $P' \equiv Q'R$, which results in the finiteness of the set $\{(Q, R) \mid P \equiv QR\}$. ∎

Definition 13 *If a derivation of $P \preceq_1 Q$ does not contain any step in which its premise and conclusion are the same, then it is called* non-redundant *derivation of $P \preceq_1 Q$.*

Note that we don't lose any derivable objects using the rules in Definition 10 even if we restrict ourselves to non-redundant derivations. Now we show the second part of the proof.

Proposition 7 *For any S the set $\{R \mid R \preceq_1 S\}$ is finite.*

Proof. We first define $Post(S) = \{R \mid R \preceq_1 S\}$ and $Post_\alpha(S) = \{R \mid \exists\pi \in \Pi_\alpha \text{ and } conc(\pi) = R \preceq_1 S\}$, where Π_α stands for the set of non-redundant derivations whose last steps is by a rule (α).

We proof the result by induction on $|S|$. The case either $S \equiv 0$ or $S \equiv 1$ is an immediate consequence of Lemma 9. Suppose $S \equiv AT$. It is enough to show that for each rule (α) in Definition 10 the set $Post_\alpha(AT)$ is finite. We distinguish the following cases according to α.

(ident) Obvious.

(pre) Let $AT \equiv PQ$ and consider the non-redundant derivations that the left-hand side of the premise in the last step is Q. Then $\bigcup_{AT \equiv PQ} Post(Q) = Post_{pre}(AT)$. Since the derivations we consider are non-redundant, $P \not\equiv 1$. It follows $|Q| < |P| + |Q| = |AT|$ from Lemma 10. Hence, by the induction hypothesis, $Post(Q)$ is finite. From Lemma 11 there are finite Q's satisfying $AT \equiv PQ$. Therefore $Post(AT)$ is finite.

(post) Let $AT \equiv PQ$ and consider the non-redundant derivations that the left-hand side of the premise in the last step is P. Then $\bigcup_{AT \equiv PQ}\{UQ \mid U \in Post(P)\} = Post_{post}(AT)$. Since the derivations we consider are non-redundant, $Q \not\equiv 1$. It follows $|P| < |P| + |Q| = |AT|$ from Lemma 10. Hence, by the induction hypothesis, $Post(P)$ is finite. From Lemma 11 there are finite P's and Q's satisfying $AT \equiv PQ$. Therefore $Post(AT)$ is finite.

(or1) From $AT + 0 \equiv AT$ and Lemma 9 (1), we learn $0 \in Post_{or1}(AT)$. On the other hand, since we exclude every derivation whose last step is from $R \preceq_1 AT$ to $R \preceq_1 0 + AT \equiv AT$ regardless of R, we do not need to consider such R's. Only when A is of the form $P+Q$, the set $Post_{or1}(AT)$ has the other elements, which are from $Post(Q)$. Since P is proper, $|Q| < |P| + |Q| + |T| = |AT|$. Hence, from the induction hypothesis, $Post(Q)$ is finite and the result follows.

(or2) Similar to the case of (or1).

(star) The only possible case is that A is of the form P^*, $T \equiv 1$ and the right-hand side of the premise of the last step is P. Then $Post_{star}(AT) = \{QP^* \mid Q \in Post(P)\}$. Since P is proper, $|P| < |P| + 1 + |T| = |AT|$. Hence, from the induction hypothesis, $Post(P)$ is finite and the result follows. ∎

Theorem 6 reach$_{\mathcal{S}}(x)$ *is finite for any* $x \in CF \times Reg/\sim_{(ACI+)}$ *so that* $gfp_{\mathcal{S}}(\emptyset)$ *is totally defined.*

Proof. Immediate consequence of Propositions 6 and 7 and Theorem 2. ∎

6. Conclusion

In this paper we have proposed a product derivative as an intersection of Brzozowski's derivatives. We have given an algorithm generating the product derivatives as $\Gamma_{\mathcal{C}} \circ gfp_{\mathcal{S}}(\emptyset)$. Complexity estimation of the algorithm is a further research.

In the motivating example in Section 1, the product derivatives are computed repetitively. One may think that the presence of intersection in $\Gamma_{\mathcal{C}}$ inhibits the repetitive computation of product derivatives unless the intersection of regular expressions are computed before further application of the algorithm to product derivatives computed. It is not the case because regular expressions with intersection only appear as the second element of the pairs to which the algorithm is applied; they simply become (a part of) product derivatives obtained, or transformed by the application of Brzozowski's derivative, which has a rule for intersection: $a^{-1}(P \cap Q) = a^{-1}P \cap a^{-1}Q$. Therefore we can compute the product derivatives repetitively without the computation of intersection of regular expressions.

Of course, the product derivatives produce redundant intersections such as $\Sigma^* \cap R$ and $0 \cap R$, which can be transformed into R and 0, respectively. The immediate simplification of the redundant intersections greatly improves the efficiency of the algorithm. Moreover, $R \cap S$ may be equal to 0 (in the set interpretation) even if neither R nor S are not 0. Detection of such a intersection is also a further research.

We note that the proposed algorithm can be used for checking inclusion of a regular expression by another: R is included in S if $R^{-1}S$ is nullable. Hosoya, et al. have proposed an inclusion checking algorithm based on the computation of greatest fixed points [10]. Since inclusion checking is an application of our algorithm, our algorithm is considered as a generalization of their algorithm.

Finally, we remark that this work is a first step for our purpose. As demonstrated in Section 1, the proposed notion in this paper is useful for transformation of XML documents with links to Web services in some cases. Because XML documents is represented as trees, however, we should consider the tree version of product derivatives in general case, which is our further research.

References

1) Abiteboul, S., et al.: Active XML: A Data-Centric Perspective on Web Services, *BDA 2002 Proceedings* (2002).

2) Antimirov, V.M. and Mosses, P.D.: Rewriting extended regular expressions, *Theoretical Computer Science*, Vol.143, No.1, pp.51–72 (1995).

3) Antimirov, V.M.: Partial Derivatives of Regular Expressions and Finite Automata

Constructions, *Theoretical Computer Science*, Vol.155, No.2, pp.291–319 (1996).

4) Brandt, M. and Henglein, F.: Coinductive axiomatization of recursive type equality and subtyping, *Fundamenta Informaticae*, Vol.33, pp.309–338 (1998).

5) Brzozowski, J.A.: Derivatives of regular expressions, *J. ACM*, Vol.11, pp.481–494 (1964).

6) Conway, J.H.: *Regular algebra and Finite Machines*, Chapman and Hall (1971).

7) Denis, F., Lemay, A. and Terlutte, A.: Residual finite state automata, *Fundamenta Informaticae*, Vol.51, No.4, pp.339–368 (2002).

8) Gapeyev, V., Levin, M.Y. and Pierce, B.C.: Recursive Subtyping Revealed, *J. Functional Programming*, Vol.12, No.6, pp.511–548 (2002).

9) Ginzburg, A.: A procedure for checking equality of regular expressions, *J. ACM*, Vol.14, No.2, pp.355–362 (1967).

10) Hosoya, H., Vouillon, J. and Pierce, B.C.: Regular Expression Types for XML, *ACM Trans. Programming Languages and Systems* (2004).

11) Krob, D.: Differentiation of K-rational expressions, *International Journal of Algebra, and Computation*, Vol.2, No.1, pp.57–87 (1992).

12) Okui, S. and Suzuki, T: Pattern Matching of Incompletely RE-Typed Expressions via Transformation, *IPSJ Transactions on Programming*, Vol.47, SIG 6 (PRO 29), pp.37–49 (2006).

13) Sperberg-McQueen, C.M.: Applications of Brzozowski derivatives to XML Schema processing, *Extreme Markup Languages 2005* (2005).

14) Suzuki, T. and Okui, S.: A rewrite system with incomplete regular expression type for transformation of XML documents, *IPSJ Transactions on Programming*, Vol.46, SIG 14 (PRO27), pp.43–54 (2005).

15) Tarski, A.: A Lattice-Theoretical Fixpoint Theorem and its Applications, *Pacific Journal of Mathematics*, Vol.5, pp.285–309 (1955).

Appendix

A.1 The proof of Proposition 5

Lemma 12 *If $R \preceq_1 TQ$ then either $R \preceq_1 Q$ holds or there exists a canonical form R' such that $R \equiv R'Q$ and $R' \preceq_1 T$.*

Proof. The result follows from Lemma 9 (3) when $R \equiv 0$. The case where either T or Q is 0 is reduced to the previous case because Lemma 9 (1) yields $R \equiv 0$. Suppose neither R, T nor Q is 0. We use structural induction on the derivations.

(ident) Since R is written TQ, the result holds by taking $R' \equiv T$.

(pre) Suppose that from $R \preceq_1 S$ we derive $R \preceq_1 PS$ using (pre), where $PS \equiv TQ$ (Note that generally $P \not\equiv T$ and $S \not\equiv Q$). Since $TQ \neq 0$, neither P nor S are 0. We distinguish the following two cases:

(a) Suppose there exists Q' satisfying $Q \equiv Q'S$ and $P \equiv TQ'$. Since Q is not 0, neither is Q'. Hence (pre) is applicable to $R \preceq_1 S$ with Q' and the result follows.

(b) Suppose there exists T' with $T \equiv PT'$ and $S \equiv T'Q$. By the induction hypothesis either $R \preceq_1 Q$ holds or there exists an R' with $R \equiv R'Q$ and $R' \preceq_1 T'$. In the former case we are done. In the latter case, the application of (post) to $R' \preceq_1 T'$ yields $R \equiv R'Q \preceq_1 T'Q \equiv T$.

(post) Suppose that $R \equiv UP \preceq_1 SP \equiv TQ$ is derived from $U \preceq_1 S$ with (post). Neither P nor S are 0 since $TQ \not\equiv 0$. As in the previous case we consider two cases.

(a) Suppose there exists Q' satisfying $Q \equiv Q'P$ and $S \equiv TQ'$. By the induction hypothesis either $U \preceq_1 Q'$ holds or $U \equiv R'Q$ and $R' \preceq_1 T'$ for some R'. In the former case $UP \preceq_1 Q'P \equiv Q$ is obtained from $U \preceq_1 Q'$ by (post). In the latter case the result follows from $R \equiv UP \equiv R'Q'P \equiv R'Q$.

(b) Suppose there exists T' with $T \equiv ST'$ and $P \equiv T'Q$. Let $R' \equiv UT'$. Then $R \equiv UP \equiv UT'Q \equiv R'T$. Applying (post) to $U \preceq_1 S$ we obtain $R' \equiv UT' \preceq_1 ST' \equiv T$.

(or1) Suppose $R \preceq_1 P+S \equiv TQ$ is derived from $R \preceq_1 S$ by (or1). When $P = 0$ then $S \equiv TQ$ and hence the result immediately follows from the induction hypothesis. Since $S \not\equiv 0$ from Lemma 9 (1) and $R \not\equiv 0$, there exists two cases: $T \equiv P + S$ and $Q \equiv 1$, or $T \equiv 1$ and $Q \equiv P + S$. In the former case the result follows from $R \equiv R1 \equiv RQ$ and $R \preceq_1 P + S \equiv T$. In the latter one, $R \preceq_1 P + S \equiv Q$ holds.

(or2) Similar to (or1).

(star) Suppose $R \equiv US^* \preceq_1 S^* \equiv TQ$ is derived from $U \preceq_1 S$ by (star). Since $R \not\equiv 0$, Lemma 9 (1) yields $S \not\equiv 0$. The only possible cases are $T \equiv S^*$ and $Q \equiv 1$, or $T \equiv 1$ and $Q \equiv S^*$. The rest of the proof follows that of the case (or1).

∎

Lemma 13 *If $P \preceq_1 QR^*$ then either $P \equiv 1$ or $P \equiv SR^*$ with some S.*

Proof. If $P \equiv 0$ then take $S \equiv 0$. If $Q \equiv 0$ then $P \equiv 0$ from Lemma 9 (1) and hence the result follows. It $R \equiv 0$ then by taking $S \equiv P$ we have $PR^* \equiv P1 \equiv P$. Suppose neither P, Q nor R is 0. We prove the result by structural induction on the derivation of $P \preceq_1 QR^*$.

(ident) Since $P \equiv QR^*$ the desired result is obtained by taking $S \equiv Q$.

(pre) Suppose $P \preceq_1 QR^*$ is obtained from $P \preceq_1 1$ by (pre). Lemma 9 (2) and $P \equiv 0$ results in $P \equiv 1$. Let $Q \equiv Q_1Q_2$ and suppose by applying (pre) to $P \preceq_1 Q_2R^*$ we obtain $P \preceq_1 Q_1Q_2R^* \equiv QR^*$. Then the result follows from $Q_2 \not\equiv 0$ and the induction hypothesis.

(post) Suppose that the application of (post) to $T \preceq_1 1$ yields $P \equiv TQR^* \preceq_1 QR^*$. Then Lemma 9 (2) and $P \not\equiv 0$ implies $T \equiv 1$ and hence $P \equiv QR^*$. Let $Q \equiv Q_1Q_2$ and suppose by applying (post) to $T \preceq_1 Q_1$ we have $P = TQ_2R^* \preceq_1 QR^*$. Then the desired result is obtained by taking $S \equiv TQ_2$.

(or1) Let $T_1 + T_2 \equiv QR^*$ and suppose $P \preceq_1 T_1 + T_2 \equiv QR^*$ is derived from $P \preceq_1 T_2$ by (or1). If $T_2 \equiv 0$ then Lemma 9 (1) yields $P \equiv 0$, which contradicts the assumption. Thus $T_2 \not\equiv 0$. If neither T_1 nor T_2 is 0, then $T_1 + T_2 \equiv (T_1 + T_2)1 \equiv QR^*$ implies $R \equiv 0$, which also contradicts the assumption. Hence T_1 should be 0 and hence $T_2 = QR^*$. Then the result follows from the induction hypothesis.

(or2) Similar to (or1).

(star) Since $R \not\equiv 0$, we have $Q \equiv 1$. Hence in the last step of the derivation $S \preceq_1 R$ yields $P \equiv SR^* \preceq_1 R^* \equiv R^*Q$ by (star).

∎

Lemma 14 *If $PR^* \preceq_1 QR^*$ then either $P \preceq_1 Q$ or $PR^* \preceq_1 R^*$ holds.*

Proof. $P \preceq_1 Q$ is obtained from Lemma 9 (3) when $P \equiv 0$ and from Lemma 9 (1) when $Q \equiv 0$, respectively. If $R \equiv 0$ then the result follows from $R^* \equiv 1$. Suppose neither P, Q nor R is 0. We use structural induction on the derivation of $PR^* \preceq_1 QR^*$.

(ident) Since $P \equiv Q$, it follows $P \preceq_1 Q$.

(pre) Suppose $PR^* \preceq_1 QR^*$ is derived from $PR^* \preceq_1 1$ by (pre). From 9 (2), however, either P or R is 0, which contradicts the assumption. Next we assume $PR^* \preceq_1 QR^*$ is derived from $PR^* \preceq_1 Q_2R^*$ by (pre), where $Q \equiv Q_1Q_2$. Since Q_2 should not be 0, the induction hypothesis is applicable and hence either $P \preceq_1 Q_2$ or $PR^* \preceq_1 R^*$ holds. In the latter case we are done. In the former case, since Q_1 cannot be 0, the application of (pre) to $P \preceq_1 Q_2$ yields the desired result.

(post) Suppose $PR^* \preceq_1 QR^*$ is derived from $T \preceq_1 1$ by (post), where $P \equiv TQ$. Then $T \equiv 1$ from $TQR^* \not\equiv 0$ and Lemma 9 (2) and hence $P \equiv Q$. It follows $P \preceq_1 Q$. Next we assume $PR^* \preceq_1 QR^*$ is derived from $T \preceq_1 Q_1$ by (post), where $Q \equiv Q_1Q_2$. Then $P \equiv TQ_2$ and we obtain $P \equiv TQ_2 \preceq_1 Q_1Q_2 \equiv Q$ by the application of (post) to $T \preceq_1 Q_1$.

(or1,or2) Similar to the the case (or1) in the proof of Lemma 13.

(star) Since $R \not\equiv 0$, to obtain $PR^* \preceq_1 QR^*$ by (star), Q should be 1. Therefore $PR^* \preceq_1 R^*$.

∎

Now we are ready to prove the transitivity of $\preceq_1$.

Proof. [Proof of Proposition 5] We use induction on the lexicographic product that consists of $|R|$ and the height of the derivation of $Q \preceq_1 R$ and that of $P \preceq_1 Q$.

The base case ($P \equiv Q \equiv R \equiv 0$ or $P \equiv Q \equiv R \equiv 1$) is obvious. In the induction step we distinguish the following cases according to the derivation rule applied to the last step of the derivation of $Q \preceq_1 R$. Note that $R \not\equiv 0$ in the induction step; otherwise Lemma 9 (1) yields $P \equiv Q \equiv 0$, which has been already considered in the base case. The case $P \preceq_1 Q$ is derived by (ident) is obvious. Next we assume the rule used in the last step is neither (post) nor (star). Then the relation to which the rule is applied is written $Q \preceq_1 S$ using some S. Since $R \not\equiv 0$, the definition of $\preceq_1$ implies $|S| \leq |R|$. Hence by the induction hypothesis we obtain $P \preceq_1 S$, to which the application of the same rule as obtained $Q \preceq_1 R$ yields $P \preceq_1 R$.

Next we consider the case $Q \preceq_1 R$ is obtained from $S \preceq_1 T$ by (post), where $Q \equiv SU$ and $R \equiv TU$. If either S, T or U is 0 the result is obvious from Lemma 9 (1). Otherwise, we distinguish the following cases according to the last step used to obtain $P \preceq_1 Q$.

(ident) Obvious.

(pre) Suppose $P \preceq_1 SU$ is obtained from $P \preceq_1 V$, where $SU \equiv WV$. We consider the following two cases.

(a) Suppose $V \equiv V'U$ and $S \equiv WV'$ for some V'. Note that neither W and

V' is 0 since $S \not\equiv 0$. From $P \preceq_1 V'U$ and Lemma 12, either $P \preceq_1 U$ holds or $P \equiv P'U$ and $P' \preceq_1 V'$ holds for some P'. In the former case, since $T \not\equiv 0$ the application of (pre) to $P \preceq_1 U$ yields $P \preceq_1 TU \equiv R$. Consider the latter case. Since $w \not\equiv 0$, applying (pre) to $P' \preceq_1 V'$ we obtain $P' \preceq_1 WV' \equiv S$. Since $U \not\equiv 0$ Lemma 10 yields $|T| \leq |TU|$ and hence from the induction hypothesis $P' \preceq_1 T$. Applying (post) we obtain $P \equiv P'U \preceq_1 TU \equiv R$.

(b) Suppose $W = SW'$ and $U = W'V$ for some W'. Then since $SU \not\equiv 0$, we have $W' \not\equiv 0$. Therefore, applying (pre) twice to $P \preceq_1 V$ with W' and T in turn, the desired result is obtained.

(post) Suppose $P \preceq_1 SU$ is obtained from $P' \preceq_1 W$ by (post), where $P \equiv P'V$ and $SU \equiv WV$. We consider the two cases.

(a) Suppose $V \equiv V'U$ and $S \equiv WV'$ for some V'. Applying (post) to $P' \preceq_1 W$, we have $P'V' \preceq_1 WV' \equiv S$. Since $U \not\equiv 0$ Lemma 10 yields $|T| \leq |TU|$ and hence from the induction hypothesis $P'V' \preceq_1 T$, to which the application of (post) produces $P \equiv P'V'U \preceq_1 TU \equiv R$.

(b) Suppose $W \equiv SW'$ and $U \equiv W'V$ for some W'. Applying (post) to $S \preceq_1 T$, we have $W \equiv SW' \preceq_1 TW'$. Since $U \equiv W'V$ and $U \not\equiv 0$, the size of TW' is less than or equal to that of TU. Moreover, $SU \preceq_1 TU$ and $SW' \preceq_1 TW'$ are both obtained from the application of (post) to $S \preceq_1 T$ and hence from the induction hypothesis we have $P' \preceq_1 TW'$, to which the application of (post) produces $P \equiv P'V \preceq_1 TW'V \equiv TU \equiv R$.

(or1) Suppose $P \preceq_1 SU$ is derived from $P \preceq_1 V$, where $SU \equiv W + V$. If $W \equiv 0$ then it follows $V \equiv SU$ and hence the desired result is obtained from the induction hypothesis. When $V \equiv 0$, by Lemma 9 (1) we have $P \equiv 0$ and hence Lemma 9 (3) yields the desired result. Suppose neither V nor W is 0. It follows $S \equiv W + V$ and $U \equiv 1$. Since $U \not\equiv 0$ Lemma 10 yields $|T| = |TU|$. Therefore, by the induction hypothesis, $P \preceq_1 T \equiv TU \equiv R$.

(or2) Similar to (or1).

(star) Suppose $P \preceq_1 SU$ is derived from $W \preceq_1 V^*$ by (star), where $V^* \equiv SU$ and $P \equiv WV^*$. If $V \equiv 0$ then $V^* \equiv 1$. Thus, from Lemma 9 (2), either $P \equiv 0$ or $P \equiv 1$ and the result follows. Suppose $V \not\equiv 0$. Then there are only two possible cases: $S = V^*$ and $U = 1$, or $S = 1$ and $U = V^*$. In the former case, Lemma 10 yields $|T| \leq |TU|$. From the induction hypothesis $P \preceq_1 T \equiv TU \equiv R$. In the latter case, since $T \not\equiv 0$ (pre) is applicable to $P \preceq_1 V^*$ with T and we obtain $P \preceq_1 TV^* \equiv TU \equiv R$.

Finally, we consider the case where $Q \preceq_1 R$ is derived from $S \preceq_1 T$, where $Q \equiv ST^*$ and $R \equiv T^*$. From $P \preceq_1 ST^*$, Lemma 13 assures that either $P \equiv 1$ holds or $P \equiv UT^*$ holds for some U. In the former case, $P \preceq_1 R$ follows from (ident) and (pre). In the latter case, from Lemma 14 either $U \preceq_1 S$ or $UT^* \preceq_1 T^*$. In the latter case, the result follows from $P = UT^*$ and $T^* = R$. In the former case, since $|T| \leq |T^*|$ the induction hypothesis yields $U \preceq_1 T$, to which the application of (star) deduces $P \equiv UT^* \preceq_1 T^* \equiv R$. ∎

(Received December 21, 2007)
(Accepted March 19, 2008)
(Released July 8, 2008)

(Original version of this article can be found in the IPSJ Transactions on Programming, Vol.1, No.1, pp.47–59.)

Taro Suzuki was born in 1964. He received his D.Sci. degree from the University of Tokyo in 1998. He has been engaged in research in higher-order rewriting, unification and functional-logic programming. He is a member of the IPSJ, ACM and JSSST.

Satoshi Okui was born in 1967. He received his D.Eng. degree from University of Tsukuba in 1995. He has been engaged in research in rewriting, unification, and functional-logic programming. He is a member of the IPSJ.