Construction of tree automata from regular expressions

Dietrich Kuske and Ingmar Meinecke
Institut für Informatik, Universität Leipzig

Abstract

Since recognizable tree languages are closed under the rational operations, every regular tree expression denotes a recognizable tree language. We provide an alternative proof to this fact that results in smaller tree automata. To this aim, we transfer Antimirov’s partial derivatives from regular word expressions to regular tree expressions. For an analysis of the size of the resulting automaton as well as for algorithmic improvements, we also transfer the methods of Champarnaud and Ziadi from words to trees.

1 Introduction

One of the most prominent topics in formal language theory is the comparison of different finite descriptions for potentially infinite objects – the languages. The result of Kleene [Kle56] states the equivalence between finite automata and regular expressions for languages of finite words. The proof of Kleene is in fact a constructive one, i.e., it gives two algorithms to transform an automaton into a regular expression and vice versa. The transformation of a finite automaton into an equivalent regular expression is a prototypical example of dynamic programming. The converse transformation is of direct practical consequence e.g. in text processing. For this reason, several methods were proposed within the last decades to find more efficient algorithms, see [Sak05, Sak03] for a survey. The most common construction is this of the standard or position automaton attributed both to Glushkov [Glu61] as well as McNaughton and Yamada [MY60]. In 1964 Brzozowski [Brz64] introduced derivatives of regular expressions and designed an algorithm to find an equivalent deterministic finite automaton. This approach was modified by Antimirov [Ant96] who defined partial derivatives to construct a non-deterministic automaton from a regular expression $E$.

Kleene’s theorem was lifted to the setting of trees [TW68], also cf. [GS97, CDG+07], which are one of the most fundamental concepts in computer science. A regular tree expression defines a language of ordered trees. In this paper, we define partial derivatives for regular tree expressions and build by their help a non-deterministic finite tree automaton recognizing the language denoted by the regular expression. The concept of partial derivatives will yield a tree automaton with at most $\|E\|$ states and $\|E\|^2$ transitions where $\|E\|$
is the number of occurrences of symbols from the ranked alphabet in $E$. The construction of this tree automaton and the correctness proof is combined with algorithmic considerations to build this automaton. We adapt and modify the approach by Champarnaud and Ziadi [CZ01, CZ02] in the word case who extended work of Berry and Sethi [BS86]. Here, we use linearizations of regular tree expressions. The main idea is to distinguish occurrences of the same symbol at different positions in the regular expression. By doing so, we can ensure a certain uniqueness of the partial derivatives. As it turns out, the partial derivatives of the original regular expression are just projections of the partial derivatives of the linearized regular expression. This approach results in two main advantages: Firstly, the desired automaton is in fact a quotient of an automaton that stems from the linearized regular expression. This way we also get the upper bound on the number of transitions mentioned above. Secondly, the theoretical results allow for an efficient algorithm working in the syntax-tree of $E$. We obtain an algorithm with $O(R \cdot |E|^2)$ space and time complexity where $R$ is the maximal rank of a symbol occurring in the finite ranked alphabet $\Sigma$ and $|E|$ is the size of the regular expression.

Beside the standard and the partial derivative construction there are other proposals in the literature how to obtain an automaton from a regular expression. Especially, it would be interesting whether the construction of the follow automaton [HSW01, IY02, CNZ04] carries over to the setting of trees. In this paper we consider ordered trees. However, regular expressions were explored for unordered trees in connection with XML. They are used in pattern matching, see e.g. [HP01]. We wonder if the concept of partial derivatives can lead to fruitful results and algorithms within this area.

2 Trees, automata, and regular expressions

Throughout this paper, we fix a finite ranked alphabet $\Sigma = (\Sigma_m)_{m \geq 0}$. The set $T_\Sigma$ of trees over $\Sigma$ is defined by the syntax

$$t = f(t, t, \ldots, t)$$

$m$ times

where $f \in \Sigma_m$. A subset $L \subseteq T_\Sigma$ is called a tree language.

A tree automaton over $\Sigma$ is a tuple $A = (Q, \Sigma, F, \Delta)$ where $Q$ is a set of states, $F \subseteq Q$ is the set of final states, and $\Delta = (\Delta_m)_{m \in \mathbb{N}}$ is the set of transitions such that $\Delta_m \subseteq Q \times \Sigma_m \times Q^m$ for every $m \in \mathbb{N}$.\footnote{In the term-rewriting terminology employed by [CDG+07], $\Delta$ is the set of rules $f(q_1(x_1), \ldots, q_m(x_m)) \rightarrow q(f(x_1, \ldots, x_m))$.} Especially, $\Delta_0 \subseteq Q \times \Sigma_0$. A finite tree-automaton (or FTA) is a tree automaton $A$ with only finitely many states and, thus, only finitely many transitions (note that there are only finitely many $m$ with $\Sigma_m \neq \emptyset$).

As to whether a tree $t$ is accepted by a tree automaton $A = (Q, \Sigma, F, \Delta)$ is defined inductively along the construction of the tree $t$: if $t = c \in \Sigma_0$, then $t$ is accepted by $A$ iff there exists a transition $(q, c) \in \Delta_0$ with $q \in F$. For $f \in \Sigma_m$ with $m > 0$, the tree $f(t_1, \ldots, t_m)$ is accepted by $A$ iff there exists a transition $(q, f, q_1, \ldots, q_m) \in \Delta_m$ such that...
$q \in F$ and, for $1 \leq i \leq m$, the tree $t_i$ is accepted by the tree automaton $(Q, \Sigma, \{q_i\}, \Delta)$. The language $L(A)$ recognized by $A$ is the set of all trees $t$ that are accepted by $A$. A tree language $L$ is recognizable if there is a FTA $A$ with $L(A) = L$.

It is well-known that a tree language $L$ is recognizable if and only if it can be denoted by a regular expression. A regular expression is defined by the following syntax

$$E = f(E, E, \ldots, E) \mid E + E \mid E \cdot c E \mid E^c$$

where $f \in \Sigma_m$ and $c \in \Sigma_0$. The set of all regular expressions over the ranked alphabet $\Sigma$ is denoted by $\exp(\Sigma)$. Let $|E|$ denote the number of occurrences of the letter $f$ in $E$. The alphabetic width $\|E\|$ of $E$ is the number $\sum_{f \in \Sigma} |E|_f$ of occurrences of symbols from $\Sigma$ in $E$. The size $|E|$ of $E$ is defined inductively by: $|c| = 1$ for $c \in \Sigma_0$, $|f(E_1, \ldots, E_m)| = \sum_{i=1}^m |E_i| + 1$, $|E + F| = |E| + |F| + 1$, and $|E^c| = |E| + 1$. Every regular expression $E$ can be understood as a tree over the ranked alphabet $\Sigma \cup \{+, \cdot, ^c \mid c \in \Sigma_0\}$ where $+$ and $\cdot$ have rank 2 and $^c$ has rank 1. This tree is called the syntax-tree $t_E$ of $E$.

To define the semantics we have to introduce some operations on tree languages. Let $f \in \Sigma_m$ and $L, L_1, \ldots, L_m \subseteq T_{\Sigma}$. Then we put

$$f(L_1, \ldots, L_m) = \{ f(t_1, \ldots, t_m) \mid t_i \in L_i \text{ for } i = 1, \ldots, m \}.$$ 

For $L \subseteq T_{\Sigma}$ and $c \in \Sigma_0$ we define for every $t \in T_{\Sigma}$ inductively the non-uniform substitution $t[c \leftarrow L]$:

- $c[c \leftarrow L] = L$ and $d[c \leftarrow L] = \{ d \}$ for every $d \in \Sigma_0$ with $d \neq c$,
- $f(t_1, \ldots, t_m)[c \leftarrow L] = f(t_1[c \leftarrow L], \ldots, t_m[c \leftarrow L]).$

Then the $c$-product of $L_1, L_2 \subseteq T_{\Sigma}$ is the language $L_1 \cdot c L_2 = \bigcup_{t \in L_1} t[c \leftarrow L_2]$. Now the iterated $c$-products are defined for $L \subseteq T_{\Sigma}$ by

$$L^0 = \{ c \} \text{ and } L^{n+1} = L^n \cdot c L^n.$$ 

The $c$-iteration of $L$ is defined as $L^c = \bigcup_{n \geq 0} L^n$. Now we define the semantics $[E]$ of a regular expression $E$ inductively by

$$[f(E_1, E_2, \ldots, E_m)] = f([E_1], [E_2], \ldots, [E_m]), \quad [E + F] = [E] \cup [F],$$

$$[E \cdot c F] = [E] \cdot [F], \text{ and } [E^c] = [E]^c.$$ 

For a set $M$ of regular expressions, we put $[M] = \bigcup_{E \in M} [E]$.

### 3 A direct construction

In this section, we will construct from a regular expression $E$ a tree automaton $A_E$ that accepts $[E]$. The finiteness of this automaton will only be proved later. Our construction is based on partial derivates that we have to define and investigate later.
Let $M$ be a set of regular expressions, $F$ some regular expression, and $c \in \Sigma$. Then $M \cdot_c F$ denotes the set
\[ M \cdot_c F = \{ E \cdot_c F \mid E \in M \} . \]
Similarly, we put for a set $\mathcal{M}$ of $m$-tuples of regular expressions
\[ \mathcal{M} \cdot_c F = \{(E_1 \cdot_c F, E_2 \cdot_c F, \ldots, E_m \cdot_c F) \mid (E_1, E_2, \ldots, E_m) \in \mathcal{M} \} . \]
Let $\Sigma_{\geq} = \bigcup_{m \geq 1} \Sigma_m = \Sigma \setminus \Sigma_0$ denote the set of non-constant symbols from the ranked alphabet $\Sigma$.

**Definition 1** For $g \in \Sigma_{\geq}$ and a regular expression $E$, we define the sets $g^{-1}(E)$ of $m$-tuples of regular expressions inductively by
\[ g^{-1}(f(E_1, E_2, \ldots, E_n)) = \begin{cases} \{(E_1, E_2, \ldots, E_n)\} & \text{if } f = g \\ \emptyset & \text{if } f \neq g \end{cases} \]
\[ g^{-1}(E + F) = g^{-1}(E) \cup g^{-1}(F) \]
\[ g^{-1}(E \cdot_c F) = \begin{cases} g^{-1}(E) \cdot_c F & \text{if } c \notin [E] \\ g^{-1}(E) \cdot_c F \cup g^{-1}(F) & \text{otherwise} \end{cases} \]
\[ g^{-1}(E^*) = g^{-1}(E) \cdot_c E^*. \]

Following Antimirov, we define further functions $\partial_w$ for finite words $w \in \Sigma_{\geq}$ over the alphabet $\Sigma_{\geq}$. By $\varepsilon$ we denote the empty word.:

**Definition 2** Let $E$ be a regular expression. Then $\partial_\varepsilon(E) = \{E\}$ and, for $w \in \Sigma_{\geq}$ and $g \in \Sigma_{\geq}$, the set $\partial_{wg}(E)$ consists of all regular expressions $F$ that appear in some tuple from $g^{-1}(E')$ for some $E' \in \partial_w(E)$. For a set of words $W \subseteq \Sigma_{\geq}$, and a regular expression $E$, we put $\partial_W(E) = \bigcup_{w \in W} \partial_w(E)$.

The function $\partial_w$ is called the partial derivative w.r.t. $w$.

Note that $\partial_{wg}(E) = \partial_g(\partial_w(E)) = \bigcup_{E' \in \partial_w(E)} \partial_g(E')$ for all $w \in \Sigma_{\geq}$ and $g \in \Sigma_{\geq}$. Further note that we consider derivatives with respect to words over the non-constant symbols from $\Sigma$ and not with regard to trees.

A symbol $f \in \Sigma$ occurs unguarded in $E$ if it is not preceded by another symbol of $\Sigma$. We will be interested in the number $\langle E \rangle_f$ of unguarded occurrences of $f$ in $E$ that can be computed inductively:
\[ \langle f(E_1, \ldots, E_m) \rangle_f = 1 \text{ and } \langle g(E_1, \ldots, E_m) \rangle_f = 0 \text{ for } g \neq f, \]
\[ \langle E_1 + E_2 \rangle_f = \langle E_1 \cdot_c E_2 \rangle_f = \langle E_1 \rangle_f + \langle E_2 \rangle_f, \text{ and} \]
\[ \langle E^* \rangle_f = \langle E \rangle_f. \]

**Proposition 3** Let $E$ be a regular expression and $g \in \Sigma_{\geq}$. Then $|g^{-1}(E)| \leq \langle E \rangle_g$. Especially, if $|E|_g = 0$ then $g^{-1}(E) = \partial_g(E) = \emptyset$.  

baeumeneu (version as of May 2, 2008) 4
Proof. The claim is shown by induction on the construction of $E$: If $E = f(E_1, \ldots, E_m)$ and $f \neq g$, then $|g^{-1}(E)| = 0$, so the claim is trivial. If $f = g$, then $|g^{-1}(E)| = 1 = \langle E \rangle_g$. Next consider the case $E + F$: then $|g^{-1}(E + F)| \leq |g^{-1}(E)| + |g^{-1}(F)| \leq \langle E \rangle_g + \langle F \rangle_g = \langle E + F \rangle_g$. For the product, we have $|g^{-1}(E \cdot F)| \leq |g^{-1}(E) \cdot F| + |g^{-1}(F)| \leq |g^{-1}(E)| + |g^{-1}(F)| \leq \langle E \rangle_g + \langle F \rangle_g = \langle E \cdot F \rangle_g$. Similarly, we obtain for the iteration $|g^{-1}(E^*)| = |g^{-1}(E) \cdot E^*| = |g^{-1}(E)| \leq \langle E \rangle_g = \langle E^* \rangle_g$. 

Next, we express the semantics of a regular expression $[E]$ in terms of the semantics of the tuples from $g^{-1}(E)$.

Proposition 4 For any regular expression $E$, we have

$$[E] = \bigcup \{g([G_1], \ldots, [G_m]) \mid g \in \Sigma, (G_1, \ldots, G_m) \in g^{-1}(E)\} \cup \{c \in \Sigma_0 \mid c \in [E]\}.$$  

Proof. Let $[E]_0 = [E] \cap \Sigma_0$. The proof proceeds by induction on the construction of the regular expression $E$. First let $E = f(E_1, \ldots, E_m)$ where $f \in \Sigma_m$ and $E_1, \ldots, E_m$ are regular expressions. We put $\overline{G} = (G_1, \ldots, G_m)$. Then

$$[E] = f([E_1], \ldots, [E_m])$$
$$= \bigcup \{f([G_1], \ldots, [G_m]) \mid \overline{G} \in f^{-1}(E)\} \text{ since } f^{-1}(E) = \{(E_1, \ldots, E_m)\}$$
$$= \bigcup \{g([G_1], \ldots, [G_m]) \mid g \in \Sigma, \overline{G} \in g^{-1}(E)\} \text{ since } g^{-1}(E) = \emptyset \text{ for } f \neq g.$$ 

The proof in case $E = E_1 + E_2$ is immediate and therefore omitted. Next let $E = E_1 \cdot_c E_2$. Then we have

$$[E] = (\langle E_1 \rangle \cdot_c \langle E_2 \rangle)$$
$$= (\langle E_1 \rangle \setminus \{c\}) \cdot_c \langle E_2 \rangle \cup (\langle E_1 \rangle \cap \{c\}) \cdot_c \langle E_2 \rangle.$$ 

By the induction hypothesis, the first of these two sets equals

$$\left(\bigcup \{f([G_1], \ldots, [G_m]) \mid f \in \Sigma, \overline{G} \in f^{-1}(E_1)\}\right) \cdot_c \langle E_2 \rangle \cup (\langle E_1 \rangle_0 \setminus \{c\})$$
$$= \bigcup \{f([G_1 \cdot_c E_2], \ldots, [G_m \cdot_c E_2]) \mid f \in \Sigma, \overline{G} \in f^{-1}(E_1)\} \cup (\langle E_1 \rangle_0 \setminus \{c\})$$
$$= \bigcup \{f([H_1], \ldots, [H_m]) \mid f \in \Sigma, \overline{H} \in f^{-1}(E_1) \cdot_c \langle E_2 \rangle\} \cup (\langle E_1 \rangle_0 \setminus \{c\}).$$

If $c \in \langle E_1 \rangle_0$, then the second of the two sets above equals $[E_2] = \bigcup \{f([H_1], \ldots, [H_m]) \mid f \in \Sigma, \overline{H} \in f^{-1}(E_2)\} \cup \langle E_2 \rangle_0$, otherwise it is empty. Hence Eq. 1 holds for $E = E_1 \cdot_c E_2$.

Finally, consider the regular expression $E^{\ast c}$. If $f(t_1, \ldots, t_m) \in [E^{\ast c}] = [E]^\ast_c$, then there exists $n \geq 0$ with $f(t_1, \ldots, t_m) \in [E]^{n+1,c} \setminus [E]^n_c$. Hence there exists $s \in [E]$ with $f(t_1, \ldots, t_m) \in \{s\} \cdot_c [E]^n_c$. Since $f(t_1, \ldots, t_m) \notin [E]^n_c$, the term $s$ is of the form
By the induction hypothesis, we find \((G_1, \ldots, G_m) \in f^{-1}(E)\) with \(s_i \in [G_i]\). Hence we obtain
\[
f(t_1, \ldots, t_m) \in \{s\} \cdot _c [E^{cc}]
\subseteq f([G_1], \ldots, [G_m]) \cdot _c [E^{cc}]
= f([G_1 \cdot _c E^{cc}], \ldots, [G_m \cdot _c E^{cc}]).
\]
Since the tuple \((G_i \cdot _c E^{cc})_{1 \leq i \leq m} = (G_i)_{1 \leq i \leq m} \cdot _c E^{cc}\) belongs to \(f^{-1}(E^{cc})\), we showed the containment \(\subseteq\) of Eq. 1.
Conversely let \(f \in \Sigma\) and \(\overline{H} \in f^{-1}(E^{cc}) = f^{-1}(E) \cdot _c E^{cc}\). Then there exists a tuple of regular expressions \(\overline{G} \in f^{-1}(E)\) with \(\overline{H} = \overline{G} \cdot _c E^{cc}\). Hence we get
\[
f([H_1], \ldots, [H_m]) = f([G_1 \cdot _c E^{cc}], \ldots, [G_m \cdot _c E^{cc}])
= f([G_1], \ldots, [G_m]) \cdot _c E^{cc}
\]
By the induction hypothesis, \(f([G_1], \ldots, [G_m]) \subseteq [E]\), so we can continue
\[
\subseteq [E] \cdot _c [E^{cc}] \subseteq [E^{cc}].
\]

\(\square\)

Let \(E\) be a regular expression and let \(Q_E = \partial_{\Sigma^{\geq 1}}(E)\). Then we define a set of transitions \(\Delta_E\) setting
\[
\Delta_E = \{(F, f, (G_1, G_2, \ldots, G_m)) \mid F \in Q_E, f \in \Sigma_m, m \geq 1, (G_1, \ldots, G_m) \in f^{-1}(F)\}
\cup \{(F, c) \mid F \in Q_E, c \in \Sigma_0, c \in [F]\}.
\]
Furthermore, let \(A_E = (Q_E, \Sigma, \{E\}, \Delta_E)\) denote the tree automaton whose only final state is the regular expression \(E\).

**Theorem 5** Let \(E\) be a regular expression over the ranked alphabet \(\Sigma\). Then \(A_E\) is a tree automaton that accepts \([E]\).

**Proof.** We show by induction on \(n \in \mathbb{N}\): for all trees \(t = f(s_1, \ldots, s_m)\) of size \(n\) and all regular expressions \(F\), the tree automaton \(A_F\) accepts \(t\) iff \(t \in [F]\).

Since there are no trees of size 0, the base case is trivial. So let \(t = f(s_1, \ldots, s_m)\) for some \(m \geq 0\). For \(m = 0\) we have \(t = c \in \Sigma_0\). Now \(c\) is accepted by \(A_F\) iff there is a transition \((F, c) \in \Delta_F\). But this is the case iff \(c \in [F]\). Now suppose \(m > 0\). Then \(t\) is accepted by \(A_F\) iff there exists a transition \((F, f, (G_1, \ldots, G_m)) \in \Delta_F\) such that \(s_i\) is accepted by the TA \((Q_F, \Sigma, \{G_i\}, \Delta_F)\) for all \(1 \leq i \leq m\). Note that the reachable part of the automaton \((Q_F, \Sigma, \{G_i\}, \Delta_F)\) is the set of states \(Q_{G_i}\). Hence, \(s_i\) is accepted by this automaton iff it is accepted by \(A_{G_i}\). By the induction hypothesis, this is equivalent to saying \(s_i \in [G_i]\). Since this holds for all \(1 \leq i \leq m\), we have that \(t\) is accepted by \(A_F\) iff

\[\text{baeumeneu (version as of May 2, 2008)}\]
there exists \((G_1, \ldots, G_m) \in f^{-1}(F)\) with \(s_i \in [G_i]\) which is, by Proposition 4 equivalent to saying \(t \in [F]\).

So far, we did not prove that the tree automaton \(A_E\) has only finitely many states, i.e., that \([E]\) is recognizable. We will actually prove that the number of states is linear and that the number of transitions is quadratic in the size of \(E\). This will only be achieved after going through the following two constructions.

### 4 An indirect construction via linearizations

The idea of the indirect construction is as follows: In a regular expression \(E\), uniquely mark the occurrences of letters from \(\Sigma_{\geq 1}\). Then we apply our direct construction to the resulting regular expression \(\overline{E}\). The projection of this automaton accepts \([E]\). As it turns out, a quotient of the automaton one obtains this way is isomorphic to the result of the direct construction.

#### 4.1 Linear regular expressions

A regular expression \(E\) is linear if every letter \(f \in \Sigma_{\geq 1}\) occurs at most once in \(E\). Note that \(c \in \Sigma_0\) may occur more than once. The following proposition is a consequence of Proposition 3.

**Proposition 6** Let \(E\) be a linear regular expression and \(g \in \Sigma_m\) for \(m \geq 1\). Then \(|g^{-1}(E)| \leq 1\) and therefore \(|\partial_g(E)| \leq m\).

We consider partial derivatives w.r.t. non-empty words for linear regular expressions:

**Proposition 7** Let \(E, F\) be linear regular expressions over the alphabet \(\Sigma\) such that also \(E + F\) and \(E \cdot c\cdot F\) are linear. Let \(w \in \Sigma_{\geq 1}^*\) and \(g \in \Sigma_{\geq 1}\). Then the following hold true:

- \(g^{-1}(\partial_w(E + F)) = \begin{cases} g^{-1}(\partial_w(E)) & \text{if } |E|_g > 0, \\ g^{-1}(\partial_w(F)) & \text{otherwise,} \end{cases}\)
- \(g^{-1}(\partial_w(E \cdot c\cdot F)) = \begin{cases} g^{-1}(\partial_w(E)) \cdot c\cdot F & \text{if } |E|_g > 0, \\ \bigcup \{g^{-1}(\partial_w(F)) \mid \exists u \in \Sigma_{\geq 1}^* : w = uv \text{ and } c \in \llbracket \partial_u(E) \rrbracket\} & \text{otherwise,} \end{cases}\)
- There are suffixes \(v_1, \ldots, v_k\) of \(w\) such that \(g^{-1}(\partial_w(E^{*c})) = \bigcup_{1 \leq i \leq k} g^{-1}(\partial_{v_i}(E)) \cdot c\cdot E^{*c}\).

**Proof.** If \(|E|_g = 0\), then \(|\partial_w(E)|_g = 0\) implying, by Proposition 3, \(g^{-1}(\partial_w(E)) = \emptyset\). Now \(g^{-1}(\partial_w(E + F)) = g^{-1}(\partial_w(E)) \cup g^{-1}(\partial_w(F))\). If \(|E|_g > 0\), then \(g^{-1}(\partial_w(F)) = \emptyset\) and \(g^{-1}(\partial_w(E + F)) = g^{-1}(\partial_w(E))\). Otherwise we get \(g^{-1}(\partial_w(E)) = \emptyset\) and \(g^{-1}(\partial_w(E + F)) = g^{-1}(\partial_w(F))\).
The remaining claims are obvious for \( |w| = 0 \). From now on let \( w = \overline{w} f \) for some \( \overline{w} \in \Sigma_{\geq 1}^* \) and \( f \in \Sigma_{\geq 1} \). By the induction hypothesis, we obtain
\[
\begin{aligned}
f^{-1}(\partial_{\overline{w}}(E \cdot_c F)) &= \begin{cases} 
    f^{-1}(\partial_{\overline{w}}(E)) \cdot_c F & \text{if } |E|_f > 0, \\
    \bigcup \{f^{-1}(\partial_c(F)) \mid \exists u \in \Sigma_{\geq 1}^*: \overline{w} = uv & c \in \llbracket \partial_u(E) \rrbracket \} & \text{otherwise.}
\end{cases}
\end{aligned}
\]

Firstly, consider the case \( |E|_f > 0 \) and \( |E|_g > 0 \). Then \( |F|_f = |F|_g = 0 \) and therefore \( g^{-1}(F) = \emptyset \). Hence we have \( g^{-1}(\partial_w(E) \cdot_c F) = g^{-1}(\partial_w(E)) \cdot_c F \).

Next, let \( |E|_f > |E|_g = 0 \). Then (i) \( g^{-1}(\partial_w(E)) = \emptyset \) and (ii) \( \partial_v(F) = \emptyset \) for all non-empty suffixes \( v \) of \( w \) (since \( f \) occurs in \( v \) but not in \( F \)). Since, by the induction hypothesis, \( f^{-1}(\partial_{\overline{w}}(E \cdot_c F)) = f^{-1}(\partial_{\overline{w}}(E)) \cdot_c F \), we get \( \partial_w(E \cdot_c F) = \partial_f(\partial_{\overline{w}}(E \cdot_c F)) = \partial_f(\partial_{\overline{w}}(E)) \cdot_c F = \partial_w(E) \cdot_c F \) and therefore
\[
\begin{aligned}
g^{-1}(\partial_w(E \cdot_c F)) &= g^{-1}(\partial_w(E)) \cdot_c F \\
&= \begin{cases} 
    g^{-1}(\partial_w(E)) \cdot_c F & \text{if } c \notin \llbracket \partial_w(E) \rrbracket \\
    g^{-1}(\partial_w(E)) \cdot_c F \cup g^{-1}(F) & \text{otherwise}
\end{cases}
\end{aligned}
\]

\[
\begin{aligned}
&= \begin{cases} 
    \emptyset & \text{if } c \notin \llbracket \partial_w(E) \rrbracket \\
    g^{-1}(F) & \text{otherwise}
\end{cases} \\
&= \bigcup \{g^{-1}(\partial_v(F)) \mid \exists u \in \Sigma_{\geq 1}^*: w = uv & c \in \llbracket \partial_u(E) \rrbracket \}
\end{aligned}
\]

as required.

Now assume \( |E|_f = 0 \). Then the induction hypothesis implies \( f^{-1}(\partial_{\overline{w}}(E \cdot_c F)) = \bigcup \{f^{-1}(\partial_v(F)) \mid \exists u : \overline{w} = uv, c \in \llbracket \partial_u(E) \rrbracket \} \). Hence we obtain
\[
\begin{aligned}
\partial_w(E \cdot_c F) &= \partial_f(\partial_{\overline{w}}(E \cdot_c F)) \\
&= \bigcup \{\partial_f(\partial_v(F)) \mid \exists u : \overline{w} = uv, c \in \llbracket \partial_u(E) \rrbracket \} \\
&= \bigcup \{\partial_v(F) \mid \exists u : w = uv, v \neq \varepsilon, c \in \llbracket \partial_u(E) \rrbracket \} \\
&= \bigcup \{\partial_v(F) \mid \exists u : w = uv, c \in \llbracket \partial_u(E) \rrbracket \}
\end{aligned}
\]

where the last equality holds since \( \partial_w(E) = \emptyset \). Applying \( g^{-1} \) to this equation yields
\[
\begin{aligned}
g^{-1}(\partial_w(E \cdot_c F)) &= \bigcup \{g^{-1}(\partial_v(F)) \mid \exists u : w = uv, c \in \llbracket \partial_u(E) \rrbracket \}
\end{aligned}
\]

If \( |E|_g = 0 \), this is precisely what we wanted to show. Otherwise, we obtain \( |F|_g = 0 \) and therefore \( g^{-1}(\partial_v(F)) = \emptyset \) for all \( v \in \Sigma_{\geq 1}^* \). Hence, in this case the last expression equals \( \emptyset \). Since also \( g^{-1}(\partial_w(E)) = \emptyset \) (due to the non-occurrence of \( f \) in \( E \)), this equals \( g^{-1}(\partial_w(E) \cdot_c F) \) as required. This shows the claim for \( E \cdot_c F \).
Now consider the regular expression $E^{sc}$. By the induction hypothesis, there are suffixes $v_1, \ldots, v_k$ of $w$ such that we have

$$g^{-1}\partial_w(E^{sc}) = g^{-1}\left( \bigcup_{1 \leq i \leq k} \partial_{v_i}(E) \cdot c\ E^{sc} \right)$$

$$= \begin{cases} \bigcup_{1 \leq i \leq k} g^{-1}(\partial_{v_i}(E)) \cdot c\ E^{sc} & \text{if } c \notin \Vert \partial_{v_i}(E) \Vert \text{ for all } 1 \leq i \leq k, \\ \bigcup_{1 \leq i \leq k} g^{-1}(\partial_{v_i}(E)) \cdot c\ E^{sc} \cup g^{-1}(E^{sc}) & \text{otherwise.} \end{cases}$$

Since $g^{-1}(E^{sc}) = g^{-1}(E) \cdot c\ E^{sc} = g^{-1}(\partial_{v}(E)) \cdot c\ E^{sc}$, the set of tuples of regular expressions $g^{-1}(\partial_w(E^{sc}))$ is of the required form. 

Proposition 8 Let $E$ be a linear regular expression, $u, w \in \Sigma^*$, and $g \in \Sigma_m$ with $m \geq 1$. Then we have:

1. $|\partial_{uw}(E)| \leq m$,
2. if $\partial_{uw}(E) \neq \emptyset$ and $\partial_{uw}(E) \neq \emptyset$, then $\partial_{uw}(E) = \partial_{uw}(E)$,
3. if $g^{-1}(\partial_{w}(E)) \neq \emptyset$ and $g^{-1}(\partial_{w}(E)) \neq \emptyset$, then $g^{-1}(\partial_{w}(E)) = g^{-1}(\partial_{w}(E))$.

Proof. Note that the second claim is a consequence of the third one. The proof of the first statement can easily be extracted from our proof of the third one.

First consider the case $E = f(E_1, \ldots, E_n)$. Since $E$ is linear, there is at most one $i$ with $|E_i|_g > 0$, if no such $i$ exists, set $i = 1$. Then we have

$$g^{-1}\partial_u(E) = \begin{cases} g^{-1}\partial_{u}(\{E_1, \ldots, E_n\}) = g^{-1}\partial_{u}(E_i) & \text{if } u = f u' \text{ since } |E_j|_g = 0 \text{ for } j \neq i, \\
\emptyset & \text{if } u = \varepsilon \& f = g, \\
& \text{otherwise} \end{cases}$$

and similarly

$$g^{-1}\partial_w(E) = \begin{cases} g^{-1}\partial_{w}(E_i) & \text{if } u = f u', \\
\{E_1, \ldots, E_n\} & \text{if } w = \varepsilon \& f = g, \\
\emptyset & \text{otherwise} \end{cases}$$

Recall that by assumption $g^{-1}\partial_u(E) \neq \emptyset$ and $g^{-1}\partial_w(E) \neq \emptyset$. If $f = g$ is the first letter of $u$, then $\emptyset \neq g^{-1}\partial_u(E) = g^{-1}\partial_{u'}(E_i) = \emptyset$ since $E$ is linear, a contradiction. Hence either $f \neq g$ is the first letter of $u$ or $f = g$ and $u$ is empty. Since the analogous holds for $w$, we obtain $u = \varepsilon$ iff $w = \varepsilon$. Now the claim follows immediately from the induction hypothesis.

For $E = E_1 + E_2$ the claim is immediate by the last proposition and the induction hypothesis.
Let $E = E_1 \cdot_c E_2$. If $|E_1|_g > 0$, then $g^{-1}\partial_u(E) = g^{-1}\partial_u(E_1) \cdot_c E_2$ as well as $g^{-1}\partial_u(E) = g^{-1}\partial_u(E_1) \cdot_c E_2$. Since these two sets are non-empty, so are the sets $g^{-1}\partial_u(E_1)$ and $g^{-1}\partial_u(E_1)$. Hence, by the induction hypothesis, the claim follows.

Suppose now $|E_1|_g = 0$. Then $g^{-1}\partial_u(E)$ is a finite union of sets of the form $g^{-1}\partial_u(E_2)$. The induction hypothesis implies that any two non-empty of them are equal, i.e., $g^{-1}\partial_u(E) = g^{-1}\partial_u(E_2)$ for some word $u'$. Similarly, $g^{-1}\partial_u(E) = g^{-1}\partial_u(E_2)$ for some word $w'$. Now the claim follows from the induction hypothesis.

A similar argument can be applied in case $E = \mathcal{F}^g$ with $g^{-1}\partial_u(F) \cdot \mathcal{F}^g$ in place of $g^{-1}\partial_u(E_2)$.

By Propositions 7 and 8 we conclude

**Corollary 9** For a linear regular expression $E$ and $w \in \Sigma_{\geq 1}^+$ we have $\partial_w(E^g) = \partial_w(E) \cdot_c E^g$ for some non-empty suffix $u$ of $w$.

Next, we bound the number of partial derivatives of a linear regular expression.

**Proposition 10** Let $E$ be a linear regular expression. Then $|\partial_{\Sigma_{\geq 1}^+}(E)| \leq \|\mathcal{E}\| - 1$ and $|\partial_{\Sigma_{\geq 1}^*}(E)| \leq \|\mathcal{E}\|$.

**Proof.** Note that $\partial_{\Sigma_{\geq 1}^+}(E) = \partial_{\Sigma_{\geq 1}^+}(E) \cup \{E\}$. By induction on $E$ we get: For $E = f(E_1, \ldots, E_n)$, $g \in \Sigma_{\geq 1}$, and $u \in \Sigma_{\geq 1}^*$

$$\partial_{gw}(E) = \begin{cases} \partial_u(\{E_1, \ldots, E_n\}) & \text{if } g = f, \\ \emptyset & \text{if } g \neq f. \end{cases}$$

Hence, $|\partial_{\Sigma_{\geq 1}^+}(E)| \leq \sum_{i=1}^n |\partial_{\Sigma_{\geq 1}^+}(E_i)| \leq \sum_{i=1}^n |\|E_i\| - 1$. For $E = E_1 + E_2$ we use Proposition 7 and the induction hypothesis and obtain the assumption. If $E = E_1 \cdot_c E_2$, then again by Proposition 7: $|\partial_{\Sigma_{\geq 1}^+}(E)| \leq |\partial_{\Sigma_{\geq 1}^+}(E_1)| + |\partial_{\Sigma_{\geq 1}^+}(E_2)| \leq \|E_1\| - 1 + \|E_2\| - 1 < \|\mathcal{E}\| - 1$. Finally, we conclude by Corollary 9 that $|\partial_{\Sigma_{\geq 1}^+}(E^g)| \leq |\partial_{\Sigma_{\geq 1}^+}(E)| \leq \|\mathcal{E}\| - 1 = \|\mathcal{E}^g\| - 1$. 

**Proposition 11** Let $E$ be a linear regular expression, $w \in \Sigma_{\geq 1}^*$, and $g \in \Sigma_{\geq 1}$. Then $|g^{-1}(\partial_w(E))| \leq 1$.

**Proof.** We proceed by induction on the construction of the linear regular expression $E$. For $E = f(E_1, \ldots, E_n)$ and $w = \varepsilon$ the claim is obvious. Otherwise, note that $g$ appears at most in one of the regular expressions $E_i$ and thus for $w = f v$ we have by induction hypothesis $|g^{-1}(\partial_w(E))| \leq |g^{-1}(\partial_v(E_i))| \leq 1$. For $E = E_1 + E_2$ the letter $g$ can appear at most in either $E_1$ or in $E_2$. Now the claim follows by Proposition 3 and the induction hypothesis. The case $E = E_1 \cdot_c E_2$ is similar. Now consider $E = F^g$. For $w = \varepsilon$ we are
done by induction hypothesis on \( F \). Otherwise, due to Corollary 9, \( \partial_w(E) = \partial_u(F) \cdot_c F^{sc} \) for some suffix \( u \) of \( w \) and thus

\[
g^{-1}(\partial_w(E)) = g^{-1}(\partial_u(F) \cdot_c F^{sc}) \\
\subseteq g^{-1}(\partial_u(F)) \cdot_c F^{sc} \cup g^{-1}(F^{sc}) \\
= g^{-1}(\partial_u(F)) \cdot_c F^{sc} \cup g^{-1}(F) \cdot_c F^{sc}.
\]

But due to Proposition 8, if \( g^{-1}(\partial_u(F)) \neq \emptyset \neq g^{-1}(F) = g^{-1}(\partial_c(F)) \), then both are equal. Hence, \(|g^{-1}(\partial_w(F^{sc}))| \leq 1 \) by induction hypothesis.

\[\square\]

4.2 The projection construction

Recall that Theorem 5 provides a possibly infinite tree automaton \( \mathcal{A}_E \) that accepts \([E]\). Assuming \( E \) to be linear, we are now in the position to improve this result:

**Corollary 12** Let \( E \) be a linear regular expression over the ranked alphabet \( \Sigma \). Then \( \mathcal{A}_E \) is a finite tree automaton with at most \( \|E\|\cdot|\Sigma| \) many transitions that accepts \([E]\).

**Proof.** The equality \( L(\mathcal{A}_E) = [E] \) was shown in Theorem 5. Since the set of states of \( \mathcal{A}_E \) equals \( \partial_{\Sigma_1}(E) \), the finite tree automaton has at most \( \|E\| \) many states by Proposition 10. For \( f \in \Sigma_{\geq 1} \) and \( D \in Q_E \), there is at most one transition of the form \((D,f,(G_1,\ldots,G_m))\) by Proposition 11, i.e., there are at most \( \|E\| \cdot |\Sigma_{\geq 1}| \) many transitions whose label belongs to \( \Sigma_{\geq 1} \). In addition, there can be \(|Q_E \times \Sigma_0| \leq \|E\| \cdot |\Sigma_0| \) many transitions of the form \((D,c)\) with \( c \in \Sigma_0 \).

Note that given two alphabets \( \Gamma \) and \( \Sigma \) with \( \Gamma_0 \subseteq \Sigma_0 \) and a mapping \( \eta : \Gamma \rightarrow \Sigma \) with \( \eta(\Gamma_m) \subseteq \Sigma_m \) for every \( m \in \mathbb{N} \), we can extend \( \eta \) naturally to \( \eta : \exp(\Gamma) \rightarrow \exp(\Sigma) \) by:

- \( \eta(f(E_1,\ldots,E_m)) = \eta(f)(\eta(E_1),\ldots,\eta(E_m)) \),
- \( \eta(E + F) = \eta(E) + \eta(F) \), \( \eta(E \cdot_c F) = \eta(E) \cdot_c \eta(F) \), and \( \eta(F^{sc}) = (\eta(E))^{sc} \).

**Definition 13** Let \( E \) be a regular expression over the ranked alphabet \( \Sigma \). A linear regular expression \( \overline{E} \) is a linearization of \( E \) w.r.t. \( \eta \) over the ranked alphabet \( \overline{\Sigma} \) if \( \eta : \overline{\Sigma} \rightarrow \Sigma \) is a mapping with \( \eta(\Sigma_m) \subseteq \Sigma_m \) such that \( \eta(c) = c \) for every \( c \in \Sigma_0 \) and \( \eta(\overline{E}) = E \).

Note that both the constants from \( \Sigma_0 \) and the operations \( \cdot_c \) and \( ^{sc} \) remain unchanged. By abuse of notation, we denote also the two natural continuations of \( \eta \) to \( \overline{\Sigma}^* \) and to \( T_{\overline{\Sigma}} \) by \( \eta \).

The following lemma is easily shown:

**Lemma 14** Let \( E \) be a regular expression and \( \overline{E} \) a linearization of \( E \) w.r.t. \( \eta \). Then \( \eta([\overline{E}]) = [E] \).
Let $E$ be an arbitrary regular expression. Then one can construct a small finite tree automaton $\mathcal{A}_E$ accepting $[E]$ as follows: firstly, construct some linearization $\overline{E}$ of $E$ w.r.t. $\eta$ (we can assume that every symbol from $\overline{\Sigma}$ appears in $\overline{E}$ and therefore $|\overline{\Sigma}| \leq \|\overline{E}\| = \|E\|$). Secondly, build the finite tree automaton $\mathcal{A}_{\overline{E}}$ which then has at most $\|\overline{E}\| = \|E\|$ many states and $\|\overline{E}\| \cdot |\overline{\Sigma}| \leq \|E\|^2$ many transitions. Thirdly, replace the transitions $(\overline{F}, \overline{f}, (\overline{G}_1, \ldots, \overline{G}_m))$ of this automaton by $(\overline{F}, \eta(\overline{f}), (\overline{G}_1, \ldots, \overline{G}_m))$. Then, by Lemma 14, the following is immediate:

**Corollary 15** Let $E$ be a regular expression. Then $\mathcal{A}_E$ is a finite tree automaton with at most $\|E\|$ many states and $\|E\|^2$ many transitions that accepts $[E]$.

### 4.3 The quotient construction

We will now identify some of the states of the automaton $\mathcal{A}_E$. The resulting automaton will turn out to be isomorphic to the automaton $\mathcal{A}_E$ from our first construction.

We define the following equivalence relation $\sim$ on $Q_{\overline{E}}$:

$$\overline{F} \sim \overline{H} : \iff \eta(\overline{F}) = \eta(\overline{H}).$$

Let $\mathcal{G}_i, \mathcal{H}_i \in Q_{\overline{E}}$ with $\mathcal{G}_i \sim \mathcal{H}_i$ for $i = 1, \ldots, m$ and $\overline{f}_1, \overline{f}_2 \in \Sigma_{\geq 1}$ with $\eta(\overline{f}_1) = \eta(\overline{f}_2) = f$. Then $\overline{f}_1(\mathcal{G}_1, \ldots, \mathcal{G}_m) \sim \overline{f}_2(\mathcal{H}_1, \ldots, \mathcal{H}_m)$. Hence, $\sim$ is a congruence relation. We denote the congruence class of $\mathcal{G} \in Q_{\overline{E}}$ by $[\mathcal{G}]$. Since $\sim$ is a congruence, the following quotient FTA is well-defined: $\tilde{\mathcal{A}}_E = \left( Q_{\overline{E}} / \sim, \Sigma, \{[E]\}, \Delta'_E \right)$ where

$$\Delta'_E = \left\{ ([\overline{F}], f, ([\mathcal{G}_1], \ldots, [\mathcal{G}_m]) \mid ([\overline{F}], f, (\mathcal{G}_1, \ldots, \mathcal{G}_m)) \in \Delta_E \right\}.$$

We will show that the FTA $\tilde{\mathcal{A}}_E$ is isomorphic to $\mathcal{A}_E$ and, thus, in particular accepts the language $[E]$. Therefore, we have to clarify that $\eta(\overline{F}) \in Q_E = \partial_{\Sigma_{\geq 1}}(E)$ for every $\overline{F} \in Q_{\overline{E}}$. We will show the following fundamental relation between the partial derivatives of $E$ and of $\overline{E}$, but first for partial derivatives w.r.t. a single letter $g$:

**Proposition 16** Let $E$ be a regular expression over the ranked alphabet $\Sigma$ and $\overline{E}$ a linearization of $E$ w.r.t. $\eta$. Then we have for every $g \in \Sigma_{\geq 1}$

$$g^{-1}(E) = \bigcup_{\overline{g} \in \eta^{-1}(g)} \eta(\overline{g}^{-1}(E)) \quad \text{and} \quad \partial_g(E) = \bigcup_{\overline{g} \in \eta^{-1}(g)} \eta(\partial_{\overline{g}}(E)).$$

**Proof.** Here, we prove the second equation. The proof for the first one runs along the same line. We proceed by induction on the construction of $E$. Let $E = f(E_1, \ldots, E_n)$. If $g \neq f$, then $\partial_g(E) = \emptyset = \partial_g(\overline{E})$ for every $\overline{g} \in \eta^{-1}(g)$. So let $g = f$. Now $\partial_f(E) = \{E_1, \ldots, E_n\}$ on the one hand. On the other hand, $\overline{E} = \overline{f}(\overline{E}_1, \ldots, \overline{E}_n)$ for some $\overline{f} \in \eta^{-1}(f)$ and some linearizations $\overline{E}_i$ of $E_i$ w.r.t. $\eta$. Hence $\eta(\partial_{\overline{f}}(\overline{E})) = \{E_1, \ldots, E_n\}$. Since $\partial_{\overline{f}_1}(\overline{E}) = \emptyset$ for all $f_1 \neq \overline{f}$, the equality follows for $E = f(E_1, \ldots, E_n)$. 


Now, let $E = E_1 + E_2$. Then there are linearizations $\overline{E_1}$ and $\overline{E_2}$ of $E_1$ and $E_2$ w.r.t. $\eta$ such that $E = \overline{E_1} + \overline{E_2}$. The partial derivatives of $E$ w.r.t. $g$ are as follows:

\[
\partial_g(E_1 + E_2) = \partial_g(E_1) \cup \partial_g(E_2) = \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E_1})) \cup \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E_2}))
\]

\[
= \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E_1}) \cup \partial_{\eta}(\overline{E_2})) = \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E})).
\]

We turn to $E = E_1 \cdot c E_2$. Again, there are linearizations $\overline{E_1}$ and $\overline{E_2}$ of $E_1$ and $E_2$ w.r.t. $\eta$ such that $E = \overline{E_1} \cdot c \overline{E_2}$. First suppose $c \notin \llbracket E_1 \rrbracket$. Since $\eta^{-1}(c) = \{c\}$, Lemma 14 implies $c \notin \llbracket E_1 \rrbracket$. Hence, we have for this case:

\[
\partial_g(E_1 \cdot c E_2) = \partial_g(E_1) \cdot c E_2 = \left( \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E_1})) \right) \cdot c E_2 = \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E_1}) \cdot c \overline{E_2})
\]

\[
= \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E_1})\cdot c \overline{E_2})) = \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E})).
\]

The case of $c \in \llbracket E_1 \rrbracket$ is similar. For $E^c$ we conclude:

\[
\partial_g(E^c) = \partial_g(E) \cdot c E^c = \left( \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E})) \right) \cdot c E^c = \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E}) \cdot c \overline{E}^c)
\]

\[
= \bigcup_{\eta \in \eta^{-1}(g)} \eta(\partial_{\eta}(\overline{E^c})).
\]

$\square$

Now we lift the result of the last proposition to arbitrary partial derivatives w.r.t. arbitrary words.

**Theorem 17** Let $E$ be a regular expression over the ranked alphabet $\Sigma$ and $\overline{E}$ a linearization of $E$ w.r.t. $\eta$. Then we have for every $w \in \Sigma^*$

\[
\partial_w(E) = \bigcup_{\eta \in \eta^{-1}(w)} \eta(\partial_{\eta}(\overline{E})).
\]

**Proof.** We proceed by induction on the length of $w$ where the case $w = \varepsilon$ is trivial. By Proposition 16, the assumption holds for $|w| = 1$. Now consider $w = u g$ with $u \in \Sigma^*$ and $g \in \Sigma^*$. Using the induction hypothesis and Proposition 16 we get:

\[
\partial_{ug}(E) = \partial_g(\partial_u(E)) = \partial_g\left( \bigcup_{\eta \in \eta^{-1}(u)} \eta(\partial_{\eta}(\overline{E})) \right) = \partial_g \left( \eta \left( \bigcup_{\eta \in \eta^{-1}(u)} \partial_{\eta}(\overline{E}) \right) \right)
\]

\[
\text{(\star)}
\]
Consider the set $H = \bigcup \{ \partial_\eta(E) \mid \pi \in \eta^{-1}(u) \}$. Since it consists of finitely many regular expressions, there exists a set of linear regular expressions $H'$ over some ranked alphabet $\Sigma'$ and a function $\theta : \Sigma' \to \Sigma$ such that $\theta(H') = H$. In other words, $H'$ consists of linearizations of the regular expressions in $H$ w.r.t. $\theta$. Then the regular expressions from $H'$ are also linearizations of the regular expressions in $H = \eta(H)$ w.r.t. $\alpha = \eta \theta$. Hence we get from Proposition 16 and the above

$$
\partial_{ug}(E) = \partial_\eta(E) = \bigcup_{\eta' \in \alpha^{-1}(\eta)} \alpha(\partial_{\eta'}(H'))
= \bigcup_{\eta' \in \alpha^{-1}(\eta)} \eta\left( \bigcup_{\eta'' \in \theta^{-1}(\eta')} \partial_{\eta''}(H') \right)
= \bigcup_{\eta' \in \alpha^{-1}(\eta)} \eta(\partial_{\eta'}(E)) = \bigcup_{\eta' \in \alpha^{-1}(u)} \eta(\partial_{\eta'}(E)).
$$

Now we can identify the result of the quotient construction.

**Theorem 18** The finite tree automaton $\tilde{A}_E$ is isomorphic to $A_E$.

**Proof.** The state isomorphism is given by $\varphi : Q_E / \sim \to Q_E : \overline{G} \mapsto \eta(G)$. Firstly, $\varphi$ really maps into $Q_E$. Indeed, $\overline{G} = \partial_\eta(E)$ for some $\pi$. By Theorem 17, $\eta(G) \in \partial_{\Sigma_1}(E) = Q_E$. Injectivity of $\varphi$ is obvious by the definition of $\sim$. Surjectivity follows from Theorem 17. Certainly, $\varphi([\overline{E}]) = E$. Now, suppose $((\overline{F}, f, ([G_1], \ldots, [G_m])) \in \Delta_E$. Then there is a $\overline{f}$ such that $(\overline{F}, \overline{f}, ([G_1], \ldots, [G_m])) \in \Delta_\overline{E}$ which means $(G_1, \ldots, G_m) \in f^{-1}(F)$. But due to Proposition 16, $(G_1, \ldots, G_m) \in f^{-1}(F)$ where $G_i = \eta(G_i)$ and $F = \eta(F)$. Vice versa, if $(G_1, \ldots, G_m) \in f^{-1}(F)$, then there is an $\overline{f} \in \Sigma_{\geq 1}(E)$ with $(G_1, \ldots, G_m) \in f^{-1}(F)$. Moreover, we have for $c \in \Sigma_0$:

$$
([\overline{F}], c) \in \Delta_\overline{E} \iff c \in [\overline{F}] \iff c \in [\eta(\overline{F})] \iff (\eta(\overline{F}), c) \in \Delta_E.
$$

Now we can finally show that the FTA $A_E$ from Theorem 5 is finite. The number of transitions of $A_E$ is obviously bounded from above by $|Q_E| \cdot |\Sigma| \cdot |Q_E|^R \leq \|E\|^{R+1} \cdot |\Sigma|$ where $R$ is the maximal rank appearing in $\Sigma$. However, as we will show next, there is a much smaller bound.

**Theorem 19** Let $E$ be a regular expression. Then $A_E$ is a finite tree automaton with at most $\|E\|^2$ many states and $\|E\|^2$ transitions that accepts $[E]$.

**Proof.** The equality $L(A_E) = [E]$ was shown in Theorem 5. The numbers of states and transitions of $A_E$ equal those of $\tilde{A}_E$ by Theorem 18. Since $\tilde{A}_E$ is a quotient of $\overline{A}_E$, the result follows from the estimates in Corollary 15.

baeumeneu (version as of May 2, 2008)
5 Algorithmic issues

Due to Theorem 18, we can construct the FTA $\tilde{A}_E$ to get the automaton $A_E$. Following this line, Champarnaud and Ziadi [CZ01] gave in the case of words an algorithm with a $O(|E| \cdot |E|^2)$ space and time complexity. By algorithmic refinements they enhanced the algorithm to one with an $O(|E|^2)$ space and time complexity. We can mainly adapt this algorithm for the construction of the FT $A_E$ from a regular tree expression $E$. Since the algorithm is based on a more detailed analysis of the structure of partial derivatives, we first prove some more facts about them.

5.1 Form of partial derivatives

The sub-expression of $E$ at position $p$ is defined inductively as follows

- The sub-expression of $E$ at position $\varepsilon$ is the regular expression $E$.
- The sub-expression of $E$ at position $ap$ with $a \in \mathbb{N}$ and $p \in \mathbb{N}^*$ is defined iff one of the following holds
  - $E = f(E_1, \ldots, E_n)$ for some $n \geq a \geq 1$.
  - $E = E_1 + E_2$ or $E = E_1 \cdot c E_2$ and $a \in \{1, 2\}$.
  - $E = E_1^c$ and $a = 1$.

In all these cases, it equals the sub-expression of $E_a$ at position $p$.

**Theorem 20** Let $E$ be a regular expression and $D \in \partial_w(E)$ a partial derivative of $E$. Then there exist positions $p_1, \ldots, p_n$ in the syntax tree $t_E$ of $E$ and constant symbols $c_1, \ldots, c_n \in \Sigma_0$ such that

- $D = H_1 \cdot c_1 H_2 \ldots H_{n-1} \cdot c_{n-1} H_n$ where $H_i$ is the sub-expression of $E$ at position $p_i$,
- the number $n$ is bounded by the number of products $\cdot c$ and stars $^c$ appearing in $E$,
- if $p_i$ is a prefix of $p_j$, then $i \geq j$.

**Proof.** It follows immediately from Proposition 7, Corollary 9, and Theorem 17 that every partial derivative $D$ is a product of sub-expressions of $E$ and that $n$ is bounded by the number of products and stars in $E$. We turn to the last claim. We prove it by induction on the length of $w$. It is trivial for $w = \varepsilon$. Next let $w = f \in \Sigma_{\geq 1}$. We proceed by induction over the construction of $E$: If $E = g(E_1, \ldots, E_m)$, then $f = g$ since $\partial_f(E) \neq \emptyset$, and therefore $D = E_i$ for some $1 \leq i \leq m$. Choosing $n = 1$ and $p_1 = i$ proves the statement. If $E = F_1 + F_2$, then $D \in \partial_f(F_i)$ for some $1 \leq i \leq 2$. By the induction hypothesis, there are positions $p_j$ such that $D$ is the product of the sub-expressions of $F_i$ at these positions. Considering the positions $i,p_j$, $D$ is the product of sub-expressions of $E$ as required. Next
let $E = F_1 \cdot \cdot F_2$. If $D \in \partial f(F_2)$, we can argue as for the sum. Otherwise, there exists $F \in \partial f(F_1)$ such that $D = F \cdot c F_2$. As in case of addition, $F$ is a product of sub-expressions of $E$ at positions that all start with 1. Since $F_2$ is the sub-expression of $E$ at position 2, the claim follows. Finally let $E = F^{\times c}$ and therefore $D = G \cdot c E$ for some $G \in \partial f(F)$. As above, $G$ is the product of sub-expressions of $E$ at positions that all start with 1, and $E$ is the sub-expression of $E$ at position $\varepsilon$. This finishes the proof in case $|w| = 1$.

Now let $w \in \Sigma_{\geq 1}$ and $f \in \Sigma_{\geq 1}$ and consider $D \in \partial w f(E) = \partial f(\partial w (E))$. Then there is $D' \in \partial w (E)$ with $D \in \partial f(D')$. By the induction hypothesis, there are positions $p_1, \ldots, p_n$ such that $D = G_1 \cdot c_1 G_2 \ldots G_{n-1} \cdot c_{n-1} G_n$ where $G_i$ is the sub-expression of $E$ at position $p_i$ and, if $p_i$ is a prefix of $p_j$, then $i \geq j$. Since $D \in \partial f(D')$, there exists $1 \leq i \leq n$ and $F \in \partial f(G_i)$ such that $D = F \cdot c_i G_{i+1} \cdot c_{i+1} \ldots G_n$. By the base case considered above, $F$ is the product of sub-expressions of $G_i$ at positions $q_1, \ldots, q_m$ and therefore of sub-expressions of $E$ at the positions $p_i q_1, p_i q_2, \ldots, p_i q_m$.

\[ \square \]

5.2 An algorithm for computing the automaton

We only give a rough sketch of the algorithm. For details of the algorithm for word expressions we refer to [CZ01]. The adaptation to tree expressions does not cause much trouble.

Firstly, compute the syntax-tree $t_E$ of $E$. Note that the syntax-tree of a linearization $\overline{E}$ is obtained from $t_E$ by labelling each $g \in \Sigma_{\geq 1}$ additionally with its position. In the sequel, we will mainly work within the syntax-tree.

**Computing the states.** Recall that the partial derivatives of $E$ are projections of the partial derivatives of the linearization $\overline{E}$, cf. Theorem 17. Moreover, due to Proposition 8 the partial derivatives $\partial w g (\overline{E})$ depend just on the last symbol $\overline{g}$, i.e., the unique position in the syntax-tree labelled by $\overline{g}$. Now we calculate the partial derivatives of $\overline{E}$. Afterwards we identify partial derivatives that describe the same partial derivative of $E$.

1. **Computing the linearized partial derivatives.** For every position in the syntax-tree $t_E$ labelled by some $g \in \Sigma_{\geq 1}$ calculate the partial derivatives $\partial \cdot \overline{g} (\overline{E})$ by following the path from the respective position of $\overline{g}$ to the root of $t_E$. Hereby, we have to collect at every node on this path labeled by a product or a star the factors of the partial derivative, cf. Theorem 20. Moreover, if we pass a $\cdot c$-node from the right, then we have to check whether $c$ is in the semantics of the sub-expression to the left of this node. Note that for $g \in \Sigma_m$ we have up to $m$ partial derivatives. We do not just save the set $\partial \cdot \overline{g} (\overline{E})$ but also the respective tuple, i.e., the ordering of the partial derivatives which stems from the ordering of the sons of the $\overline{g}$-node.

2. **Identification of partial derivatives.** This step is done by a lexicographic ordering and an identification of consecutive partial derivatives.
Computing the transitions \((D, c)\) for \(c \in \Sigma_0\). We compute for every sub-expression \(F\) of \(E\) the set \([F] \cap \Sigma_0\) which can be easily done in the syntax-tree. Afterwards we calculate for every partial derivative \(D\) the set \([D] \cap \Sigma_0\) which can be done using the product structure of \(D\), cf. Theorem 20.

Computing the transitions for \(f \in \Sigma_{\geq 1}\). We can calculate for every linearized sub-expression \(\overline{F}\) the so-called FIRST-sets, i.e., those \(\overline{f} \in \Sigma_{\geq 1}\) such that \(\overline{f}\) has an unguarded occurrence in \(\overline{F}\). Now we can compute from those sets the respective FIRST-sets for every linearized partial derivative \(\overline{D}\). But for every linearized symbol \(\overline{f}\) in the FIRST-set we obtain the unique \(\overline{f}\)-transition from \(\overline{D}\) by the unique tuple of the linearized partial derivatives from \(\partial_{\overline{f}}(\overline{E})\). Note that this tuple was obtained already in the computation of states. A projection gives the transitions for \(f\).

Complexity. The algorithm for word expressions has an \(O(||E|| \cdot |E|^2)\) space and time complexity, cf. [CZ01]. The algorithm for regular tree expressions as sketched above follows exactly the same lines but has to keep track of tuples of partial derivatives instead of singletons as it is the case for words. Hence, the algorithm has an \(O(R \cdot ||E|| \cdot |E|^2)\) space and time complexity where \(R \geq 1\) is the maximal rank appearing in the ranked alphabet \(\Sigma\) (and at least 1).

The improvements suggested by Champarnaud and Ziadi, mainly a preprocessing of star sub-expressions of \(E\) and an improved computation of the FIRST-sets in the syntax-tree, carry over to the above algorithm. Hence, we shall get for such an improved algorithm an \(O(R \cdot |E|^2)\) space and time complexity.

References


REFERENCES


