Reordering Derivatives of Trace Closures of Regular Languages

Hendrik Maarand
Department of Software Science, Tallinn University of Technology, Estonia
hendrik@cs.ioc.ee

Tarmo Uustalu
School of Computer Science, Reykjavik University, Iceland
Department of Software Science, Tallinn University of Technology, Estonia
tarmo@ru.is

Abstract
We provide syntactic derivative-like operations, defined by recursion on regular expressions, in the styles of both Brzozowski and Antimirov, for trace closures of regular languages. Just as the Brzozowski and Antimirov derivative operations for regular languages, these syntactic reordering derivative operations yield deterministic and nondeterministic automata respectively. But trace closures of regular languages are in general not regular, hence these automata cannot generally be finite. Still, as we show, for star-connected expressions, the Antimirov and Brzozowski automata, suitably quotiented, are finite. We also define a refined version of the Antimirov reordering derivative operation where parts-of-derivatives (states of the automaton) are nonempty lists of regular expressions rather than single regular expressions. We define the uniform scattering rank of a language and show that, for a regexp whose language has finite uniform scattering rank, the truncation of the (generally infinite) refined Antimirov automaton, obtained by removing long states, is finite without any quotienting, but still accepts the trace closure. We also show that star-connected languages have finite uniform scattering rank.

2012 ACM Subject Classification Theory of computation → Regular languages; Theory of computation → Concurrency

Keywords and phrases Mazurkiewicz traces, trace closure, regular languages, finite automata, language derivatives, scattering rank, star-connected expressions

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.40

Related Version A full version of this article is available as a preprint [12], https://arxiv.org/abs/1908.03551.

Funding Both authors: supported by the Estonian Ministry of Education and Research institutional grant no. IUT33-13. Hendrik Maarand: supported by the ERDF funded Estonian national CoE project EXCITE (2014-2020.4.01.15-0018).

Acknowledgements We thank Pierre-Louis Curien, Jacques Sakarovitch, Simon Doherty, Georg Struth and Ralf Hinze for inspiring discussions, and our anonymous reviewers for the exceptionally thorough and constructive feedback they gave us.

1 Introduction

Traces were introduced to concurrency theory by Mazurkiewicz [13, 14] as an alternative to words. A word can be seen as a linear order that is labelled with letters of the alphabet. Intuitively, the main idea of traces is that the linear order, corresponding to sequentiality, is replaced with a partial order. Sets of words (or word languages) can be used to describe the behaviour of concurrent systems. Similarly, sets of traces (or trace languages) can also be used for this purpose. The difference is that descriptions in terms of traces do not distinguish
between different linear extensions (words) of the same partial order (trace) – they are
considered equivalent. Different linear extensions of the same partial order can be seen as
different observations of the same behaviour.

Given a word language \( L \) and a letter \( a \), the derivative of \( L \) along \( a \) is the language
consisting of all the words \( v \) such that \( av \) belongs to \( L \). An essential difference between words
and traces is that a nonempty word (a linear order) has its first letter as the unique minimal
element, but a nonempty trace (a partial order) may have several minimal elements. A trace
from a trace language can be derived along any of its minimal letters. Clearly, a minimal
letter of a trace need not be the first letter of a word representing this trace.

It is well-known that the derivative of a regular word language along a letter is again regular. Brzozowski [5] showed that a regexp for it can be computed from a regexp for the
given language, and Antimirov [2] then further optimized this result. We show that these
syntactic derivative operations generalize to trace closures (i.e., closures under equivalence)
of regular word languages in the form of syntactic reordering derivative operations.

The syntactic derivative operations for regular word languages provide ways to construct
automata from a regexp. The Brzozowski derivative operation is a function on regexps while
the Antimirov derivative operation is a relation. Accordingly, they yield deterministic and
nondeterministic automata. The set of Brzozowski derivatives of a regexp (modulo
appropriate equations) and the set of Antimirov parts-of-derivatives are finite, hence so are
the resulting automata. Our generalizations to trace closures of regular languages similarly
give deterministic and nondeterministic automata, but these cannot be finite in general.
Still, as we show, for a star-connected expression, the Antimirov and Brzozowski automata,
suitably quoted, are finite. We also develop a finer version of the Antimirov reordering
derivative, where parts-of-derivatives are nonempty lists of regexps rather than single regexps,
and we show that the set of expressions that can appear in these lists for a given initial
regexp is finite. We introduce a new notion of uniform scattering rank (a variant of Hashiguchi’s scattering rank [7]) and show that, for a regexp whose language has
finite uniform rank, a truncation of the refined reordering Antimirov automaton accepts its
trace closure despite the removed states, and is finite, without any quotienting.

A full version of this article, with proofs and background material on classical language
derivatives and trace closures of regular languages is available as a preprint [12].

2 Preliminaries on Word Languages

An alphabet \( \Sigma \) is a finite set (of letters). A word over \( \Sigma \) is a finite sequence of letters. The
set \( \Sigma^* \) of all words over \( \Sigma \) is the free monoid on \( \Sigma \) with the empty word \( \varepsilon \) as the unit
and concatenation of words (denoted by \( \cdot \) that can be omitted) as the multiplication. We write
\( |u| \) for the length of a word \( u \) and also \( |X| \) for the size of a subalphabet \( X \). By \( |u|_a \) we mean
the number of occurrences of \( a \) in \( u \). By \( \Sigma(u) \) we denote the set of letters that appear in \( u \).

A (word) language is a subset of \( \Sigma^* \). The empty word and concatenation of words lift to
word languages via \( 1 =_{dt} \{ \varepsilon \} \) and \( L \cdot L' =_{dt} \{ uv \mid u \in L \land v \in L' \} \).

2.1 Derivatives of a Language

A word language \( L \) is said to be nullable \( L \downarrow_L \), if \( \varepsilon \in L \). The derivative (or left quotient)\(^1 \) of \( L \)
along a word \( u \) is defined by \( D_u L =_{dt} \{ v \mid uv \in L \} \). For any \( L \), we have \( D_{\varepsilon} L = L \) as well
as \( D_{uv} L = D_u(D_v L) \) for any \( u, v \in \Sigma^* \), i.e., the operation \( D : \mathcal{P}\Sigma^* \times \Sigma^* \to \mathcal{P}\Sigma^* \)
is a right action of \( \Sigma^* \) on \( \mathcal{P}\Sigma^* \). We also have \( L = \{ \varepsilon \mid L \downarrow_L \} \cup \bigcup \{ \{a\} : D_u L \mid a \in \Sigma \} \), and for any \( u \in \Sigma^* \),
we have \( u \in L \) iff \( (D_u L) \downarrow_L \).

\(^1 \) We use the word “derivative” both for languages and expressions, reserving the word “quotient” for
quotients of sets by equivalence relations.
The set $RE$ of regular expressions (in short, regexps) over $\Sigma$ is given by the grammar
$E, F ::= a \mid 0 \mid E + F \mid 1 \mid EF \mid E^*$ where $a$ ranges over $\Sigma$.

The word-language semantics of regular expressions is given by a function $[\_] : RE \to \mathcal{P} \Sigma^*$
defined recursively by

\[
\begin{align*}
[a] &=_{df} \{a\} \\
[0] &=_{df} \emptyset \\
[1] &=_{df} 1 \\
[EF] &=_{df} [E] \cdot [F] \\
[E + F] &=_{df} [E] \cup [F] \\
[E^*] &=_{df} \mu_X. 1 \cup [E] \cdot X
\end{align*}
\]

A word language $L$ is said to be regular (or rational) if $L = [E]$ for some regexp $E$.
Kleene algebras are defined by an equational theory. It was shown by Kozen [11] that the
set $\{[E] \mid E \in RE\}$ of all regular languages together with the language operations $\emptyset, \cup, 1,
\cdot, (\_)^*$ is the free Kleene algebra on $\Sigma$. An important property for us is that $E = F$ iff
$\llbracket E \rrbracket = \llbracket F \rrbracket$ where $\llbracket \cdot \rrbracket$ refers to valid equations in the Kleene algebra theory.

Kleene’s theorem [9] says that a word language is rational if it is
recognizable, i.e., accepted
by a finite deterministic automaton (acceptance by a finite nondeterministic automaton is
an equivalent condition because of determinizability [17]).

2.3 Brzozowski and Antimirov Derivatives

Derivatives of regular languages are regular. A remarkable fact is that they can be computed
syntactically, on the level of regular expressions. There are two constructions for this, due to
Brzozowski [5] and Antimirov [2]. The Brzozowski and Antimirov derivative operations yield
deterministic resp. nondeterministic automata accepting the language of a regular expression $E$.
The Antimirov automaton is finite. The Brzozowski automaton becomes finite when
quotiented by associativity, commutativity and idempotence for $\cdot$. Identified up to the
Kleene algebra theory, the states of the Brzozowski automaton correspond to the derivatives
of the language $\llbracket E \rrbracket$. Regular languages can be characterized as languages with finitely
many derivatives.

3 Trace Closures of Regular Languages

3.1 Trace Closure of a Word Language

An independence alphabet is an alphabet $\Sigma$ together with an irreflexive and symmetric relation
$I \subseteq \Sigma \times \Sigma$ called the independence relation. The complement $D$ of $I$, which is reflexive and
symmetric, is called dependence. We extend independence to words by saying that two words
$u$ and $v$ are independent, $uIv$, if $aIb$ for all $a, b$ such that $a \in \Sigma(u)$ and $b \in \Sigma(v)$.

Let $\sim^I \subseteq \Sigma^* \times \Sigma^*$ be the least congruence relation on the free monoid $\Sigma^*$ such that
$aIb$ implies $ab \sim^I ba$ for all $a, b \in \Sigma$. If $uIv$, then $uv \sim^I vu$.

A (Mazurkiewicz) trace is an equivalence class of words wrt. $\sim^I$. The equivalence class
of a word $w$ is denoted by $[w]^I$.

A word $a_1 \ldots a_n$ where $a_i \in \Sigma$ yields a directed node-labelled acyclic graph as follows.
Take the vertex set to be $V =_{df} \{1, \ldots, n\}$ and label vertex $i$ with $a_i$. Take the edge set to be
$E =_{df} \{(i, j) \mid i < j \wedge a_iDa_j\}$. This graph $(V, E)$ for a word $w$ is called the dependence
graph of $w$ and is denoted by $\langle w \rangle_D$. If $w \sim^I z$, then the dependence graphs of $w$ and $z$ are
isomorphic, i.e., traces can be identified with dependence graphs up to isomorphism.

The set $\Sigma^*/\sim^I$ of all traces is the free partially commutative monoid on $(\Sigma, I)$. If $I = \emptyset$,
then $\Sigma^*/\sim^I \cong \Sigma^*$, the set of words, i.e., we recover the free monoid. If $I = \{(a, b) \mid a \neq b\}$,
then $\Sigma^*/\sim^I \cong M_I(\Sigma)$, the set of finite multisets over $\Sigma$, i.e., the free commutative monoid.
A trace language is a subset of $\Sigma^*/\sim^I$. Trace languages are in bijection with word languages that are (trace) closed in the sense that, if $z \in L$ and $w \sim^I z$, then also $w \in L$. If $T$ is a trace language, then its flattening $L = \text{def} \bigcup T$ is a closed word language. On the other hand, the trace language corresponding to a closed word language $L$ is $T = \text{def} \{ t \in \Sigma^*/\sim^I \mid \exists z \in t, z \in L \} = \{ t \in \Sigma^*/\sim^I \mid \forall z \in t, z \in L \}$.

Given a general (not necessarily closed) word language $L$, we define its (trace) closure $[L]^I$ as the least closed word language that contains $L$. Clearly $[L]^I = \{ w \in \Sigma^* \mid \exists z \in L. w \sim^I z \}$ and also $[L]^I = \bigcup \{ t \in \Sigma^*/\sim^I \mid \exists z \in t, z \in L \}$. For any $L$, we have $[[L]^I]^I = [L]^I$, so $[\underline{L}]^I$ is a closure operator. Note also that $L$ is closed iff $[L]^I = L$.

As seen in Section 2.1, the derivative of a word language is the set of all suffixes for a prefix. We now look at what the prefixes and suffixes of a word as a representative of a trace should be. For a word $uvw'$ such that $vfu$, we can consider $u$ to be its prefix, up to reordering, and $vv'$ to be the suffix. This is because an equivalent word $uwv'$ strictly has $u$ as a prefix and $vv'$ as the suffix. Similarly, we may also want to consider $u'$ to be a prefix of $uvw'$ when $u' \sim^I u$ since $u'uvv' \sim^I uvw' \sim^I uvw'$. Note that if $a$ is such a prefix of $z$, then, by irreflexivity of $I$, this $a$ is the first of $z$. In general, when $u$ is a prefix of $z$, then the letter occurrences in $u$ uniquely map to letter occurrences in $z$. We scale these ideas to allow $u$ to be scattered in $z = v_0v_1...v_nv_n$ in either the sense that $u = u_1...u_n$ or $u \sim^I u_1...u_n$. We also define bounded versions of scattering that become relevant in Section 5.

**Definition 3.** For all $u_1, u_2, ..., u_n \in \Sigma^*, v_0 \in \Sigma^*, v_1, ..., v_{n-1} \in \Sigma^*, v_n \in \Sigma^*$, $u_1, ..., u_n \triangleleft z \triangleright v_0, ..., v_n = \text{def} \exists n \in \mathbb{N}, u_1, ..., u_n, v_0, ..., v_n. u = u_1 ... u_n \land v = v_0 ... v_n \land u_1, ..., u_n \triangleleft z \triangleright v_0, ..., v_n$.

**Definition 4.** For all $u, v, z \in \Sigma^*$ and $N \in \mathbb{N}$, $u <_N z \triangleright v = \text{def} \exists n \leq N, u_1, ..., u_n, v_0, ..., v_n. u = u_1 ... u_n \land v = v_0 ... v_n \land u_1, ..., u_n \triangleleft z \triangleright v_0, ..., v_n$.

**Example 5.** Let $\Sigma = \text{def} \{ a, b, c \}$ and $aIb$ and $aIc$. Take $z = \text{def} aabcba$. We have $a < z \triangleright ca$ since $a, b < z \triangleright e, a, c$. We can visualize this by underlining the subwords of $u = \text{def} a$ in $z = aabcba$. This scattering is valid because $\varepsilon Ia$, $\varepsilon Ib$ and $aIb$: recall that Def. 1 requires all underlined subwords $u_i$ to be independent with all non-underlined subwords $v_i$ to their left in $z$. Similarly we have $aa, a < z \triangleright \varepsilon, bac, \varepsilon$ because $z = aabcba\varepsilon, \varepsilon Iaa, \varepsilon Ia$ and $bIc$. Note that neither $aabcba\varepsilon$ nor $aabcba\varepsilon$ satisfies the conditions about independence and thus there is no such that $ba < z \triangleright v$. We do have $ba < z \triangleright ca$ though, since $ba \sim^I ab$ and $a, b < z \triangleright e, a, c$.bacha.

**Proposition 6.** For all $u, v, z \in \Sigma^*$, $w \sim^I z$ $\iff$ $u \sim^I z \triangleright v$.

### 3.2 Trace-Closing Semantics of Regular Expressions

We now define a nonstandard word-language semantics of regeps that directly interprets $E$ as the trace closure $[[E]]^I$ of its standard regular word-language denotation $[E]$.
We have $\{a\}^I = \{a\}, \{\emptyset\}^I = \emptyset, [L \cup L']^I = [L]^I \cup [L']^I$ and $[1]^I = 1$. But for general $I$, we do not have $[L \cdot L']^I = [L]^I \cdot [L']^I$. For example, for $\Sigma = \{a, b\}$ and $ab$, we have $\{a\}^I = \{a\}, \{(b)\}^I = \{(b)\}$ whereas $\{ab\}^I = \{(ab, ba) \neq \{ab\} = \{\{a\}\}^I \cdot \{(b)\}^I$. Hence we need a different concatenation operation.

**Definition 7.**
1. The $I$-reordering concatenation of words $\cdot^I : \Sigma^* \times \Sigma^* \rightarrow P\Sigma^*$ is defined by

$$
\varepsilon \cdot^I v = \{v\},
\quad u \cdot^I \varepsilon = \{u\},
\quad au \cdot^I bv = \{a\} \cdot (u \cdot^I bv) \cup \{b | auIb\} \cdot (au \cdot^I v)
$$

2. The lifting of $I$-reordering concatenation to languages is defined by

$$
L \cdot^I L' = \bigcup \{u \cdot^I v | u \in L \land v \in L'\}
$$

Note that $\{b | auIb\}$ acts as a test: it is either $\emptyset$ or $\{b\}$.

**Example 8.** Let $\Sigma = \{a, b\}$ and $ab$. Then $a \cdot^I b = \{ab, ba\}$, $aa \cdot^I b = \{aab, aba, baa\}$, $a \cdot^I bb = \{abb, bab, bba\}$ and $ab \cdot^I ba = \{aba\}$. The last example shows that although $I$-reordering concatenation is defined quite similarly to shuffle, it is different.

**Proposition 9.** For any $u, v, z \in \Sigma^*$, $z \in u \cdot^I v \iff u \triangleleft z \triangleright v$.

**Proposition 10.** For any languages $L$ and $L'$, $[L \cdot L']^I = [L]^I \cdot [L']^I$.

Evidently, if $I = \emptyset$, then reordering concatenation is just ordinary concatenation: $u \cdot^I v = \{uv\}$. For $I = \Sigma \times \Sigma$, which is forbidden in independence alphabets, as $I$ is required to be irreflexive, it is shuffle: $u \cdot^I v = u \triangleright v$. For general $I$, it has properties similar to concatenation. In particular, we have

$$
1 \cdot^I L = L, \quad \emptyset \cdot^I L = \emptyset
$$

$$
L \cdot^I 1 = L, \quad (L_1 \cup L_2) \cdot^I L = L_1 \cdot^I L \cup L_2 \cdot^I L
$$

$$
(L \cdot^I L') \cdot^I L'' = L \cdot^I (L' \cdot^I L'') \quad (L_1 \cup L_2) \cdot^I (L_3 \cup L_4) \subseteq (L_1 \cdot^I L_3) \cup (L_2 \cdot^I L_4)
$$

but also other equations of the concurrent Kleene algebra theory introduced in [8].

We are ready to introduce the closing semantics of regular expressions.

**Definition 11.** The trace-closing semantics $\llbracket \cdot \rrbracket^I : \text{RE} \rightarrow P\Sigma^*$ of regular expressions is defined recursively by

$$
\llbracket a \rrbracket^I = \{a\}, \quad \llbracket \emptyset \rrbracket^I = \emptyset, \quad \llbracket [E + F] \rrbracket^I = \llbracket E \rrbracket^I \cup \llbracket F \rrbracket^I, \quad \llbracket [E \cdot F] \rrbracket^I = \llbracket E \rrbracket^I \cdot \llbracket F \rrbracket^I, \quad \llbracket [E^*] \rrbracket^I = \nu X. 1 \cup \llbracket E \rrbracket^I \cdot X
$$

Compared to the standard semantics of regular expressions, the difference is in the handling of the $EF$ case (and consequently also the $E^*$ case) due to the cross-commutation that happens in concatenation of traces and must be accounted for by $\cdot^I$.

With $I = \emptyset$, we fall back to the standard interpretation of regular expressions: $\llbracket E \rrbracket^\emptyset = \llbracket E \rrbracket$. For $I$ a general independence relation, we obtain the desired property that the semantics delivers the trace closure of the language of the regexp.

**Proposition 12.** For any $E$, $\llbracket E \rrbracket^I$ is trace closed; moreover, $\llbracket E \rrbracket^I = \llbracket [E] \rrbracket^I$. 

---

**H. Maarand and T. Uustalu** 40:5

**CONCUR 2019**
40:6 Reordering Derivatives of Trace Closures of Regular Languages

3.3 Properties of Trace Closures of Regular Languages

Trace closures of regular languages are theoretically interesting, as they have intricate properties. For a thorough survey, see Ochmański’s handbook chapter [16].

The most important property for us is that the trace closure of a regular language is not necessarily regular. Consider \( \Sigma = \{a, b\} \), \( aIb \). Let \( L = \{(ab)^*\} \). The language \([L]^I = \{u \mid |u|_a = |u|_b\}\) is not regular.

The class of trace closures of regular languages over an independence alphabet behaves quite differently from the class of regular languages over an alphabet. For example, the class of trace closures of regular languages over \((\Sigma, I)\) is closed under complement iff \(I\) is quasi-transitive (i.e., its reflexive closure is transitive) [3, 1, 18] (cf. [16, Thm. 6.2.5]); the question of whether the trace closure of the language of a regexp over \((\Sigma, I)\) is regular is decidable iff \(I\) is quasi-transitive [19] (cf. [16, Thm. 6.2.7]).

A closed language is regular iff the corresponding trace language is accepted by a finite asynchronous (a.k.a. Zielonka) automaton [21, 22]. In Section 4.4, we will see further characterizations of regular closed languages based on star-connected expressions.

4 Reordering Derivatives

We are now ready to generalize the Brzozowski and Antimirov constructions for trace closures of regular languages. To this end, we switch to what we call reordering derivatives.

4.1 Reordering Derivative of a Language

Let \((\Sigma, I)\) be a fixed independence alphabet. We generalize the concepts of (semantic) nullability and derivative of a language to concepts of reorderable part and reordering derivative.

▶ Definition 13. We define the \(I\)-reorderable part of a language \(L\) wrt. a word \(u\) by \(R^I_uL = \{v \mid vIu\}\) and the \(I\)-reordering derivative along \(u\) by \(D^I_uL = \{v \mid \exists z \in L. u \sim_I z \triangleright v\}\). By Prop. 9, we can equivalently say that \(D^I_uL = \{v \mid \exists z \in L. z \in [a]^{I.I} v\}\). For a single-letter word \(a\), we get \(D^I_aL = \{v_Iv_r \mid v_Iav_r \in L \land v_Ia \} = \{v \mid \exists z \in L. z \in a.I v\}\). That is, we require some reordering of \(u\) (resp. \(a\)) to be a prefix, up to reordering, of some word \(z\) in \(L\) with \(v\) as the corresponding strict suffix. (In other words, for the sake of precision and emphasis, we allow reordering of letters within \(u\) and across \(u\) and \(v\), but not within \(v\).)

▶ Example 14. Let \(\Sigma = \{a, b, c\}\) and \(aIb\). Take \(L = \{(\varepsilon, a, b, ca, aa, bbb, babca, abba)\). We have \(R^I_aL = R^I_{aa}L = \{\varepsilon, b, bbb\}\), \(D^I_aL = \{\varepsilon, a, babca, abba\}\) and \(D^I_{aa}L = \{\varepsilon, bbb\}\).

In the special case \(I = \emptyset\), we have \(R^I_uL = L\), \(R^I_uL = \{e \mid L \downarrow\} \) for any \(u \neq \varepsilon\), and \(D^I_uL = D_uI L\). In the general case, the reorderable part and reordering derivative enjoy the following properties.

▶ Lemma 15. For every \(L\).
1. \(R^I_L = L\); for every \(u, v \in \Sigma^*\), \(R^I_{(u)_{\varepsilon}}L = R^I_{(u)_{\varepsilon}}L\);
2. for every \(u, u' \in \Sigma^*\), \(R^I_{(\Sigma_{u})_{\varepsilon}}L = R^I_{(\Sigma_{u'})_{\varepsilon}}L\).

We extend \(R^I\) to subsets of \(\Sigma\): by \(R^I_uL\), we mean \(R^I_uL\) where \(u\) is any enumeration of \(X\).

▶ Lemma 16. For every \(L\).
1. \(D^I_L = L\); for any \(u, v \in \Sigma^*\), \(D^I_{(L)_{u}}L = D^I_{(L)_{u}}L\);
2. for any \(u, u' \in \Sigma^*\) such that \(u \sim^I u'\), we have \(D^I_uL = D^I_{u'}L\).
Recursively by regexp are given by functions $W$. We now show that they can be computed syntactically, generalizing the classical syntactic nullability and Brzozowski derivative operations [5].

**Proposition 17.** For every $L$, 
1. for any $u \in \Sigma^*$, $W_u([L]^I) = [W_uL]^I$;
   if $L$ is closed (i.e., $[L]^I = L$), then, for any $u \in \Sigma^*$, $W_uL$ is closed and $W_uL = W_uL$; 
2. for any $u, v \in \Sigma^*$, $uv \in [L]^I$ iff $v \in [W_uL]^I$; 
3. for any $u \in \Sigma^*$, $u \in [L]^I$ iff $(W_uL)_\downarrow$; 
4. $[L]^I = \{ \varepsilon \mid L \downarrow \} \cup \bigcup_{a \in \Sigma} \{a\} \cdot [W_aL]^I$.

**Example 18.** Let $\Sigma = \{a, b\}$ and $ab$. Take $L$ to be the regular language $[(ab)^*]$. We have already noted that the language $[L]^I = \{u \mid |u|_a = |u|_b\}$ is not regular. For any $n \in \mathbb{N}$, $[W_aL] = \{a^n\} \cdot L = [a^n(ab)^*]$ whereas $D_{b^*}([L]^I) = \{a^n\} \cdot [L]^I = \{u \mid |u|_a = |u|_b + n\}$.

We can see that $[L]^I$ has infinitely many derivatives, none of which are regular, and $L$ has infinitely many reordering derivatives, all regular.

### 4.2 Brzozowski Reordering Derivative

The reorderable parts and reordering derivatives of regular languages turn out to be regular. We now show that they can be computed syntactically, generalizing the classical syntactic nullability and Brzozowski derivative operations [5].

**Definition 19.** The $I$-reorderable part and the Brzozowski $I$-reordering derivative of a regexp are given by functions $R^I, D^I : \mathbb{R} \times \Sigma \rightarrow \mathbb{R}$ and $R^I, D^I : \mathbb{R} \times \Sigma^+ \rightarrow \mathbb{R}$ defined recursively by

$$
R^I_n = \begin{cases} 
1 & \text{if } n \text{ is a letter} \\
0 & \text{otherwise}
\end{cases}
\quad \quad 
D^I_0 = \begin{cases} 
1 & \text{if } a = b \text{ then 1 else 0} \\
0 & \text{otherwise} 
\end{cases}
$$

$$
D^I_n(E + F) = D^I_n(E) + D^I_n(F) 
\quad \quad 
D^I_n(E^\ast) = (D^I_nE)^\ast
$$

$$
D^I_n(a) = a 
\quad \quad 
D^I_n(E) = D^I_n(aE)
$$

The regexp $R_aE$ is nothing but $E$ with all occurrences of letters dependent with $u$ replaced with 0. The definition of $D$ is more interesting. Compared to the classical Brzozowski derivative, the nullability condition $E \downarrow$ in the $EF$ case has been replaced with concatenation with the reorderable part $R_aE$, and the $E^\ast$ case has also been adjusted.

The functions $R$ and $D$ on regexps compute their semantic counterparts on the corresponding regular languages.

**Proposition 20.** For any $E$,
1. for any $a \in \Sigma$, $R^I_a[E] = [R^I_aE]$ and $D^I_a[E] = [D^I_aE]$;
2. for any $u \in \Sigma^*$, $R^I_u[E] = [R^I_uE]$ and $D^I_u[E] = [D^I_uE]$.

**Proposition 21.** For any $E$,
1. for any $a \in \Sigma$, $v \in \Sigma^*$, $av \in [E]^I$ $\iff$ $v \in [D^I_aE]^I$;
2. for any $u, v \in \Sigma^*$, $uv \in [E]^I$ $\iff$ $v \in [D^I_uE]^I$;
3. for any $u \in \Sigma^*$, $u \in [E]^I$ $\iff$ $(D^I_uE)_\downarrow$. 

CONCUR 2019
Example 22. Let \( \Sigma = \{a, b\} \), \( aIb \) and \( E = \{aa + ab + b\} \).

\[
D_a^1E = D_b^1aa + D_b^1ab + D_b^1b \\
= ((D_b^1a)a + (R_b^1a)(D_b^1a)) + ((D_b^1a)b + (R_b^1a)(D_b^1b)) + D_b^1b \\
\approx (0a + a0) + (0b + a1) + 1 \equiv a + 1
\]

\[
D_a^1(E^*) = (R_b^1E)^* D_b^1E E^* \\
= (aa + a0 + 0)^*(((a0 + a0) + (0b + a1) + 1)\equiv (aa)^*(a + 1)E^*
\]

\[
D_a^1(E^*) \equiv D_b^1((aa)^*(a + 1)E^*) \equiv (aa)^*(a + 1)(aa)^*(a + 1)E^*
\]

As with the classical Brzozowski derivative, we can use the reordering Brzozowski derivative to construct deterministic automata. For a regexp \( E \), take \( Q^E = \{ q_u^E | u \in \Sigma^* \} \), \( q_0^E = \{ E' \in Q^E | E' \downarrow \} \), \( \delta_a^E E' = \{ D_a^1E' \} \) for \( E' \in Q^E \). By Prop. 21, this automaton accepts the closure \([E]^I\). But even quotiented by the full Kleene algebra theory, the quotient of \( Q^E \) is not necessarily finite, i.e., we may be able to construct infinitely many different languages by taking reordering derivatives. For the regexp from Example 18, we have \( D_a^1((ab)^*) \equiv a^\omega(ab)^* \), so it has infinitely many Brzozowski reordering derivatives even up to the Kleene algebra theory. This is only to be expected, as the closure \([((ab)^*)]^I\) is not regular and cannot possibly have an accepting finite automaton.

### 4.3 Antimirov Reordering Derivative

Like the classical Brzozowski derivative that was optimized by Antimirov [2], the Brzozowski reordering derivative construction can be optimized by switching from functions on regexps to multivalued functions or relations.

#### Definition 23.

The Antimirov \( I \) reordering parts-of-derivatives of a regexp along a letter and a word are relations \( \rightarrow^I \subseteq RE \times \Sigma \times RE \) and \( \rightarrow^I^* \subseteq RE \times \Sigma^* \times RE \) defined inductively by

\[
\begin{array}{llll}
E \rightarrow^I (a, E') & F \rightarrow^I (a, F') \\
E + F \rightarrow^I (a, E') & E + F \rightarrow^I (a, F') \\
E \rightarrow^I (a, E') & F \rightarrow^I (a, F') & E \rightarrow^I (a, E') \\
EF \rightarrow^I (a, ER) & EF \rightarrow^I (a, (R_a^E)F') & E^* \rightarrow^I (a, (R_a^E)^*E') \\
E \rightarrow^I^* (u, E') & E' \rightarrow^I (a, E'') \\
E \rightarrow^I^* (\varepsilon, E) & E \rightarrow^I^* (ua, E'')
\end{array}
\]

Here \( R^I \) is defined as before. Similarly to the Brzozowski reordering derivative from the previous subsection, the condition \( E^I \downarrow \) in the second \( EF \) rule has been replaced with concatenation with \( R_a^E \), and the \( E^* \) rule has been adjusted.

Collectively, the Antimirov reordering parts-of-derivatives of a regexp compute the semantic reordering derivative of the language \([E]^I\).

#### Proposition 24.

For any \( E \),
1. for any \( a \in \Sigma \), \( D_a^1[E]^I = \cup \{(E') \mid E \rightarrow^I (a, E')\} \);
2. for any \( u \in \Sigma^* \), \( D_u^1[E]^I = \cup \{(E') \mid E \rightarrow^I^* (u, E')\} \).

#### Proposition 25.

For any \( E \),
1. for any \( a \in \Sigma \), \( v \in \Sigma^*, av \in [E]^I \iff \exists E'. E \rightarrow^I (a, E') \land v \in [E']^I \);
2. for any \( u, v \in \Sigma^* \), \( uv \in [E]^I \iff \exists E'. E \rightarrow^I^* (u, E') \land v \in [E']^I \);
3. for any \( u \in \Sigma^* \), \( u \in [E]^I \iff \exists E'. E \rightarrow^I^* (u, E') \land E' \downarrow \).
Example 26. Let us revisit Example 22. The Antimirov reordering parts-of-derivatives of $E$ along $b$ are $a1$ and 1:

\[
\begin{align*}
W &\rightarrow (b, 1) \\
ab &\rightarrow (b, a1) \\
ab + b &\rightarrow (b, a1) \\
ab + b &\rightarrow (b, 1) \\
\end{align*}
\]

The Antimirov reordering parts-of-derivatives of $E^*$ along $b$ are therefore $E_b^*(a1)E^*$ and $E_b^*1E^*$ where $E_b = \mathcal{R}_1E = aa+a0+0$. Recall that, for the Brzozowski reordering derivative, we computed $D_b^1E = (0a + a0) + (0b + a1) + 1$ and $D_b^1E^* = E_b^*((0a + a0) + (0b + a1) + 1)E^*$.

Like the classical Antimirov construction, the Antimirov reordering parts-of-derivatives of a regexp $E$ give a nondeterministic automaton by $Q_E = \{E' \mid \exists u \in \Sigma^*. E \rightarrow^*_u (a, E')\}$, $I_E = \{E\}$, $F_E = \{E' \in Q_E \mid E' \downarrow\}$. $E' = (a, E'') \rightarrow^*_u E''$ for $E', E'' \in Q_E$. This automaton accepts $[E]^f$ by Prop. 25, but is generally infinite, also if quotiented by the full Kleene algebra theory. Revisiting Example 18 again, $(ab)^*$ must have infinitely many Antimirov reordering parts-of-derivatives modulo the Kleene algebra theory since $[(ab)^*]^f$ is not regular and cannot have a finite accepting nondeterministic automaton. Specifically, it has $(a0)^*((a1)\ldots((a0)^*((a1)(ab)^*))\ldots) \doteq a^n(ab)^*$ as its single reordering part-of-derivative along $b^n$.

However, if quotienting the Antimirov automaton for $E$ by some sound theory (a theory weaker than the Kleene algebra theory) makes it finite, then the Brzozowski automaton can also be quotiented to become finite.

Proposition 27. For any $E$,
1. for any $a \in \Sigma$, $D_a^tE \doteq \Sigma\{E' \mid E \rightarrow^t (a, E')\}$;
2. for any $u \in \Sigma^*$, $D_u^tE \doteq \Sigma\{E' \mid E \rightarrow^t (u, E')\}$

(using the semilattice equations for $0+, +, 0$ is zero, and distributivity of $\cdot$ over $+$).

Corollary 28. If some quotient of the Antimirov automaton for $E$ (accepting $[E]^f$) is finite, then also some quotient of the Brzozowski automaton is finite.

4.4 Star-Connected Expressions

Star-connected expressions are important as they characterize regular closed languages. A corollary of that is a further characterization of such languages in terms of a “concurrent” semantics of regexps that interprets Kleene star nonstandardly as “concurrent star”.

Definition 29. A word $w \in \Sigma^*$ is connected if its dependence graph $\langle w \rangle_D$ is connected. A language $L \subseteq \Sigma^*$ is connected if every word $w \in L$ is connected.

Definition 30.
1. Star-connected expressions are a subset of the set of all regexps defined inductively by: 0, $a \in \Sigma$ are star-connected. If $E$ and $F$ are star-connected, then so are $E + F$ and $EF$. If $E$ is star-connected and $[E]$ is connected, then $E^*$ is star-connected.
2. A language $L$ is said to be star-connected if $L = [E]$ for some star-connected regexp.

Ochmański [15] proved that a closed language is regular iff it is the closure of a star-connected language (cf. [16, Thm. 6.3.13]). This means that, for any regexp $E$, the language $[E]^f$ is regular iff there exists a (generally different!) star-connected expression $E'$ such that $[E]^f = [E']^f$. As a corollary, a closed language is regular iff it is the closure of the concurrent denotation of some regexp (cf. [16, Thm. 6.3.16]).
4.5 Automaton Finiteness for Star-Connected Expressions

We now show that the set of Antimirov reordering parts-of-derivatives of a star-connected expression is finite modulo suitable equations.

- **Lemma 31.**  If $\llbracket E \rrbracket$ is connected, then, for every $u \in \Sigma^+$ and $E'$ such that $E \rightarrow^{I^*} (u, E')$, either $R_u^I E' \neq 0$ or $R_u^I E' \neq 1$ (using the equations involving 0 and 1 only and that 0 is zero).

- **Lemma 32.**  If $\llbracket E \rrbracket$ is connected and $E^* \rightarrow^{I^*} (u, E')$, then there exists $E''$ such that $E' \neq E''$ (using, additionally, the monoid equations for 1, · and the equation $F^* \cdot F^* \neq F^*$) and $E''$ contains at most $|\Sigma|$ subexpressions of the form $(R_X^I E)^*$ where $\emptyset \subset X \subseteq \Sigma$.

- **Proposition 33.**  If $E$ is star-connected, then a suitable sound quotient of the state set $\{E' \mid \exists u \in \Sigma^+. E \rightarrow^{I^*} (u, E')\}$ of the Antimirov automaton for $E$ (accepting $\llbracket E \rrbracket^I$) is finite.

5 Uniform Scattering Rank of a Language

We proceed to defining the notion of uniform scattering rank of a language and show that star-connected expressions define languages with uniform scattering rank.

5.1 Scattering Rank vs. Uniform Scattering Rank

The notion of scattering rank of a language (a.k.a. distribution rank, $k$-block testability) was introduced by Hashiguchi [7].

- **Definition 34.**  A language $L$ has (I-scattering) rank at most $N$ if 
  \[ \forall u, v. uv \in [L]^I \implies \exists z \in L. u \sim_{\llcorner N} z \triangleright v. \]

  Hashiguchi [7] showed that having rank is a sufficient condition for regularity of the trace closure of a regular language (cf. [16, Prop. 6.3.2]). But it is not a necessary condition: for $\Sigma =_{df} \{a, b\}$, $aIb$, the language $L =_{df} \llbracket (aa + ab + ba + bb)^* \rrbracket$ has rank 1 (as does any nontrivial closed language), but it is not star-connected.

  We wanted to show that a truncation of the refined Antimirov automaton (which we define in Section 6) is finite for regexps whose language has rank (i.e., the language has rank at most $N$ for some $N \in \mathbb{N}$). But it turns out, as we shall see, that rank does not quite work for this. For this reason, we introduce a stronger notion that we call uniform scattering rank.

- **Definition 35.**  A language $L$ has uniform (I-scattering) rank at most $N$ if 
  \[ \forall w \in [L]^I. \exists z \in L. \forall u, v. w = uv \implies u \sim_{\llcorner N} z \triangleright v. \]

  The difference between the two definitions is that, in the uniform case, the choice of $z$ depends only on $w$ whereas, in the non-uniform case, it depends on the particular split of $w$ as $w = uv$, i.e., for every such split of $w$ we may choose a different $z$.

- **Lemma 36.**  If $L$ has uniform rank at most $N$, then $L$ has rank at most $N$.

  The converse of the above lemma does not hold – there are languages with uniform rank greater than rank. Furthermore, there are languages that have rank but no uniform rank.

- **Proposition 37.**  Let $\Sigma =_{df} \{a, b, c\}$, $aIb$ and $E =_{df} a^* b^* c(ab)^*(a^* + b^*) + (ab)^*(a^* + b^*)ca^* b^*$. The language $\llbracket E \rrbracket$ has rank 2, but no uniform rank.
5.2 Star-Connected Languages Have Uniform Rank

Klunder et al. [10] established that star-connected languages have rank. We will now show that star-connected languages also have uniform rank, by strengthening their proof.

It can be seen that if $L_1$ and $L_2$ have uniform rank at most $N_1$ and $N_2$, then $L_1 \cup L_2$ has uniform rank at most $\max(N_1, N_2)$ and $L_1 \cdot L_2$ has uniform rank at most $N_1 + N_2$. If a general $L$ has uniform rank at most $N$, then $L^*$ need not have uniform rank. For example, for $\Sigma = \{a, b\}$, the language $\{ab\}$ has uniform rank 1, but $\{ab\}^*$ is without rank, so also without uniform rank. But if $L$ is also connected, then $L^*$ has uniform rank at most $(N + 1) \cdot |\Sigma|$. 

Proposition 38. If $E$ is star-connected, then the language $[E]$ has uniform rank.

6 Antimirov Reordering Derivative and Uniform Rank

We have seen that the reordering language derivative $D^IL$ allows $u$ to be scattered in a word $z \in L$ as $u_1, \ldots, u_n < z > v_0, \ldots, v_n$ where $u \sim^I u_1 \ldots u_n$. We will now consider a version of the Antimirov reordering derivative operation that delivers lists of regexps for the possible $v_0, \ldots, v_n$ rather than just single regexps for their concatenations $v_0 \ldots v_n$.

6.1 Refined Antimirov Reordering Derivative

The refined reordering parts-of-derivative of a regexp $E$ along a letter $a$ are pairs of regexps $E_l, E_r$. For any word $w = av \in [E]^I$, there must be an equivalent word $z = v_0av_r \in [E]$. Instead of describing the words $v_0v_r$, obtainable by removing a minimal occurrence of $a$ in a word $z \in [E]$, the refined parts-of-derivative describe the subwords $v_l, v_r$ that were to the left and right of this $a$ in $z$: it must be the case that $v_l \in [E_l]$ and $v_r \in [E_r]$ for one of the pairs $E_l, E_r$. For a longer word $u$, the refined reordering derivative operation gives lists of regexps $E_0, \ldots, E_n$ fixing what the lists of subwords $v_l, v_r$ can be in words $z = v_0u_1 \ldots u_nv_n \in [E]$ equivalent to a given word $w = wv \in [E]^I$.

Definition 39. The (unbounded and bounded) refined Antimirov $I$-reordering parts-of-derivatives of a regexp along a letter and a word are given by relations $\Rightarrow^I \subseteq \text{RE} \times \Sigma \times \text{RE}$, $\Rightarrow^I \subseteq \text{RE}^+ \times \Sigma \times \text{RE}^+$, $\rightarrow^I \subseteq \text{RE} \times \Sigma^* \times \text{RE}^+$, $\Rightarrow^I_N \subseteq \text{RE}^+\leq_{N+1} \times \Sigma \times \text{RE}^+\leq_{N+1}$, and $\rightarrow^I_N \subseteq \text{RE} \times \Sigma^* \times \text{RE}^+\leq_{N+1}$ defined inductively by

\[
\begin{align*}
E \rightarrow^I (a; E_l, E_r) & \quad \frac{E \rightarrow^I (a; E_l, E_r)}{E + F \rightarrow^I (a; E_l, E_r)} \quad \frac{F \rightarrow^I (a; E_l, E_r)}{E + F \rightarrow^I (a; E_l, E_r)} \\
E \rightarrow^I (a; E_l, E_r) & \quad \frac{E \rightarrow^I (a; E_l, E_r)}{E \rightarrow^I (a; F_1, F_r)} \quad \frac{F \rightarrow^I (a; F_l, F_r)}{E \rightarrow^I (a; F_l, F_r)} \\
E \rightarrow^I (a; E_l, E_r) & \quad \frac{E \rightarrow^I (a; E_l, E_r) \quad \frac{E \rightarrow^I (a; E_l, E_r)}{E \rightarrow^I (a; E_l, E_r)}}{E \rightarrow^I (a; E_l, E_r)} \quad \frac{E \rightarrow^I (a; E_l, E_r)}{E \rightarrow^I (a; E_l, E_r)}
\end{align*}
\]

By $\text{RE}^+\leq_{N+1}$ we mean nonempty lists of regexps of length at most $N + 1$. The relations $\Rightarrow^I$ and $\rightarrow^I_N$ are defined exactly as $\Rightarrow^I_N$ and $\rightarrow^I_N$ but with the condition $|\Gamma; \Delta| < N$ of the first rule of $\Rightarrow^I_N$ dropped. The operation $R^I_a$ is extended to lists of regexps in the obvious way.
We have several rules for deriving a list of regexps along $a$. If $E$ is split into $E_l$, $E_r$ and neither of them is nullable, then, in the $N$-bounded case, we require that the given list is shorter than $N+1$ since the new list will be longer by 1. If one of $E_l$, $E_r$ is nullable, not the first resp. last in the list and we choose to drop it, then the new list will be of the same length. If both are nullable, not the first resp. last and we opt to drop both, then the new list will be shorter by 1. They must be droppable under these conditions to handle the situation when a word $z$ has been split as $v_0 u_1 v_1 \ldots u_k v_k u_{k+1} \ldots u_n v_n$ and $v_k$ is further being split as $v_{l} a v_{r}$ while $v_l$ or $v_r$ is empty. If $k \neq n$ and $v_l$ is empty, we must join $u_k$ and $a$ into $u_k a$. If $k \neq n$ and $v_r$ is empty, we must join $a$ and $u_{k+1}$ into $au_{k+1}$. If $k$ is neither 0 nor $n$ and both $v_l$ and $v_r$ are empty, we must join all three of $u_k$, $a$ and $u_{k+1}$ into $u_k a u_{k+1}$. The length of the new list of regexps is always at least 2.

**Proposition 40.** For any $E$,

1. for any $a \in \Sigma$, $v_l, v_r \in \Sigma^*$,
   \[v_l a \land v_{l} a v_r \in [E] \iff \exists E_l, E_r. E \rightarrow^I (a; E_l, E_r) \land v_l \in [E_l] \land v_r \in [E_r];\]

2. for any $u \in \Sigma^*$, $n \in \mathbb{N}$, $v_0 \in \Sigma^*$, $v_1, \ldots, v_{n-1} \in \Sigma^*$, $v_n \in \Sigma^*$,
   \[\exists z \in \Sigma^*, v_1, \ldots, v_n \in \Sigma^+. z \in [E] \land u \sim^I u_1 \ldots u_n \land u_1, \ldots, u_n \prec z \triangleright v_0, \ldots, v_n \iff \exists E_0, \ldots, E_n. E \rightarrow^{I+} (u; E_0, \ldots, E_n) \land \forall j. v_j \in [E_j].\]

**Proposition 41.** For any $E$,

1. for any $a \in \Sigma$, $v \in \Sigma^*$, the following are equivalent:
   a. $a v \in [E]^I$;
   b. $\exists v_l, v_r \in \Sigma^*. v \sim^I v_l v_r \land v_l a \land v_{l} a v_r \in [E]$;
   c. $\exists v_l, v_r \in \Sigma^*.
      \[v \sim^I v_l v_r \land \exists E_l, E_r. E \rightarrow^I (a; E_l, E_r) \land v_l \in [E_l] \land v_r \in [E_r];\]
   d. $\exists v_l, v_r \in \Sigma^*.
      \[v \in v_l \sim^I v_r \land \exists E_l, E_r. E \rightarrow^I (a; E_l, E_r) \land v_l \in [E_l] \land v_r \in [E_r];\]

2. for any $a \in \Sigma^*$, the following are equivalent:
   a. $u \in [E]^I$;
   b. $\exists z \in [E], u \sim^{<} z \triangleright u \sim^I v$;
   c. $\exists n \in \mathbb{N}, v_0 \in \Sigma^*$, $v_1, \ldots, v_{n-1} \in \Sigma^*$, $v_n \in \Sigma^$, $u \sim^I v_0 v_1 \ldots v_n \land \exists E_0, \ldots, E_n. E \rightarrow^{I+} (u; E_0, \ldots, E_n) \land \forall j. v_j \in [E_j];$
   d. $\exists n \in \mathbb{N}, v_0 \in \Sigma^*$, $v_1, \ldots, v_{n-1} \in \Sigma^*$, $v_n \in \Sigma^*$, $v \in v_0 \sim^I v_1 \sim^I \ldots \sim^I v_n \land \exists E_0, \ldots, E_n. E \rightarrow^{I+} (u; E_0, \ldots, E_n) \land \forall j. v_j \in [E_j];$

3. for any $a \in \Sigma^*$,
   \[u \in [E]^I \iff (u = \varepsilon \land E_\downarrow) \lor (u \neq \varepsilon \land \exists E_0, E_1. E \rightarrow^{I+} (u; E_0, E_1) \land E_0 \downarrow \land E_1 \downarrow).\]

**Corollary 42.** For any $E$ such that $[E]$ has uniform rank at most $N$,

1. for any $a \in \Sigma$, the following are equivalent:
   a. $u \in [E]^I$;
   b. $\exists z \in [E], u \sim^{<} w \sim^I u \sim a \sim u$;
   c. $\exists n \leq N, v_0 \in \Sigma^*$, $v_1, \ldots, v_{n-1} \in \Sigma^*$, $v_n \in \Sigma^*$, $u \sim^I v_0 v_1 \ldots v_n \land \exists E_0, \ldots, E_n. E \rightarrow_{N}^{I+} (u; E_0, \ldots, E_n) \land \forall j. v_j \in [E_j];$
   d. $\exists n \leq N, v_0 \in \Sigma^*$, $v_1, \ldots, v_{n-1} \in \Sigma^*$, $v_n \in \Sigma^*$, $v \in v_0 \sim^I v_1 \sim^I \ldots \sim^I v_n \land \exists E_0, \ldots, E_n. E \rightarrow_{N}^{I+} (u; E_0, \ldots, E_n) \land \forall j. v_j \in [E_j];$

2. for any $a \in \Sigma^*$,
   \[u \in [E]^I \iff (u = \varepsilon \land E_\downarrow) \lor (u \neq \varepsilon \land \exists E_0, E_1. E \rightarrow_{N}^{I+} (u; E_0, E_1) \land E_0 \downarrow \land E_1 \downarrow).\]
Example 43. We go back to Example 22. Recall that $E = ab + b$ and $E_b = ab + a0 + 0$. Here is one of the refined reordering parts-of-derivatives of $E^*$ along $bb$.

\[
\begin{align*}
\frac{b \to (b; 1, 1)}{ab \to (b; a1, 1)} & \quad \frac{b \to (b; 1, 1)}{ab \to (b; a1, 1)} \\
\frac{ab + b \to (b; a1, 1)}{aa + ab + b \to (b; a1, 1)} & \quad \frac{ab + b \to (b; a1, 1)}{aa + ab + b \to (b; a1, 1)} \\
E^* \to \frac{1}{2} (b; E_b^a(a1), 1E^*) & \quad E^* \to \frac{1}{2} (b; E_b^a(a1), 1E^*) \\
0 < 2 & \quad 1 < 2 \\
E^* \to \frac{1}{2} (b; E_b^a(a1), 1E^*) & \quad E^* \to \frac{1}{2} (b; E_b^a(a1), 1E^*) \\
E^* \to \frac{1}{2} (b; E_b^a(a1), 1E^*) & \quad E^* \to \frac{1}{2} (b; E_b^a(a1), 1E^*) \\
E = (a, E^*) & \quad E = (a, E^*) \\
\end{align*}
\]

In this example, we chose $N = 2$. The regexp $1(E_b^a(a1)) = (aa)^a a$ is not nullable, so we could not have dropped it. From here we cannot continue by deriving along a third $b$ by again taking it from the summand $ab$ of $E$ in $1E^*$, as this would produce another nondroppable $1(E_b^a(a1))$ and make the list too long (longer than 3). For example, we are not allowed to establish $w = ab ababababab \in [E^*]^*$ (by deriving $E^*$ along $w$ and checking if we can arrive at $E_b, E_1$ with both $E_b, E_1$ nullable), mandated by $z = ababababab \in [E^*]$. We are allowed to do so because of $z' = ababababababababab \in [E^*]$. The word $z$ is not useful since after the splits of $w$ as $w = u$ there is $u = ababababababababab$, which splits $z$ as $u \sim z \supset v$ scattering $u$ into 3 blocks as $z = ababababababababab$ (we underline the letters from $u$); the full sequence of these corresponding splits of $z$ is $abababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababababab
Reordering Derivatives of Trace Closures of Regular Languages

Definition 44. We define functions \((\_)^{++}\), \((\_)^{++}\), \((\_)^{++}\) : \(\mathcal{RE} \rightarrow \mathcal{PRE}\) by

\[
\begin{align*}
a^{++} &= \delta_t \{1\} \\
n^{++} &= \delta_t \emptyset \\
(E + F)^{++} &= \delta_t E^{++} + F^{++} \\
1^{++} &= \delta_t 0 \\
(\mathcal{E} \mathcal{F})^{++} &= \delta_t E^{++} + F^{++} + \mathcal{E} \cdot \mathcal{F}^{++} + \mathcal{F}^{++} \\
(\mathcal{E}^*)^{++} &= \delta_t E^{++} + \{\mathcal{E}^*\} \cdot E^{++} + E^{++} + \mathcal{E} \cdot (\{\mathcal{E}^*\} \cdot E^{++}) + (\mathcal{E}^{++} + \{\mathcal{E}^*\}) \cdot F^{++} \\
\mathcal{R} \mathcal{E}^{++} &= \delta_t \left( R^I_X E \mid X \subseteq \Sigma \right) \\
\mathcal{E}^{++} &= \delta_t R(\mathcal{E}^{++}) \\
\mathcal{E}^{++} &= \delta_t \left( E \right) \cup \mathcal{E}^{++}
\end{align*}
\]

Proposition 45. 
1. For any \(E\), the set \(E^{++}\) is finite.
2. For any \(E\) and \(X\), we have \((R^I_X E)^{++} \subseteq R^I_X (E^{++})\).
3. For any \(E, a\) and \(E_l, E_r\), if \(E \rightarrow^I (a; E_l, E_r)\), then \(E_l \in R^I_X (E^{++})\) and \(E_r \in E^{++}\).
4. For any \(E, E', X, a, E'_l, E'_r\), if \(E' \in R^I_X (E^{++})\) and \(E' \rightarrow^I (a; E'_l, E'_r)\), then \(E'_l \in R^I_X (E^{++})\) and \(E'_r \in R^I_X (E^{++})\).
5. For any \(E, u\) and \(E_0, \ldots, E_n\), if \(E \rightarrow^{I*} (u; E_0, \ldots, E_n)\), then \(\forall j. E_j \in E^{++}\).

Proposition 46. For every \(E\) and \(N\), the state set \(\{\Gamma \mid \exists u \in \Sigma^*. E \rightarrow_P (\mathcal{E} \mathcal{F}) \} \) of the \(N\)-truncated refined Antimirov automaton for \(E\) (accepting \([E]I\) if \([E]\) has uniform rank at most \(N\)) is finite.

Related Work

Syntactic derivative constructions for regular expressions extended with constructors for (versions of) the shuffle operation have been considered, for example, by Sulzmann and Thiemann [20] for the Brzozowski derivative and by Broda et al. [4] for the Antimirov derivative. This is relevant to our derivatives since \(L \cdot L'\) and \(L \cup L'\). Thus our Brzozowski and Antimirov reordering derivatives of \(EF\) must be between the classical Brzozowski and Antimirov derivatives of \(EF\) and \(E \cup F\).

Conclusion and Future Work

We have shown that the Brzozowski and Antimirov derivative operations generalize to trace closures of regular languages in the form of reordering derivative operations. The sets of Brzozowski resp. Antimirov reordering (parts-of-)derivatives of a regexp are generally infinite, so the deterministic and nondeterministic automata that they give, accepting the trace closure, are generally infinite. Still, if the regexp is star-connected, their appropriate quotients are finite. Also, the set of \(N\)-bounded refined Antimirov reordering parts-of-derivatives is finite without quotienting, and we showed that, if the language of the regexp has uniform rank at most \(N\), the \(N\)-truncated refined Antimirov automaton accepts the trace closure. We also proved that star-connected expressions define languages with finite uniform rank.

Our intended application for this is operational semantics in the context of relaxed memory (where, e.g., shadow writes, i.e., writes from local buffers to shared memory, can be reorderable with other actions). For sequential composition \(EF\) it is usually required that, to execute any action from \(F\), execution of \(E\) must have completed. In the jargon of derivatives, this is to say that for an action from \(E\) to become executable, what is left of \(E\) has to have become nullable (i.e., one can consider the execution of \(E\) completed). With reordering derivatives, we can execute an action from \(F\) successfully even when what is left of \(E\) is not nullable. It suffices that some sequence of actions to complete the residual of \(E\) is reorderable with the selected action of \(F\).
In the definitions of the derivative operations we only use $I$ in one direction, i.e., we do not make use of its symmetry. It would be interesting to see if our results can be generalized to the setting of semi-commutations [6] and which changes are required for that.

References

