Partial Derivatives of Regular Expressions over Alphabet-invariant and User-defined Labels*

Stavros Konstantinidis[1], Nelma Moreira[2], João Pires[2], and Rogério Reis[2]

[1] Saint Mary's University, Halifax, Nova Scotia, Canada,
s.konstantinidis@smu.ca
[2] CMUP & DCC, Faculdade de Ciências da Universidade do Porto, Rua do Campo Alegre, 4169-007 Porto Portugal {nam,rvr}@dcc.fc.up.pt

Abstract. We are interested in regular expressions that represent word relations in an alphabet-invariant way—for example, the set of all word pairs u,v where v is a prefix of u independently of what the alphabet is. This is the second part of a recent paper on this topic which focused on labelled graphs (transducers and automata) with alphabet-invariant and user-defined labels. In this paper we study derivatives of regular expressions over labels (atomic objects) in some set B. These labels can be any strings as long as the strings represent subsets of a certain monoid. We show that one can define partial derivative labelled graphs of type B expressions, whose transition labels can be elements of another label set X as long as X and B refer to the same monoid. We also show how to use derivatives directly to decide whether a given word pair is in the relation of a regular expression over pairing specs. Set specs and pairing specs are useful label sets allowing one to express languages and relations over large alphabets in a natural and compact way.

Keywords: Alphabet-invariant expressions, regular expressions, partial derivatives, algorithms, monoids

1 Introduction

SEC:Introduction

We are interested in regular expressions whose alphabet is not of fixed cardinality, or whose alphabet is even unknown. Consider the alphabet $\Gamma = \{0, 1, \ldots, n-1\}$, where n is variable, and the 2D regular expressions[3]

EQ:rex

$$\big(0/0 + \cdots + (n-1)/(n-1)\big)^* \big(0/\mathbf{e} + \cdots + (n-1)/\mathbf{e}\big)^*, \tag{1}$$

$$\big(0/0 + \cdots + (n-1)/(n-1)\big)^* \ \big(\boldsymbol{r}_0 + \cdots + \boldsymbol{r}_{n-1}\big) \ \big(0/0 + \cdots + (n-1)/(n-1)\big)^* \tag{2}$$

where $\mathbf{e}$ represents the empty string, and each $\boldsymbol{r}_i$ is the sum of all i/j with $j \neq i$ and $i, j \in \Gamma$. The first expression has $O(n)$ symbols and represents the prefix

* Research supported by NSERC (Canada) and by FCT project UID/-MAT/00144/2013 (Portugal).

[3] These are expressions for word relations.

relation, that is, all word pairs (u, v) such that v is a prefix of u. The second regular expression has $O(n^2)$ symbols and represents all word pairs (u, v) such that the Hamming distance of u, v is 1. We want to be able to use special labels in expressions such as those in the expression below.

EQ:psrex $$(\forall/=)^* \ (\forall/\forall\neq) \ (\forall/=)^* \tag{3}$$

The label $(\forall/=)$ represents the set $\{(a, a) \mid a \in \Gamma\}$ and the label $(\forall/\forall\neq)$ represents the set $\{(a, a') \mid a, a' \in \Gamma, a \neq a'\}$ (these labels are called *pairing specs*). This expression has only a fixed number of symbols. Similarly, using these special labels, the expression (1) can be written as

EQ:psrex2 $$(\forall/=)^*(\forall/\mathbf{e})^*. \tag{4}$$

Note that the new regular expressions are *alphabet invariant* as they contain no symbol of the intended alphabet Γ.

The present paper is the continuation of the recent paper [10] on the topic of labelled graphs (e.g., automata, transducers) and regular expressions whose labels are strings such that each string represents a subset of a specific monoid. The intention is to define algorithms that *work directly* on regular expressions and graphs with special labels, without of course having to expand these labels to sets of monoid elements. Thus, for example, we would like to have an algorithm that computes whether a pair (u, v) of words is in the relation represented by either of the expressions (3) and (4). While the first paper [10] focused on labelled graphs, the present paper focuses on derivatives of regular expressions over any desirable set of labels B. An expression with special labels in this work can be considered to be a *syntactic version* of a regular expression over some monoid M in the sense of [16].

Paper Structure and Main results. The next section discusses *alphabets* Γ of non-fixed size and provides a summary of concepts from [10]. In particular, a *label set* B is a nonempty set such that each $\beta \in B$ is simply a nonempty string that represents a subset $\mathcal{I}(\beta)$ of a monoid denoted by $\mathsf{mon}\, B$. Section 3 defines the set of partial derivatives $\mathsf{PD}(\boldsymbol{r})$ of any type B regular expression $\boldsymbol{r}$, where $\mathsf{mon}\, B$ is a graded monoid. As in [1], partial derivatives are defined via the concept of linear form $\mathsf{n}(\boldsymbol{r})$ of $\boldsymbol{r}$. Here we define partial derivatives $\partial_x(\boldsymbol{r})$ of $\boldsymbol{r}$ *with respect to* $x \in X$, where X is a second label set (which could be B) such that $\mathsf{mon}\, X = \mathsf{mon}\, B$. Theorem 1 says that the set $\mathsf{PD}(\boldsymbol{r})$ of partial derivatives of $\boldsymbol{r}$ is finite. Section 4 defines the type X graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ corresponding to any given type B regular expression $\boldsymbol{r}$ and shows (Theorem 2) that $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ and $\boldsymbol{r}$ have the same behaviour. We note that the states of $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ are elements of $\mathsf{PD}(\boldsymbol{r})$ and the transitions of $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ are elements of X. Section 5 uses derivatives to decide whether a given word pair is in the relation represented by a regular expression involving pairing specs, without constructing the associated transducer (Theorem 3).

2 Terminology and Summary of Concepts from [10]

SEC:Terminology

The set of positive integers is denoted by $\mathbb{N}$. Then, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. An *alphabet space* Ω is an infinite and totally ordered set whose elements are called *symbols*. We shall assume that Ω is fixed and contains the digits $0, 1, \ldots, 9$ and the letters $\mathsf{a}, \mathsf{b}, \ldots, \mathsf{z}$, which are ordered as usual, as well as the following *special symbols:* $\forall$, $\exists$, $\nexists$, $=$, $\neq$, $/$, $\mathbf{e}$, $\oplus$, $\oslash$.

As usual we use the term *string* or *word* to refer to any finite sequence of symbols. The *empty string* is denoted by ε. Let $g \in \Omega$ and w be a string. The expression $|w|_g$ denotes the number of occurrences of g in w, and the expression $\mathsf{alph}\, w$ denotes the set $\{g \in \Omega : |w|_g > 0\}$, that is, the set of symbols that occur in w. For example, $\mathsf{alph}(1122010) = \{0, 1, 2\}$.

An *alphabet* is any finite nonempty subset of Ω. In the following definitions we shall consider alphabets Σ, Δ as well as an alphabet Γ, called the alphabet of *reference*, and we assume that Γ *contains at least two symbols and no special symbols* and that Γ is not of fixed size (it is unbounded). Let Σ, Δ be alphabets. A (binary word) *relation* of type $[\Sigma, \Delta]$ is a subset R of $\Sigma^* \times \Delta^*$.

2.1 Set Specifications and Pairing Specifications

Set specs are intended to represent nonempty subsets of the alphabet Γ. These can be used as labels in automata-type objects (labelled graphs) and regular expressions defined in subsequent sections.

Definition 1. *A* set specification*, or* set spec *for short, is any string of one of the three forms* $\forall$, $\exists w$, $\nexists w$, *where* w *is any sorted nonempty string containing no repeated symbols and no special symbols. The set of set specs is denoted by* SSP. DEF:ssets

Definition 2. *Let* Γ *be an alphabet of reference and let* F *be a set spec. We say that* F respects Γ*, if the following restrictions hold when* F *is of the form* $\exists w$ *or* $\nexists w$*: "*$w \in \Gamma^*$ *and* $0 < |w| < |\Gamma|$*." In this case, the* language $\mathcal{L}(F)$ *of* F *(*with respect *to* Γ*) is the subset of* Γ *defined as follows:* $\mathcal{L}(\forall) = \Gamma$, $\mathcal{L}(\exists w) = \mathsf{alph}\, w$, $\mathcal{L}(\nexists w) = \Gamma \setminus \mathsf{alph}\, w$. *The set of set specs that respect* Γ *is denoted as* $\mathrm{SSP}[\Gamma] = \{\alpha \in \mathrm{SSP} \mid \alpha \text{ respects } \Gamma\}$. DEF:sset:lan

Now we define expressions for describing certain finite relations that are subsets of $\big((\Gamma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\})\big) \setminus \{(\varepsilon, \varepsilon)\}$.

Definition 3. *A* pairing specification*, or* pairing spec *for short, is a string of one the five forms* $\mathbf{e}/G$, $F/\mathbf{e}$, F/G, $F/=$, $F/G{\neq}$, *where* F, G *are set specs. The set of pairing specs is denoted by* PSP. *A pairing spec is called* alphabet invariant *if it contains no set spec of the form* $\exists w, \nexists w$. *The alphabet invariant pairing specs are* $\mathbf{e}/\forall, \forall/\mathbf{e}, \forall/\forall, \forall/=, \forall/\forall{\neq}$. DEF:spairs

Definition 4. *Let* Γ *be an alphabet of reference and let* p *be a pairing spec. We say that* p respects Γ*, if any set spec occurring in* p *respects* Γ*. The set of pairing specs that respect* Γ *is denoted as* $\mathrm{PSP}[\Gamma] = \{\mathsf{p} \in \mathrm{PSP} : \mathsf{p} \text{ respects } \Gamma\}$. *The* relation $\mathcal{R}(\mathsf{p})$ *described by* p *(with respect to* Γ*) is the subset of* $\Gamma^* \times \Gamma^*$ *defined as follows.* DEF:spairs:r

$\mathcal{R}(\mathbf{e}/G) = \{(\varepsilon, y) \mid y \in \mathcal{L}(G)\}$; $\mathcal{R}(F/\mathbf{e}) = \{(x, \varepsilon) \mid x \in \mathcal{L}(F)\}$;
$\mathcal{R}(F/G) = \{(x, y) \mid x \in \mathcal{L}(F), y \in \mathcal{L}(G)\}$; $\mathcal{R}(F/=) = \{(x, x) \mid x \in \mathcal{L}(F)\}$;
$\mathcal{R}(F/G{\neq}) = \{(x, y) \mid x \in \mathcal{L}(F), y \in \mathcal{L}(G), x \neq y\}$.

2.2 Label Sets and their Monoid Behaviours

SEC:labelsets

We shall use the notation ε_M for the *neutral element* of the monoid M. If S, S' are any two subsets of M then, as usual, we define $SS' = \{mm' \mid m \in S,\ m' \in S'\}$, $S^i = S^{i-1}S$ and $S^* = \cup_{i=0}^{\infty} S^i$, where $S^0 = \{\varepsilon_M\}$ and the monoid operation is denoted by simply concatenating elements. We shall only consider *finitely generated* monoids M where each $m \in M$ has a *canonical* (string) representation $\underline{m}$. Then, we write $\underline{M} = \{\underline{m} \mid m \in M\}$.

EX:standard-examples *Example 1.* We shall consider two standard monoids. (i) The free monoid Γ^* whose neutral element is ε. The canonical representation of a nonempty word w is w itself and that of ε is $\mathbf{e}$, that is, $\underline{\varepsilon} = \mathbf{e}$. (ii) The monoid $\Sigma^* \times \Delta^*$ (or $\Gamma^* \times \Gamma^*$) whose neutral element is $(\varepsilon, \varepsilon)$. The canonical representation of a word pair (u, v) is $\underline{u}/\underline{v}$. In particular, $\underline{(\varepsilon, \varepsilon)} = \mathbf{e}/\mathbf{e}$.

A *label set* B is a nonempty set of nonempty strings (over Ω). A *label behaviour* is a mapping $\mathcal{I} : B \to 2^M$, where M is a monoid. Thus, the behaviour $\mathcal{I}(\beta)$ is a subset of M. We shall consider label sets B with *fixed behaviours*, so we shall denote by $\mathsf{mon}\, B$ the *monoid of* B via its fixed behaviour.

We shall make the *convention* that for any label sets B_1, B_2 with fixed behaviours $\mathcal{I}_1, \mathcal{I}_2$, if $\mathsf{mon}\, B_1 = \mathsf{mon}\, B_2$ then $\mathcal{I}_1(\beta) = \mathcal{I}_2(\beta)$, for all $\beta \in B_1 \cap B_2$. With this convention we can simply use a single behaviour notation $\mathcal{I}$ for all label sets with the same behaviour monoid, that is, we shall use $\mathcal{I}$ for any B_1, B_2 with $\mathsf{mon}\, B_1 = \mathsf{mon}\, B_2$. This convention is applied in the example below: we use $\mathcal{L}$ for the behaviour of both the label sets Σ and $\mathrm{SSP}[\Gamma]$.

EX:fixed-behs *Example 2.* We shall use the following label sets and their fixed label behaviours.

1. Σ with behaviour $\mathcal{L} : \Sigma \to 2^{\Sigma^*}$ such that $\mathcal{L}(g) = \{g\}$, for $g \in \Sigma$. Thus, $\mathsf{mon}\, \Sigma = \Sigma^*$.
2. $\mathrm{SSP}[\Gamma]$ with behaviour $\mathcal{L} : \mathrm{SSP}[\Gamma] \to 2^{\Gamma^*}$, as specified in Def. 2. Thus, $\mathsf{mon}\, \mathrm{SSP}[\Gamma] = \Gamma^*$.
3. $[\Sigma, \Delta] = \{x/y \mid x \in \Sigma \cup \{\mathbf{e}\}, y \in \Delta \cup \{\mathbf{e}\}\} \setminus \{\mathbf{e}/\mathbf{e}\}$ with behaviour $\mathcal{R}()$ such that $\mathcal{R}(x/\mathbf{e}) = \{(x, \varepsilon)\}$, $\mathcal{R}(\mathbf{e}/y) = \{(\varepsilon, y)\}$, $\mathcal{R}(x/y) = \{(x, y)\}$, for any $x \in \Sigma$ and $y \in \Delta$. Thus, $\mathsf{mon}[\Sigma, \Delta] = \Sigma^* \times \Delta^*$.
4. $\mathrm{PSP}[\Gamma]$ with behaviour $\mathcal{R} : \mathrm{PSP}[\Gamma] \to 2^{\Gamma^* \times \Gamma^*}$ as specified in Def. 4. Thus, $\mathsf{mon}\, \mathrm{PSP}[\Gamma] = \Gamma^* \times \Gamma^*$.
5. If B_1, B_2 are label sets with behaviours $\mathcal{I}_1, \mathcal{I}_2$, respectively, then $[B_1, B_2]$ is the label set $\{\beta_1/\beta_2 \mid \beta_1 \in B_1, \beta_2 \in B_2\}$ with behaviour and monoid such that $\mathcal{I}(\beta_1/\beta_2) = \mathcal{I}_1(\beta_1) \times \mathcal{I}_2(\beta_2)$ and $\mathsf{mon}[B_1, B_2] = \mathsf{mon}\, B_1 \times \mathsf{mon}\, B_2$.

For any monoid of interest M and $m \in M$, $\underline{M}$ is a label set such that $\mathsf{mon}\, \underline{M} = M$ and $\mathcal{I}(\underline{m}) = \{m\}$. Thus, $\mathcal{I}(\underline{\varepsilon_M}) = \{\varepsilon_M\}$. Also, as $\mathsf{mon}\, \mathrm{PSP}[\Gamma] = \mathsf{mon}\, \underline{\Gamma^* \times \Gamma^*} = \Gamma^* \times \Gamma^*$ and the behaviour of PSP is denoted by $\mathcal{R}$, we have $\mathcal{R}(\underline{(0,1)}) = \mathcal{R}(0/1) = \{(0,1)\} = \mathcal{R}(\exists 0/\exists 1)$.

2.3 Labelled Graphs, Automata, Transducers

Let B be a label set with behaviour $\mathcal{I}$. A *type B graph* is a quintuple $\hat{g} = \big(Q, B, \delta, I, F\big)$ such that Q is a nonempty set whose elements are called *states*; $I \subseteq Q$ is the nonempty set of initial, or start states; $F \subseteq Q$ is the set of final states; δ is a set, called the set of *edges* or *transitions*, consisting of triples (p, β, q) such that $p, q \in Q$ and $\beta \in B \cup \{\underline{\varepsilon_{\mathsf{mon}\, B}}\}$. The set of *labels of* $\hat{g}$ is the set $\mathrm{Labels}(\hat{g}) = \{\beta \mid (p, \beta, q) \in \delta\}$. We shall use the term *labelled graph* to mean a type B graph as defined above, for some label set B. The labelled graph is called *finite* if Q and δ are both finite. *In the sequel, a labelled graph will be assumed to be finite.* A *path* P of $\hat{g}$ is a sequence of consecutive transitions, that is, $P = \langle q_{i-1}, \beta_i, q_i\rangle_{i=1}^{\ell}$ such that each (q_{i-1}, β_i, q_i) is in δ. The path P is called *accepting*, if $q_0 \in I$ and $q_\ell \in F$. If $\ell = 0$ then P is empty and it is an accepting path if $I \cap F \neq \emptyset$. SEC:graphs

Definition 5. *Let $\hat{g} = \big(Q, B, \delta, I, F\big)$ be a labelled graph, for some label set B with behaviour $\mathcal{I}$. We define the* behaviour *$\mathcal{I}(\hat{g})$ of $\hat{g}$ as the set of all $m \in \mathsf{mon}\, B$ such that there is an accepting path $\langle q_{i-1}, \beta_i, q_i\rangle_{i=1}^{\ell}$ of $\hat{g}$ with $m \in \mathcal{I}(\beta_1) \cdots \mathcal{I}(\beta_\ell)$. The* expansion *$\exp \hat{g}$ of $\hat{g}$ is the labelled graph $\big(Q, \underline{\mathsf{mon}\, B}, \delta_{\exp}, I, F\big)$ such that $\delta_{\exp} = \{(p, \underline{m}, q) \mid \exists\, (p, \beta, q) \in \delta : m \in \mathcal{I}(\beta)\}$.* DEF:graph:be

Lemma 1. *For each labelled graph $\hat{g}$, we have that $\mathcal{I}(\hat{g}) = \mathcal{I}(\exp \hat{g})$.* LEM:expanded

Example 3. Let Σ, Δ, Γ be alphabets. An *automaton*, or *ε-NFA*, is a labelled graph $\hat{a} = (Q, \Sigma, \delta, I, F)$. If $\mathrm{Labels}(\hat{a}) \subseteq \Sigma$ then $\hat{a}$ is called an *NFA*. The *language* $\mathcal{L}(\hat{a})$ is the behaviour of $\hat{a}$. An *automaton with set specs* is a labelled graph $\hat{b} = (Q, \mathrm{SSP}[\Gamma], \delta, I, F)$. The *language* $\mathcal{L}(\hat{b})$ is the behaviour of $\hat{b}$. A *transducer (in standard form)* is a labelled graph $\hat{t} = (Q, [\Sigma, \Delta], \delta, I, F)$. The *relation* $\mathcal{R}(\hat{t})$ *realized* by $\hat{t}$ is the behaviour of $\hat{t}$. A *transducer with set specs* is a labelled graph $\hat{s} = (Q, \mathrm{PSP}[\Gamma], \delta, I, F)$. The *relation* $\mathcal{R}(\hat{s})$ *realized* by $\hat{s}$ is the behaviour of $\hat{s}$.

2.4 Regular Expressions over Label Sets

We extend the definition of regular expressions to include set specs and pairing specs, respectively. We start off with a definition that would work with any label set (called set of atomic formulas in [16]). SEC:sym:RE

Definition 6. *Let B be a label set with behaviour $\mathcal{I}$ such that no $\beta \in B$ contains the special symbol $\oslash$. The set* $\mathrm{REG}\, B$ *of* type B regular expressions *is the set of strings consisting of the 1-symbol string $\oslash$ and the strings in the set Z that is defined inductively as follows: (i) $\underline{\varepsilon_{\mathsf{mon}\, B}}$ is in Z. (ii) Every $\beta \in B$ is in Z. (iii) If $\boldsymbol{r}, \boldsymbol{s} \in Z$ then $(\boldsymbol{r} + \boldsymbol{s}), (\boldsymbol{r}\boldsymbol{s}), (\boldsymbol{r}^*)$ are in Z. The* behaviour *$\mathcal{I}(\boldsymbol{r})$ of a type B regular expression $\boldsymbol{r}$ is defined inductively as follows.* DEF:reg:gen

- $\mathcal{I}(\oslash) = \emptyset$ *and* $\mathcal{I}(\underline{\varepsilon_{\mathsf{mon}\, B}}) = \{\varepsilon_{\mathsf{mon}\, B}\}$;
- $\mathcal{I}(\beta)$ *is the subset of* $\mathsf{mon}\, B$ *already defined by the behaviour $\mathcal{I}$ on B;*
- $\mathcal{I}(\boldsymbol{r} + \boldsymbol{s}) = \mathcal{I}(\boldsymbol{r}) \cup \mathcal{I}(\boldsymbol{s})$; $\mathcal{I}(\boldsymbol{r}\boldsymbol{s}) = \mathcal{I}(\boldsymbol{r})\mathcal{I}(\boldsymbol{s})$; $\mathcal{I}(\boldsymbol{r}^*) = \mathcal{I}(\boldsymbol{r})^*$.

Example 4. Using Σ as a label set, we have that $\mathrm{REG}\,\Sigma$ is the set of ordinary regular expressions over Σ. For the label set $[\Sigma, \Delta]$, we have that $\mathrm{REG}[\Sigma, \Delta]$ is the set of rational expressions over $\Sigma^* \times \Delta^*$ in the sense of [16]. The expressions (3) and (4) are examples of type $\mathrm{PSP}[\Gamma]$ regular expressions.

3 Partial Derivatives of type B Regular Expressions

SEC:derivatives

Here we consider any label set B with some behaviour $\mathcal{I}$ such that no $\beta \in B$ contains the special symbol $\oslash$, and we define the partial derivatives of a type B regular expression $\boldsymbol{r}$ w.r.t. an element $x \in X$, where X is also a label set such that $\mathsf{mon}\,B = \mathsf{mon}\,X$. The intention is that further below (Section 4) one can define the labelled graph corresponding to $\boldsymbol{r}$ such that the states are partial derivatives of $\boldsymbol{r}$ (type B regular expressions) and the transition labels are in X.

Derivative based methods for the manipulation of regular expressions have been widely studied [4,1,12,11,3,7,5]. In recent years, partial derivative automata were defined and characterised for several kinds of expressions. Not only they are in general more succinct than other equivalent constructions but also for several operators they are easily defined (e.g. for intersection [2] or tuples [8]). The partial derivative automaton of an ordinary (type Σ) regular expression was introduced independently by Mirkin [12] and Antimirov [1]. Champarnaud and Ziadi [6] proved that the two formulations are equivalent. Lombardy and Sakarovitch [11] generalised these constructions to weighted regular expressions, and recently Demaille [8] defined derivative automata for multitape weighted regular expressions.

Without further mention, the operator $\mathcal{I}$ as well as the operators $\mathsf{n}, \mathsf{c}, \downarrow_2, \pi_i, \pi$ defined below are extended in a union-respecting way, i.e $\phi(S) = \cup_{s \in S} \phi(s)$ for any operator ϕ.

Our assumptions about the label set B and the monoid $\mathsf{mon}\,B$:

$$\forall \beta \in B : \mathcal{I}(\beta) \neq \emptyset \tag{5}$$

$$\forall \beta \in B : \mathcal{I}(\beta) \neq \{\underline{\varepsilon_{\mathsf{mon}\,B}}\}. \tag{6}$$

Moreover we shall assume that $\mathsf{mon}\,B$ is a *graded monoid*, [15, p. 383]. For our purposes, we only need the following implication of this assumption.

EQ:monoid

$$\forall m_1, m_2 \in \mathsf{mon}\,B : \; m_1 m_2 = \varepsilon_{\mathsf{mon}\,B} \longrightarrow m_1 = m_2 = \varepsilon_{\mathsf{mon}\,B}. \tag{7}$$

The *size* of an expression $\boldsymbol{r}$ is inductively defined as follows:

$$\| \oslash \| = 0, \; \|\underline{\varepsilon_{\mathsf{mon}\,B}}\| = 0, \; \|\beta\| = 1$$

$$\|\boldsymbol{r} + \boldsymbol{s}\| = \|\boldsymbol{r}\| + \|\boldsymbol{s}\|, \; \|\boldsymbol{r}\boldsymbol{s}\| = \|\boldsymbol{r}\| + \|\boldsymbol{s}\|, \; \|\boldsymbol{r}^*\| = \|\boldsymbol{r}\|.$$

We define the *constant part* $\mathsf{c} : \mathrm{REG}\,B \to \{\underline{\varepsilon_{\mathsf{mon}\,B}}, \oslash\}$ by $\mathsf{c}(\boldsymbol{r}) = \underline{\varepsilon_{\mathsf{mon}\,B}}$ if $\varepsilon_{\mathsf{mon}\,B} \in \mathcal{I}(\boldsymbol{r})$, and $\mathsf{c}(\boldsymbol{r}) = \oslash$ otherwise. For a set R of regular expressions, $\mathsf{c}(R) = \underline{\varepsilon_{\mathsf{mon}\,B}}$ if and only if there exists $\boldsymbol{r} \in R$ such that $\mathsf{c}(\boldsymbol{r}) = \underline{\varepsilon_{\mathsf{mon}\,B}}$.

The second label set X. The linear form $\mathsf{n}(\boldsymbol{r})$ of a regular expression $\boldsymbol{r}$, defined below, is a set of pairs $(x, \boldsymbol{r}')$, in which case $\boldsymbol{r}'$ is a partial derivative of

$\boldsymbol{r}$ *with respect to* $x \in X$. The label set X is such that $\mathsf{mon}\, X = \mathsf{mon}\, B$. When the following condition (8) is satisfied, the partial derivative graph of $\boldsymbol{r}$ can be defined (Section 4) and will have as states the partial derivatives of $\boldsymbol{r}$, including $\boldsymbol{r}$, and transitions $(\boldsymbol{r}_1, x, \boldsymbol{r}_2)$ when $(x, \boldsymbol{r}_2) \in \mathsf{n}(\boldsymbol{r}_1)$.

$$\forall \beta \in B : \ \mathcal{I}(\beta) \ = \ \mathcal{I}(\mathsf{c}(\beta)) \cup \mathcal{I}(\mathsf{n}(\beta)), \tag{8}$$ EQ:gra

Some Notation. Let $\beta \in B$ and let $\boldsymbol{r}, \boldsymbol{r}', \boldsymbol{s}, \boldsymbol{s}', \boldsymbol{z} \in \mathrm{REG}\, B \setminus \{\oslash\}$ such that $\mathcal{I}(\boldsymbol{s}) \neq \{\varepsilon_{\mathsf{mon}\, B}\}$, $\mathcal{I}(\boldsymbol{s}') \neq \{\varepsilon_{\mathsf{mon}\, B}\}$, $\mathcal{I}(\boldsymbol{z}) = \{\varepsilon_{\mathsf{mon}\, B}\}$. The binary operation $\diamond$ between any two expressions in $\mathrm{REG}\, B \setminus \{\oslash\}$ is defined as follows: $\boldsymbol{r} \diamond \boldsymbol{z} = \boldsymbol{z} \diamond \boldsymbol{r} = \boldsymbol{r}$ and $\boldsymbol{s} \diamond \boldsymbol{s}' = \boldsymbol{s}\boldsymbol{s}'$. For any $\tilde{S} \subseteq X \times (\mathrm{REG}\, B \setminus \{\oslash\})$, we define $\mathcal{I}(\tilde{S}) = \bigcup_{(x,\boldsymbol{s}) \in \tilde{S}} \mathcal{I}(x)\mathcal{I}(\boldsymbol{s})$ and

$$\oslash \tilde{S} = \emptyset, \ \ \tilde{S}\boldsymbol{s}' = \{\, (x, \boldsymbol{s} \diamond \boldsymbol{s}') \mid (x, \boldsymbol{s}) \in \tilde{S} \,\}, \ \ \boldsymbol{s}'\tilde{S} = \{\, (x, \boldsymbol{s}' \diamond \boldsymbol{s}) \mid (x, \boldsymbol{s}) \in \tilde{S} \,\}$$

For any $R \subseteq \mathrm{REG}\, B \setminus \{\oslash\}$, we also define $\oslash R = \emptyset$, $R\boldsymbol{s}' = \{\, \boldsymbol{s} \diamond \boldsymbol{s}' \mid \boldsymbol{s} \in R \,\}$, $\boldsymbol{s}'R = \{\, \boldsymbol{s}' \diamond \boldsymbol{s} \mid \boldsymbol{s} \in R \,\}$.

Definition 7. *A* linear form *(of type (X, B)) of a regular expression is defined inductively as follows:* DEF:lform

$$\mathsf{n}(\oslash) = \emptyset, \qquad \mathsf{n}(\underline{\varepsilon_{\mathsf{mon}\, B}}) = \emptyset,$$
$$\mathsf{n}(\beta) = \textit{a chosen finite nonempty subset of } \ X \times \{\underline{\varepsilon_{\mathsf{mon}\, B}}\},$$
$$\mathsf{n}(\boldsymbol{r} + \boldsymbol{r}') = \mathsf{n}(\boldsymbol{r}) \cup \mathsf{n}(\boldsymbol{r}'), \quad \mathsf{n}(\boldsymbol{r}\boldsymbol{r}') = \mathsf{n}(\boldsymbol{r})\boldsymbol{r}' \cup \mathsf{c}(\boldsymbol{r})\, \mathsf{n}(\boldsymbol{r}'), \qquad \mathsf{n}(\boldsymbol{r}^*) = \mathsf{n}(\boldsymbol{r})\boldsymbol{r}^*.$$

Example 5. The default linear form: $X = B$ and $\forall \beta \in B : \mathsf{n}(\beta) = \{(\beta, \underline{\varepsilon_{\mathsf{mon}\, B}})\}$. Trivially, this n satisfies condition (8). *The* expanding linear form: when EX:lf

$$\mathcal{I}(\beta) \subseteq \Phi, \quad X = \underline{\Phi}, \quad \forall \beta \in B : \mathsf{n}(\beta) = \{(\underline{m}, \underline{\varepsilon_{\mathsf{mon}\, B}}) \mid m \in \mathcal{I}(\beta)\},$$

where Φ is a finite set of generators of $\mathsf{mon}\, B$. Again, the expanding linear form satisfies condition (8). For example, if $B = \mathrm{SSP}[\Gamma]$ then $\Phi = \Gamma$ and, for any set spec F, $\mathsf{n}(F) = \{(f, \mathbf{e}) \mid f \in \mathcal{L}(F)\}$. □

For any $x \in X$ and any $\boldsymbol{r} \in \mathrm{REG}\, B$, the *set of partial derivatives of* $\boldsymbol{r}$ *w.r.t.* x is $\partial_x(\boldsymbol{r}) \ = \ \{\boldsymbol{r}' \mid (x, \boldsymbol{r}') \in \mathsf{n}(\boldsymbol{r})\}$. For all $\boldsymbol{r}, \boldsymbol{r}' \in \mathrm{REG}\, B \setminus \{\oslash\}$ and $x \in X$, one can confirm that

$$\partial_x(\oslash) = \partial_x(\underline{\varepsilon_{\mathsf{mon}\, B}}) = \emptyset, \qquad \partial_x(\boldsymbol{r} + \boldsymbol{r}') = \partial_x(\boldsymbol{r}) \cup \partial_x(\boldsymbol{r}'),$$
$$\partial_x(\boldsymbol{r}\boldsymbol{r}') = \partial_x(\boldsymbol{r})\boldsymbol{r}' \cup \mathsf{c}(\boldsymbol{r})\partial_x(\boldsymbol{r}'), \quad \partial_x(\boldsymbol{r}^*) = \partial_x(\boldsymbol{r})\boldsymbol{r}^*.$$

As in the case of ordinary derivatives, [1], the following result explains how the behaviour of the linear form of $\boldsymbol{r}$ relates to the behaviour of $\boldsymbol{r}$.

Lemma 2. *Let the linear form* n *satisfy condition* (8)*. For all* $\boldsymbol{r} \in \mathrm{REG}\, B$*, we have* $\mathcal{I}(\boldsymbol{r}) = \mathcal{I}(\mathsf{c}(\boldsymbol{r})) \ \cup \ \mathcal{I}(\mathsf{n}(\boldsymbol{r}))$. lem:clfeqreg

Next we explain how to iterate $\mathsf{n}(\boldsymbol{r})$ to obtain the set of derivatives of the regular expression $\boldsymbol{r}$. We start with defining the operator

$$\pi_0(\boldsymbol{r}) = \downarrow_2(\mathsf{n}(\boldsymbol{r})),$$

where $\downarrow_2(s,t) = t$ is the standard second projection on pairs of objects.

We can iteratively apply the operator π_0 on any expression $\boldsymbol{x} \in \pi_0(\boldsymbol{r})$. The set of all the resulting expressions is denoted by $\pi(\boldsymbol{r})$, and iteratively defined by

$$\pi_i(\boldsymbol{r}) = \pi_0(\pi_{i-1}(\boldsymbol{r})) \quad (i \in \mathbb{N}), \qquad \pi(\boldsymbol{r}) = \bigcup_{i \in \mathbb{N}_0} \pi_i(\boldsymbol{r}).$$

Let $\mathsf{PD}(\boldsymbol{r}) = \pi(\boldsymbol{r}) \cup \{\boldsymbol{r}\}$ be the *set of partial derivatives* of $\boldsymbol{r}$.

Example 6. Consider the case of the default linear form and the type $\mathrm{PSP}[\Gamma]$ regular expression $\boldsymbol{r} = (\forall/=)^*(\forall/\mathbf{e})^*$. We have

$$\mathsf{n}((\forall/=)^*) = \mathsf{n}(\forall/=)(\forall/=)^* = \{(\forall/=, \mathbf{e}/\mathbf{e})\}(\forall/=)^* = \{(\forall/=, (\forall/=)^*)\}$$
$$\mathsf{n}((\forall/\mathbf{e})^*) = \mathsf{n}(\forall/\mathbf{e})(\forall/\mathbf{e})^* = \{(\forall/\mathbf{e}, \mathbf{e}/\mathbf{e})\}(\forall/\mathbf{e})^* = \{(\forall/\mathbf{e}, (\forall/\mathbf{e})^*)\}.$$

As $(\varepsilon, \varepsilon) \in \mathcal{R}((\forall/=)^*)$, we have $\mathsf{n}(\boldsymbol{r}) = \mathsf{n}((\forall/=)^*)(\forall/\mathbf{e})^* \cup \mathsf{n}((\forall/\mathbf{e})^*) = \{(\forall/=, \boldsymbol{r}), (\forall/\mathbf{e}, (\forall/\mathbf{e})^*)\}$. Then, $\pi_0(\boldsymbol{r}) = \{\boldsymbol{r}, (\forall/\mathbf{e})^*\}, \pi_1(\boldsymbol{r}) = \pi_0(\boldsymbol{r}) \cup \pi_0((\forall/\mathbf{e})^*) = \pi_0(\boldsymbol{r})$, $\pi(\boldsymbol{r}) = \{\boldsymbol{r}, (\forall/\mathbf{e})^*\}$.

Example 7. Consider the type $\mathrm{SSP}[\Gamma]$ regular expression $\boldsymbol{r} = (\forall^*)(\exists \mathsf{b})$ and the case of the expanding linear form n such that $\mathsf{n}(F) = \{(f, \mathbf{e}) \mid f \in \mathcal{L}(F)\}$, for any set spec F, and $X = \Gamma = \{\mathsf{a}, \mathsf{b}, \ldots, \mathsf{z}\}$. We have

$$\mathsf{n}(\forall^*) = \mathsf{n}(\forall)(\forall^*) = (\Gamma \times \{\mathbf{e}\})(\forall^*) = \Gamma \times \{\forall^*\} \quad \text{and} \quad \mathsf{n}(\exists \mathsf{b}) = \{(\exists \mathsf{b}, \mathbf{e})\}.$$

Also, as $\varepsilon \in \mathcal{L}(\forall^*)$, we have $\mathsf{n}(\boldsymbol{r}) = \mathsf{n}(\forall^*)(\exists \mathsf{b}) \cup \mathsf{n}(\exists \mathsf{b}) = \Gamma \times \{\boldsymbol{r}\} \cup \{(\mathsf{b}, \mathbf{e})\}$. Then, $\pi_0(\boldsymbol{r}) = \{\boldsymbol{r}, \mathbf{e}\}$, $\pi_1(\boldsymbol{r}) = \pi_0(\boldsymbol{r})$, $\pi(\boldsymbol{r}) = \pi_0(\boldsymbol{r})$, $\partial_{\mathsf{a}}(\boldsymbol{r}) = \{\boldsymbol{r}, \mathbf{e}\}$ and $\partial_{\mathsf{b}}(\boldsymbol{r}) = \{\boldsymbol{r}\}$.

Theorem 1. *Suppose that partial derivatives are defined based on some type (X, B) linear form. For all $\boldsymbol{r} \in \mathrm{REG}\, B$, $|\pi(\boldsymbol{r})| \leq \|\boldsymbol{r}\|$ and $|\mathsf{PD}(\boldsymbol{r})| \leq \|\boldsymbol{r}\| + 1$.*

4 The partial derivative graph of a regular expression

sec:PDgraph

Here we consider a label set B to be used for type B regular expressions $\boldsymbol{r}$ and a label set X that will be used to define the type X labelled graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$, such that $\mathsf{mon}\, B = \mathsf{mon}\, X$ and condition (8) is satisfied for all $\beta \in B$—recall that $\mathsf{n}(\beta) \subseteq X \times \{\underline{\varepsilon_{\mathsf{mon}\, B}}\}$. The objective is to prove that the behaviour of $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ is exactly $\mathcal{I}(\boldsymbol{r})$—see Theorem 2. This is analogous to the case of ordinary regular expressions [12,1]. Thus, to decide whether a given $m \in \mathsf{mon}\, B$ is in $\mathcal{I}(\boldsymbol{r})$, one computes $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ and then tests whether $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ accepts m. This test depends on the particular monoid $\mathsf{mon}\, B$ [10].

Definition 8. *The type X* partial derivative graph *of a type B regular expression $\boldsymbol{r}$ is the labelled graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r}) = \big(\mathsf{PD}(\boldsymbol{r}), X, \delta_{\mathrm{PD}}, \{\boldsymbol{r}\}, \lambda(\boldsymbol{r})\big)$, where $\lambda(\boldsymbol{r}) = \{\, \boldsymbol{r}_1 \in \mathsf{PD}(\boldsymbol{r}) \mid c(\boldsymbol{r}_1) = \underline{\varepsilon_{\mathsf{mon}\, B}} \,\}$ and $\delta_{\mathrm{PD}} = \{\, (\boldsymbol{r}_1, x, \boldsymbol{r}_2) \mid \boldsymbol{r}_1 \in \mathsf{PD}(\boldsymbol{r}) \wedge (x, \boldsymbol{r}_2) \in \mathsf{n}(\boldsymbol{r}_1) \,\}$.*

Theorem 2. *Suppose that partial derivatives are defined based on some type (X, B) linear form. For any $\boldsymbol{r} \in \mathrm{REG}\, B$, we have that $\mathcal{I}(\hat{a}_{\mathrm{PD}}(\boldsymbol{r})) = \mathcal{I}(\boldsymbol{r})$.*

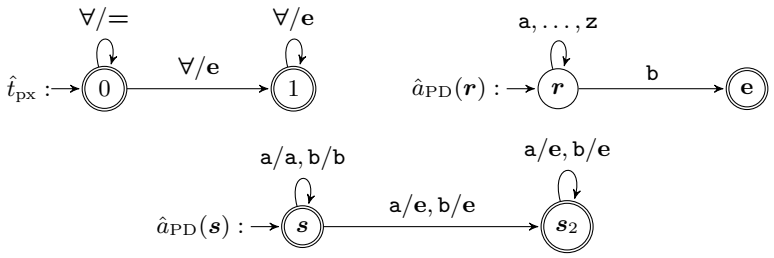

Fig. 1: The transducer $\hat{t}_{\mathrm{px}}$ is alphabet invariant and realizes all (u, v) such that v is a prefix of u. The automaton $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ accepts all words in $\{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}^*$ ending with $\mathtt{b}$. The transducer $\hat{a}_{\mathrm{PD}}(\boldsymbol{s})$ realizes all (u, v) such that v is a prefix of u and $\Gamma = \{\mathtt{a}, \mathtt{b}\}$.

FIG:apd2

Example 8. Consider again the regular expression $\boldsymbol{r} = (\forall/\mathtt{=})^*(\forall/\mathtt{e})^*$ over $\mathrm{PSP}[\Gamma]$ representing all word pairs (u, v) such that v is a prefix of u. We compute the partial derivative graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ using the default linear form for $X = B = \mathrm{PSP}[\Gamma]$. In Ex. 6, we computed $\mathsf{n}(\boldsymbol{r}) = \{(\forall/\mathtt{=}, \boldsymbol{r}), (\forall/\mathtt{e}, (\forall/\mathtt{e})^*)\}$ and $\pi(\boldsymbol{r}) = \{\boldsymbol{r}, (\forall/\mathtt{e})^*\}$. Using the linear forms $\mathsf{n}(\boldsymbol{r})$ and $\mathsf{n}((\forall/\mathtt{e})^*)$, we see that the partial derivative graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ is exactly the transducer $\hat{t}_{\mathrm{px}}$ in Fig. 1. EX:apd1

Example 9. Consider again the type $\mathrm{SSP}[\Gamma]$ regular expression $\boldsymbol{r} = (\forall^*)(\exists\mathtt{b})$ of Ex. 7, representing all words ending with $\mathtt{b}$. The partial derivative graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ is the automaton in Fig. 1. EX:apd2

Corollary 1. *Consider the default linear form for $X = B = [\Sigma, \Delta]$. For any type $[\Sigma, \Delta]$ regular expression $\boldsymbol{r}$, the type $[\Sigma, \Delta]$ partial derivative graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ is a transducer (in standard) form such that $\mathcal{R}(\boldsymbol{r}) = \mathcal{R}(\hat{a}_{\mathrm{PD}}(\boldsymbol{r}))$.* COR:2D

m Let $\Sigma = \Delta = \{\mathtt{a}, \mathtt{b}\}$ and let n be the default linear form for $X = B = [\Sigma, \Delta]$. The type $[\Sigma, \Delta]$ expression $\boldsymbol{s} = (\mathtt{a}/\mathtt{a} + \mathtt{b}/\mathtt{b})^*(\mathtt{a}/\mathtt{e} + \mathtt{b}/\mathtt{e})^*$ represents all (u, v) such that v is a prefix of u. Let $\boldsymbol{s}_1 = (\mathtt{a}/\mathtt{a} + \mathtt{b}/\mathtt{b})^*$ and $\boldsymbol{s}_2 = (\mathtt{a}/\mathtt{e} + \mathtt{b}/\mathtt{e})^*$. Then, $\mathsf{n}(\boldsymbol{s}_1) = \{(\mathtt{a}/\mathtt{a}, \boldsymbol{r}_1), (\mathtt{b}/\mathtt{b}, \boldsymbol{r}_1)\}$, $\mathsf{n}(\boldsymbol{s}_2) = \{(\mathtt{a}/\mathtt{e}, \boldsymbol{s}_2), (\mathtt{b}/\mathtt{e}, \boldsymbol{s}_2)\}$, $\mathsf{n}(\boldsymbol{r}) = \{(\mathtt{a}/\mathtt{a}, \boldsymbol{r}), (\mathtt{b}/\mathtt{b}, \boldsymbol{r}), (\mathtt{a}/\mathtt{e}, \boldsymbol{s}_2), (\mathtt{b}/\mathtt{e}, \boldsymbol{s}_2)\}$. The graph $\hat{a}_{\mathrm{PD}}(\boldsymbol{s})$ is shown in Fig. 1.

5 2D Regular Expressions

By 2D regular expressions we mean type B regular expressions with $\mathsf{mon}\, B = \Sigma^* \times \Delta^*$ (or $\mathsf{mon}\, B = \Gamma^* \times \Gamma^*$). We want a direct algorithm to decide if (u, v) belongs to $\mathcal{R}(\boldsymbol{r})$, without constructing the transducer $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ and then testing whether $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$ accepts (u, v). To this end, we shall define partial derivatives $\partial_\psi(\beta)$, where $\psi \in X$, a little differently. Due to space limitation we shall deal only with the case of $X = \{F/\mathtt{e}, \mathtt{e}/F \mid F \in \mathrm{SSP}[\Gamma]\}$ and[4] $B = \mathrm{PSP}_{\neq\emptyset}[\Gamma]$. See [14] for the case of $X = \{x/\mathtt{e}, \mathtt{e}/y \mid x \in \Sigma, y \in \Delta\}$ and $B = [\Sigma, \Delta]$. sec:pdpairspec

[4] Because of condition (5), here we consider labels in $\mathrm{PSP}_{\neq\emptyset}[\Gamma]$, that is, only those labels $\mathsf{p} \in \mathrm{PSP}[\Gamma]$ for which $\mathcal{R}(\mathsf{p}) \neq \emptyset$.

Consider the monoid $\Sigma^* \times \Delta^*$ with set of generators $\{(x,\varepsilon),(\varepsilon,y) \mid x \in \Sigma \wedge y \in \Delta\}$ and set of equations $\{\,(x,\varepsilon)(\varepsilon,y) \doteq (x,y), (\varepsilon,y)(x,\varepsilon) \doteq (x,y) \mid x \in \Sigma \wedge y \in \Delta\,\}$. The partial derivatives of this section are related to the quotient of relations $R \subseteq \Sigma^* \times \Delta^*$, by word pairs. But now one needs to take in account the above equations. For instance, for $x \in \Sigma$ and $y \in \Delta$, we have

$$(x,\varepsilon)^{-1}R = \{\,(\varepsilon,y)w \mid (x,y)w \in R\}, \qquad (\varepsilon,y)^{-1}R = \{\,(x,\varepsilon)w \mid (x,y)w \in R\}.$$

Quotients can be extended appropriately: For θ a pair as above and ω a concatenation of such pairs, $(\varepsilon,\varepsilon)^{-1}R_1 = R_1$, $(\omega\theta)^{-1}R_1 = \theta^{-1}(\omega^{-1}R_1)$. For $R_1 \subseteq (\Sigma^* \times \{\varepsilon\}) \cup (\{\varepsilon\} \times \Delta^*)$, we have $R_1{}^{-1}R_2 = \bigcup_{\theta \in R_1} \theta^{-1}R_2$.

The partial derivatives of any $\mathsf{p} \in \mathrm{PSP}_{\neq\emptyset}[\Gamma]$ are defined w.r.t. elements in $X = \{F/\mathbf{e},\, \mathbf{e}/F \mid F \in \mathrm{SSP}[\Gamma]\}$ are as follows.

$$\begin{aligned}
\partial_{\mathbf{e}/F}(G/\mathbf{e}) &= \partial_{F/\mathbf{e}}(\mathbf{e}/G) = \emptyset,\\
\partial_{\mathbf{e}/F}(\mathbf{e}/G) &= \partial_{F/\mathbf{e}}(G/\mathbf{e}) = \{\mathbf{e}/\mathbf{e}\} \quad \text{if } \mathcal{L}(F) \cap \mathcal{L}(G) \neq \emptyset,\\
\partial_{\mathbf{e}/F}(G/C) &= \{G/\mathbf{e}\} \quad \text{if } \mathcal{L}(F) \cap \mathcal{L}(C) \neq \emptyset,\\
\partial_{\mathbf{e}/F}(G/{=}) &= \{(F \cap G)/\mathbf{e}\} \quad \text{if } \mathcal{L}(F) \cap \mathcal{L}(G) \neq \emptyset,\\
\partial_{\mathbf{e}/F}(G/C{\neq}) &= \begin{cases} \{(G \cap \not\ni b)/\mathbf{e}\} & \text{if } \mathcal{L}(F \cap C) = \{b\} \wedge \mathcal{L}(G) \setminus \{b\} \neq \emptyset,\\ \{G/\mathbf{e}\} & \text{if } |\mathcal{L}(F \cap C)| \geq 2, \end{cases}\\
\partial_{F/\mathbf{e}}(G/{=}) &= \{\mathbf{e}/(F \cap G)\} \quad \text{if } \mathcal{L}(F) \cap \mathcal{L}(G) \neq \emptyset,\\
\partial_{F/\mathbf{e}}(G/C{\neq}) &= \begin{cases} \{\mathbf{e}/(C \cap \not\ni b)\} & \text{if } \mathcal{L}(F \cap G) = \{b\} \wedge \mathcal{L}(C) \setminus \{b\} \neq \emptyset,\\ \{\mathbf{e}/C\} & \text{if } |\mathcal{L}(F \cap G)| \geq 2. \end{cases}
\end{aligned}$$

For each case above, if the conditions do not hold then the set of partial derivatives is $\emptyset$. Above we have used the operation $\cap$ between any two set specs, defined in [10] in a natural way, e.g., $\exists 035 \cap \exists 1358 = \exists 35$, $\not\ni 035 \cap \exists 1358 = \exists 18$.

Partial derivatives $\partial_{F/\mathbf{e}}(\boldsymbol{r})$ and $\partial_{\mathbf{e}/F}(\boldsymbol{r})$ of any $\boldsymbol{r} \in \mathrm{REG}\,\mathrm{PSP}_{\neq\emptyset}[\Gamma]$ are defined as in Section 3, except for the concatenation $\boldsymbol{r}\boldsymbol{s}$. Let φ be either of $F/\mathbf{e}, \mathbf{e}/F$:

$$\begin{aligned}
&\partial_\varphi(\oslash) = \partial_\varphi(\mathbf{e}/\mathbf{e}) = \emptyset, \quad \partial_\varphi(\boldsymbol{r}+\boldsymbol{s}) = \partial_\varphi(\boldsymbol{r}) \cup \partial_\varphi(\boldsymbol{s}), \quad \partial_\varphi(\boldsymbol{r}^*) = \partial_\varphi(\boldsymbol{r})\boldsymbol{r}^*,\\
&\partial_{F/\mathbf{e}}(\boldsymbol{r}\boldsymbol{s}) = \partial_{F/\mathbf{e}}(\boldsymbol{r})\boldsymbol{s} \cup \mathsf{c}_{in}(\boldsymbol{r})\partial_{F/\mathbf{e}}(\boldsymbol{s}), \quad \partial_{\mathbf{e}/F}(\boldsymbol{r}\boldsymbol{s}) = \partial_{\mathbf{e}/F}(\boldsymbol{r})\boldsymbol{s} \cup \mathsf{c}_{out}(\boldsymbol{r})\partial_{\mathbf{e}/F}(\boldsymbol{s});
\end{aligned}$$

where c_{in} is the *constant-input part* defined such that $\mathsf{c}_{in}(\mathbf{e}/\mathbf{e}) = \mathbf{e}/\mathbf{e}$, $\mathsf{c}_{in}(\mathbf{e}/F) = \mathbf{e}/F$, and $\mathsf{c}_{in}(\mathsf{p}) = \oslash$ for all other pairing specs p. Moreover for $F \in \mathrm{SSP}[\Gamma]$ and $\boldsymbol{r}, \boldsymbol{s} \in \mathrm{REG}\,\mathrm{PSP}_{\neq\emptyset}[\Gamma]$,

$$\begin{aligned}
\mathsf{c}_{in}(\boldsymbol{r}\boldsymbol{s}) &= \mathsf{c}_{in}(\boldsymbol{r})\,\mathsf{c}_{in}(\boldsymbol{s}), \quad \mathsf{c}_{in}(\oslash\boldsymbol{s}) = \mathsf{c}_{in}(\boldsymbol{r}\oslash) = \mathsf{c}_{in}(\oslash) = \oslash,\\
\mathsf{c}_{in}(\boldsymbol{r}+\boldsymbol{s}) &= \mathsf{c}_{in}(\boldsymbol{r}) + \mathsf{c}_{in}(\boldsymbol{s}), \quad \mathsf{c}_{in}(\oslash+\boldsymbol{s}) = \mathsf{c}_{in}(\boldsymbol{s}), \quad \mathsf{c}_{in}(\boldsymbol{r}+\oslash) = \mathsf{c}_{in}(\boldsymbol{r}),\\
\mathsf{c}_{in}(\boldsymbol{r}^*) &= (\mathsf{c}_{in}(\boldsymbol{r}))^*,\ \mathsf{c}_{in}(\oslash^*) = \mathbf{e}/\mathbf{e}.
\end{aligned}$$

The *constant-output part* is analogous except that $\mathsf{c}_{out}(F/\mathbf{e}) = F/\mathbf{e}$.

Lemma 3. *For all $\boldsymbol{r} \in \mathrm{REG}\,\mathrm{PSP}_{\neq\emptyset}[\Gamma]$, we have $\mathcal{R}(\mathsf{c}_{in}(\boldsymbol{r})) = \mathcal{R}(\boldsymbol{r}) \downarrow \{\varepsilon\}$ and $\mathcal{R}(\mathsf{c}_{out}(\boldsymbol{r})) = \mathcal{R}(\boldsymbol{r}) \uparrow \{\varepsilon\}$. Moreover $\mathcal{R}(\mathsf{c}_{in}(\mathsf{c}_{out}(\boldsymbol{r}))) = \mathcal{R}(\mathsf{c}(\boldsymbol{r})) = \mathcal{R}(\mathsf{c}_{out}(\mathsf{c}_{in}(\boldsymbol{r})))$.*

Theorem 3. *For all (u, v) with $u = x_1 \cdots x_n$ and $v = y_1 \cdots y_m$, we have that $(u, v) \in \mathcal{R}(\boldsymbol{r})$ if and only if $\mathsf{c}(\partial_{\exists x_1/\mathbf{e},\ldots,\exists x_n/\mathbf{e},\, \mathbf{e}/\exists y_1,\ldots,\mathbf{e}/\exists y_m}(\boldsymbol{r})) = \mathbf{e}/\mathbf{e}$.* TH:wordpairsp

Remark 1. It can be shown that $\partial_{\exists x/\mathbf{e}}$ and $\partial_{\mathbf{e}/\exists y}$ commute on any $\boldsymbol{r}$; thus, we have that $\mathsf{c}(\partial_{\exists x_1/\mathbf{e},\ldots,\exists x_n/\mathbf{e},\, \mathbf{e}/\exists y_1,\ldots,\mathbf{e}/\exists y_m}(\boldsymbol{r})) = \mathsf{c}(\partial_{\exists x_1/\mathbf{e},\, \mathbf{e}/\exists y_1,\exists x_2/\mathbf{e},\mathbf{e}/\exists y_2,\ldots}(\boldsymbol{r}))$.

Added this. Very little space left

Example 10. Consider the word pair $(\mathtt{aaba}, \mathtt{aaaa})$ and the type $\mathrm{PSP}_{\neq\emptyset}[\Gamma]$ regular expression $\boldsymbol{r} = (\forall/=)^*(\forall/\forall\neq)(\forall/=)^*$. We shall confirm that $(\mathtt{aaba}, \mathtt{aaaa}) \in \mathcal{R}(\boldsymbol{r})$ using Theorem 3, that is, by showing that ex:psmenbershi

there is $\boldsymbol{r}_1 \in \partial_{\exists\mathtt{a}/\mathbf{e},\exists\mathtt{a}/\mathbf{e},\exists\mathtt{b}/\mathbf{e},\exists\mathtt{a}/\mathbf{e},\exists\mathtt{e}/\mathtt{a},\exists\mathtt{e}/\mathtt{a},\exists\mathtt{e}/\mathtt{a},\exists\mathtt{e}/\mathtt{a}}(\boldsymbol{r})$ such that $(\varepsilon, \varepsilon) \in \mathcal{R}(\boldsymbol{r}_1)$.

Note that the only information about the alphabet is that it contains the letters $\mathtt{a}$ and $\mathtt{b}$. We shall compute only the necessary derivatives that lead to such an $\boldsymbol{r}_1$.

First we have: $\partial_{\exists\mathtt{a}/\mathbf{e}}((\forall/=)^*) = \{(\mathbf{e}/\exists\mathtt{a})(\forall/=)^*\}$, $\partial_{\exists\mathtt{a}/\mathbf{e}}(\forall/\forall\neq) = \{\mathbf{e}/\nexists\mathtt{a}\}$, $\mathsf{c}_{in}((\forall/=)^*) = \mathbf{e}/\mathbf{e}$, $\mathsf{c}_{in}(\forall/\forall\neq) = \varnothing$.

Let $\boldsymbol{r}_1 = (\forall/=)^*$ and $\boldsymbol{r}_2 = (\forall/\forall\neq)$. Then

$$\begin{aligned}
\partial_{\exists\mathtt{a}/\mathbf{e}}(\boldsymbol{r}) &= \partial_{\exists\mathtt{a}/\mathbf{e}}(\boldsymbol{r}_1)\boldsymbol{r}_2\boldsymbol{r}_1 \cup \mathsf{c}_{in}(\boldsymbol{r}_1)\partial_{\exists\mathtt{a}/\mathbf{e}}(\boldsymbol{r}_2\boldsymbol{r}_1) \\
&= \{(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}\} \cup \partial_{\exists\mathtt{a}/\mathbf{e}}(\boldsymbol{r}_2)\boldsymbol{r}_1 \cup \mathsf{c}_{in}(\boldsymbol{r}_2)\partial_{\exists\mathtt{a}/\mathbf{e}}(\boldsymbol{r}_1) \\
&= \{(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r},\ (\mathbf{e}/\nexists\mathtt{a})\boldsymbol{r}_1\}. \\
\partial_{\exists\mathtt{a}/\mathbf{e}}((\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}) &= (\mathbf{e}/\exists\mathtt{a})\partial_{\exists\mathtt{a}/\mathbf{e}}(\boldsymbol{r}) = \{(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}\} \cup \cdots \\
\partial_{\exists\mathtt{b}/\mathbf{e}}\big((\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}\big) &= \{(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\nexists\mathtt{b})\boldsymbol{r}_1\} \cup \cdots \\
\partial_{\exists\mathtt{a}/\mathbf{e}}\big((\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\nexists\mathtt{b})\boldsymbol{r}_1\big) &= \{(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\nexists\mathtt{b})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\} \cup \cdots \\
\partial_{\mathbf{e}/\exists\mathtt{a}}\big((\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\nexists\mathtt{b})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\big) &= \{(\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\nexists\mathtt{b})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\} \cup \cdots \\
\partial_{\mathbf{e}/\exists\mathtt{a}}\big((\mathbf{e}/\exists\mathtt{a})(\mathbf{e}/\nexists\mathtt{b})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\big) &= \{(\mathbf{e}/\nexists\mathtt{b})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\} \cup \cdots \\
\partial_{\mathbf{e}/\exists\mathtt{a}}\big((\mathbf{e}/\nexists\mathtt{b})(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\big) &= \{(\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\} \cup \cdots \\
\partial_{\mathbf{e}/\exists\mathtt{a}}\big((\mathbf{e}/\exists\mathtt{a})\boldsymbol{r}_1\big) &= \{\boldsymbol{r}_1\} \cup \cdots \qquad \text{Thus, } (\varepsilon, \varepsilon) \in \mathcal{R}(\boldsymbol{r}_1).
\end{aligned}$$

6 Concluding Remarks

Label sets can have any desired format as long as one provides their monoidal behaviour. Using the elements of a label set B we can build type B regular expressions, which can have a significantly reduced size when the alphabet of reference is large. At this broad level, we were able to obtain a few basic results on partial derivatives of these expressions. Already FAdo [9] includes implementations of partial derivative (PD) algorithms for ordinary (1D) regular expressions and of type $[\Sigma, \Delta]$ regular expressions. We are currently implementing PD algorithms for type $\mathrm{SSP}[\Gamma]$ and $\mathrm{PSP}[\Gamma]$ expressions. SEC:conclude

A research direction is to investigate the efficiency of the two approaches to the regular expression $\boldsymbol{r}$ membership (word) problem: directly or via building $\hat{a}_{\mathrm{PD}}(\boldsymbol{r})$. Solving the regular expression membership problem directly for 2D expressions required a modified definition of partial derivatives (PDs). So another research direction is to find a good way to generalize the definition of linear form n such that $\mathsf{n}(\beta)$ is a finite nonempty subset of $X \times B \cup \{\varepsilon_{\mathsf{mon}\,B}\}$ and $\mathsf{n}(\boldsymbol{r}\boldsymbol{s})$ is defined appropriately to include both the original and the modified PDs.

Remove this sentence? (see Rev.2)

If yes, fix here.

Acknowledgement

We are grateful to the reviewers of CIAA 2019 for their constructive suggestions for improvement. We have applied most of these suggestions, and we plan to apply the remaining ones in the journal version where more pages are allowed. The idea of using special labels on automata to denote sets is also explored in [13] with different objectives.

I added this because there is no space to address all Ref. comments.

Possibly add this sentence (Rev 2)

References

1. Antimirov, V.M.: Partial derivatives of regular expressions and finite automaton constructions. Theoret. Comput. Sci. 155(2), 291–319 (1996)
2. Bastos, R., Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the average complexity of partial derivative automata for semi-extended expressions. J. Automata, Languages and Combinatorics 22(1–3), 5–28 (2017)
3. Broda, S., Machiavelo, A., Moreira, N., Reis, R.: On the average state complexity of partial derivative automata: an analytic combinatorics approach. International Journal of Foundations of Computer Science 22(7), 1593–1606 (2011), mR2865339
4. Brzozowski, J.: Derivatives of regular expressions. J. Association for Computer Machinery (11), 481–494 (1964)
5. Caron, P., Champarnaud, J.M., Mignot, L.: Partial derivatives of an extended regular expression. In: Dediu, A.H., Inenaga, S., Martín-Vide, C. (eds.) Proc. 5th LATA 2011. vol. 6638, pp. 179–191. Springer (2011)
6. Champarnaud, J.M., Ziadi, D.: From Mirkin's prebases to Antimirov's word partial derivatives. Fundam. Inform. 45(3), 195–205 (2001)
7. Champarnaud, J.M., Ziadi, D.: Canonical derivatives, partial derivatives and finite automaton constructions. Theoret. Comput. Sci. 289, 137–163 (2002)
8. Demaille, A.: Derived-term automata of multitape expressions with composition. Scientific Annals of Computer Science 27(2), 137–176 (2017)
9. FAdo: Tools for formal languages manipulation, uRL address: `http://fado.dcc.fc.up.pt/` Accessed in March, 2019
10. Konstantinidis, S., Moreira, N., Reis, R., Young, J.: Regular expressions and transducers over alphabet-invariant and user-defined labels Submitted for publication. See also arXiv:1805.01829
11. Lombardy, S., Sakarovitch, J.: Derivatives of rational expressions with multiplicity. Theor. Comput. Sci. 332(1-3), 141–177 (2005), https://doi.org/10.1016/j.tcs.2004.10.016
12. Mirkin, B.G.: An algorithm for constructing a base in a language of regular expressions. Engineering Cybernetics 5, 51—57 (1966)
13. Newton, J.: Representing and Computing with Types in Dynamically Typed Languages. Ph.D. thesis, Sorbonne Université, Paris, France (Nov 2018)
14. Pires, J.: Transducers and 2D Regular Expressions. Master's thesis, Departamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto, Porto, Portugal (2018)
15. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, Berlin (2009)
16. Sakarovitch, J.: Automata and rational expressions. arXiv.org arXiv:1502.03573 (2015)