Earliest Query Answering for Deterministic Stepwise Hedge Automata

Antonio Al Serhali, Joachim Niehren

Earliest Query Answering for Deterministic Stepwise Hedge Automata

Antonio Al Serhali and Joachim Niehren

Inria and University of Lille, France

Abstract. Earliest query answering (EQA) is the problem to enumerate certain query answers on streams at the earliest events. We consider *EQA* for regular monadic queries on hedges or nested words defined by deterministic stepwise hedge automata (dSHAs). We present an *EQA* algorithm for dSHAs that requires time $O(c\,m)$ per event, where m is the size of the automata and c the concurrency of the query. We show that our EQA algorithm runs efficiently on regular XPath queries in practice.

1 Introduction

Streaming algorithms for hedges or nested words were widely studied for complex event processing [11,7] and for transforming *XML* documents in an online manner [16,4]. The open end of a stream can be instantiated continuously. Therefore, streams can be seen as incomplete databases, for which the notion of *certain query answers (CQAs)* was widely studied [9]. An element is a certain query answer if it is selected by all completions of the incomplete database. For instance, the XPATH query `following::a[following::b]` selects all a-elements of a nested word (modeling an *XML* document) that are followed eventually by some b-element. On the stream $aabaabaaaaa\ldots$ all a-positions before the b are *CQAs* of this query, while those after are not.

Earliest query answering (*EQA*) is the problem of detecting *CQAs* on streams at the earliest event [6]. We study *EQA* for regular monadic queries which select nodes of trees or hedges. For this, we consider streams that elongate prefixes of nested words such as $aa\langle b\langle a\rangle\langle b\ldots$ to well-nested words. *EQA* requires to decide the existence of *CQAs* which is computationally hard problem even for tiny fragments of regular XPATH queries [3], since CQA is a universality problem concerning all completions of the stream. Gauwin et al. [6] showed that *EQA* can be done in polynomial time for monadic queries defined by deterministic nested word automata (dNWAs) [2,15]. Their algorithm, however, requires time $O(c\ n^2)$ per event, where the concurrency c is the number of alive candidates of the query (not of the algorithm) and n the number of the automaton states. In the worst case c may be the length of the stream for monadic queries, so the overall complexity may be quadratic in the size of the stream too.

For complex event processing [11,7] *EQA* has often been avoided, by restricting the class of queries such that the certainty of an answer candidate depends only on the past of the stream and not on its future [11,7]. This rules

out XPath query with filters such as `following::a[following::b]`. Riveros et al. [12] proposed instead to enumerate the query answers late at the end of the stream, which however requires to buffer a large number of candidates. For XML streaming, *EQA* was often approximated [16,4] so that all *CQAs* are eventually selected but not always earliest. Or else, as done by Saxon [8], only very restricted regular XPath queries were permitted.

A major bottleneck for automata based *EQA* algorithms in practice [4] was the difficulty to compile regular path queries on nested words to deterministic automata that are reasonably small. This problem was solved recently [1] based on stepwise hedge automata (SHAs) [13] and schema-based determinization [14]. SHAs naturally generalize on finite state automata for words and trees. They can recognize all regular languages of hedges equally to NWAs but don't have any explicit stack (such as tree automata). SHAs can always be determinized in a bottom-up and left-to-right manner. NWAs can also be determinized but differently, since their determinization has to deal with top-down processing via explicit stacks, often leading to a huge size increase.

The availability of dSHAs for regular path queries gave new hope for the feasibility of *EQA* in practice. For this, however, more efficient *EQA* algorithms are needed. In particular, the time per event should no more be quadratic in n and neither should the preprocessing time be cubic in n. Sakho [17] showed recently that *EQA* for boolean dSHA queries can be done time $O(m)$ per event, where m is the overall size of the automaton. This improvement relies on the fact, that the set of accessible states of an dSHA can be computed time $O(m)$, while for *d*NWA*s* it requires time $O(n^2)$ after $O(n^3)$ preprocessing, where n the number of the states of the automaton.

In the present paper, we present a new *EQA* algorithm for monadic dSHA queries. Our approach is to adapt the general idea's of Gauwin from dNWAs to dSHAs. This yields an *EQA* algorithm in time $O(c\ m)$ per event where c is the concurrency c of the query. Gauwin's quadratic factor n^2 is reduced to m while the cubic preprocessing in time $O(n^3)$ is removed. The algorithm obtained is more efficient than the best previous *EQA* algorithm, based on a reduction to Gauwin's *EQA* algorithm by compiling dSHAs to dNWAs in linear time.

We implemented our new *EQA* algorithm in the AStream tool and applied it the regular XPath queries from the XPathMark collection [5] scaling to huge documents, and to the regular XPath queries extracted from practical XSLT programs by Lick and Schmitz [10] but on smaller documents. It turns out that AStream runs efficiently on huge XML documents (>100GB) for all queries with low concurrency. Some queries can be answered in streaming mode where the best existing non earliest query answering algorithm failed to be earliest [4].

Outline. We start with preliminaries in Section 2, 3, and 4. An earliest membership tester for dSHAs is presented in Section 5 and a late streaming algorithm for answering monadic queries in Section 6. We then present our new *EQA* algorithm in Section 7 and discuss experimental results with AStream in Section 8. Complete proofs and supplementary material are given in the appendix.

2 Preliminaries

Let A and B be sets. A partial function $f : A \hookrightarrow B$ is a relation $f \subseteq A \times B$ that is functional. The domain of a partial function is $dom(f) = \{a \in A \mid f(a) \in B\}$. A total function $f : A \to B$ is a partial function $f : A \hookrightarrow B$ with $dom(f) = A$. Let $\mathbb{N}$ be the set of natural numbers including 0.

Words. Let alphabet Ω be a set. The set of words over Ω is $\Omega^* = \cup_{n\in\mathbb{N}}^{\infty}\Omega^n$. A word $(a_1, \ldots, a_n) \in \Omega^n$ is written as $a_1 \ldots a_n$. We denote the empty word of length $\varepsilon \in \Omega^0$ and by $v_1 \cdot v_2 \in \Omega^*$ the concatenation of two words $v_1, v_2 \in \Omega^*$. For any word $v \in \Omega^*$ let $prefs(v) \subseteq \Omega^*$ be the set of its prefixes. For any $v \in \Omega^*$ and $a \in \Omega$ let $\#_a(v)$ be the number of occurrences of a in v.

Hedges. Hedges are sequences of letters and trees $\langle h \rangle$ with some hedge h. More formally, a hedge $h \in \mathcal{H}_\Omega$ has the following abstract syntax:

$$h, h' \in \mathcal{H}_\Omega ::= \varepsilon \quad | \quad a \quad | \quad \langle h \rangle \quad | \quad h \cdot h' \qquad \text{where } a \in \Omega$$

We assume $\varepsilon \cdot h = h \cdot \varepsilon = h$ and $(h \cdot h_1) \cdot h_2 = h \cdot (h_1 \cdot h_2)$. Therefore, we consider any word in Ω^* as a hedge in $\mathcal{H}_\Omega$, i.e., $\Omega^* \ni aab = a \cdot a \cdot b \in \mathcal{H}_\Omega$. For any $h \in \mathcal{H}_\Omega$ and $a \in \Omega$ let $\#_a(h)$ be the number of occurrences of a in h. The size $|h|$ is the number of the letters and opening parenthesis of h. The nesting depth d of h is the maximal number of nested opening parenthesis of trees in h. The set of positions of a hedge $h \in \mathcal{H}_\Omega$ is $pos(h) = \{1, \ldots, |h|\}$.

Nested Words. Hedges can be identified with nested words, i.e., words over alphabet $\hat{\Omega} = \Omega \cup \{\langle, \rangle\}$ in which all parentheses are well-nested. This is done by the function $nw(h) : \mathcal{H}_\Omega \to (\Omega \cup \{\langle, \rangle\})^*$ such that: $nw(\varepsilon) = \varepsilon$, $nw(\langle h \rangle) = \langle \cdot nw(h) \cdot \rangle$, $nw(a) = a$, and $nw(h \cdot h') = nw(h) \cdot nw(h')$. The set of *nested word prefixes* is $nwprefs_\Omega = prefs(nw(\mathcal{H}_\Omega)) \subseteq \hat{\Omega}^*$. Note that nested word prefixes may lack closing parenthesis, in which case they are not well-nested.

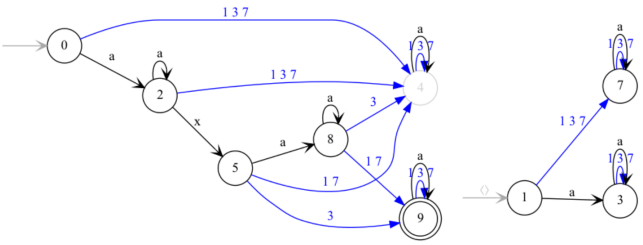
Monadic Queries. Let Σ be a set. A monadic query $\mathbf{Q}$ on hedges with signature Σ is a function that maps any hedge $h \in \mathcal{H}_\Sigma$ to a subset of its positions $\mathbf{Q}(h) \subseteq pos(h)$. We next relate monadic queries on hedges to hedge languages. For this, we fix a selection variable $x \notin \Sigma$ arbitrarily and consider hedge languages over signature $\Sigma^x = \Sigma \cup \{x\}$. For any $h \in \mathcal{H}_\Sigma$ let $\widetilde{h} \in \mathcal{H}_{\Sigma \cup pos(h)}$ be its annotation with its positions. For instance $\widetilde{aa\langle\rangle a} = a1a2\langle 3\rangle a4$. For any variable assignment $\alpha : \{x\} \hookrightarrow pos(h)$ we define the hedge $h * \alpha \in \mathcal{H}_{\Sigma^x}$ annotated with x by substituting in $\widetilde{h}$ the position $\alpha(x)$ by x and removing all other positions. For instance, $aa\langle\rangle a * [x/2] = aax\langle\rangle a$. The monadic query on hedges with signature Σ defined by a hedge language $L \subseteq \mathcal{H}_{\Sigma^x}$ is $qry_L(h) = \{\alpha(x) \mid \alpha : \{x\} \to pos(h),\ h * \alpha \in L\}$.

3 Stepwise Hedge Automata

We define regular hedge languages by stepwise hedge automata (dSHAs).

Definition 1. *A* dSHA *is a tuple* $A = (\Omega, \mathcal{Q}, \delta, q_{init}, F)$ *where* Ω *and* $\mathcal{Q}$ *are finite sets,* $q_{init} \in \mathcal{Q}$, $F \subseteq \mathcal{Q}$, *and* $\delta = ((a^\delta)_{a\in\Omega}, \langle\rangle^\delta, @^\delta)$ *where:* $a^\delta : \mathcal{Q} \hookrightarrow \mathcal{Q}$, $\langle\rangle^\delta \in \mathcal{Q}$, *and* $@^\delta : \mathcal{Q} \times \mathcal{Q} \hookrightarrow \mathcal{Q}$.

Fig. 1: A dSHA for the monadic query on hedges with letters in $\Sigma = \{a\}$ that selects the positions $1, \ldots, n-1$ on hedges of the form $a^n \cdot \langle h \rangle \cdot h'$ if h does not start with letter "a" and position n otherwise.



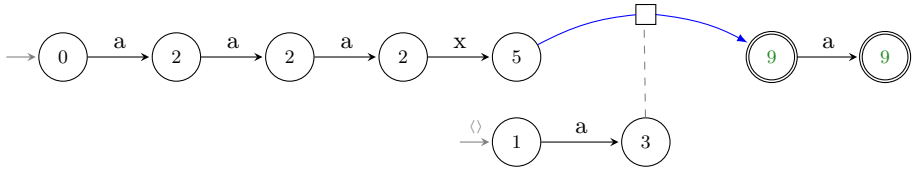


Fig. 2: A successful run of the dSHA A in Fig. 1 on $aaax\langle a \rangle a$.

There are states $q \in \mathcal{Q}$, an initial state is q_{init} and final states in F. The transition rules in δ have three forms: If $a^\delta(q) = q'$ then we have an internal rule $q \xrightarrow{a} q'$, if $q@^\delta p = q'$ then we have an apply rule $q \xrightarrow{p} q'$, and if $q = \langle\rangle^\delta \in \mathcal{Q}$ then we have a tree initial rule $\xrightarrow{\langle\rangle} q$. We denote by $n = |\mathcal{Q}|$ the number of states of A, and by $m = n + |\Omega| + \sum_{a \in \Omega} |a^\delta| + |@^\delta| + |F| + 2$ its overall size. Note that $m \in O(n^2 + |\Omega|\ n)$ by determinism. For any hedge $h \in \mathcal{H}_\Sigma$ we define the transition $[\![h]\!]^\delta = [\![h]\!] : \mathcal{Q} \hookrightarrow \mathcal{Q}$ such that for all $q \in \mathcal{Q}$, $a \in \Omega$, and $h, h' \in \mathcal{H}_\Sigma$:

$$[\![\varepsilon]\!](q) = q \qquad [\![a]\!](q) = a^\delta(q)$$
$$[\![h \cdot h']\!](q) = [\![h']\!]([\![h]\!](q)) \qquad [\![\langle h \rangle]\!](q) = q@^\delta([\![h]\!](\langle\rangle^\delta))$$

A hedge is accepted if its transition from the initial state yields some final state. The language $\mathcal{L}(A)$ is the set of all accepted hedges: $\mathcal{L}(A) = \{h \in \mathcal{H}_\Omega \mid [\![h]\!](q_{init}) \in F\}$. We call a hedge language $L \subseteq \mathcal{H}_\Omega$ regular if it can be defined by some dSHA. A monadic query over hedges in $\mathcal{H}_\Sigma$ is called regular if it is equal to qry_L for some regular hedge language $L \subseteq \mathcal{H}_{\Sigma^x}$.

For example let $\Omega = \Sigma^x$ where $\Sigma = \{a\}$. We draw in Fig. 1 the graph of a dSHA for the query on hedges in $\mathcal{H}_\Sigma$ that selects the positions $1, \ldots, n-1$ on hedges of the form $a^n \cdot \langle h \rangle \cdot h'$ if h does not start with letter "a" and position n otherwise. The drawing of dSHAs are similar as for usual finite state automata, except that now, edges may also be labeled by states and not only by letters.

A successful run of this automaton on the hedge $aaax\langle a \rangle a$ is given in Fig. 2. In state 5 the transition must suspend on the result of the evaluation of the subhedge, which is started by the tree initial rule $\xrightarrow{\langle\rangle} 1$. The two edges $5 \xrightarrow{\square} 9$

and 3 - - -□ are justified by the apply rule 5 $\xrightarrow{3}$ 9: the suspended computation in state 5 is resumed in state 9 when going up from the subtree in state 3.

The set of states that are accessible from a state $q \in \mathcal{Q}$ through some hedge is $acc^\delta(q) = \{q' \mid q' = [\![h]\!](q),\ h \in \mathcal{H}_\Omega\}$. For any $Q \subseteq \mathcal{Q}$, the set $acc^\delta(Q)$ can be computed in time $O(m)$ as well as $invacc^\delta(Q) = \{q' \mid q \in acc^\delta(q'),\ q \in Q\}$. A tree state is a state in $\mathcal{P} = acc^\delta(\langle\rangle^\delta)$. We call a set of transition rules δ complete if $@^\delta|_{\mathcal{Q}\times\mathcal{P}}$ is a total function, as well as all a^δ where $a \in \Omega$. For instance, the dSHA in Fig. 1 has the tree states $\mathcal{P} = \{1, 3, 7\}$. Note that δ is not complete since x^δ is not total. But its restriction to the letters in $\Sigma = \{a\}$ is complete due to the sink state 4.

4 Nested Word Automata

We define streaming algorithms for dSHAs by infinitary nested word automata ($dNWAs^\infty$). These have the advantage to run naturally in streaming mode, while being able to pass information top-down, bottom-up, and left to right. In contrast, dSHAs cannot pass any information top-down.

Definition 2. *An $dNWA^\infty$ is a tuple $B = (\Omega, \mathcal{Q}, \Gamma, \delta, q_{init}, F)$, where Ω, Γ and $\mathcal{Q}$ are sets, $q_{init} \in \mathcal{Q}$, $F \subseteq \mathcal{Q}$, and $\delta = ((a^\delta)_{a\in\Omega}, \langle^\delta, \rangle^\delta)$ contains partial functions $a^\delta : \mathcal{Q} \hookrightarrow \mathcal{Q}$, $\langle^\delta : \mathcal{Q} \hookrightarrow \Gamma \times \mathcal{Q}$, and $\rangle^\delta : \mathcal{Q} \times \Gamma \hookrightarrow \mathcal{Q}$. A dNWA is a $dNWA^\infty$ whose sets $\mathcal{Q}$, Ω, and Γ are finite.*

The elements of Γ are called stack symbols. The transition rules in δ have three forms: letter rules $q \xrightarrow{a} q'$, opening rules $q \xrightarrow{\langle\downarrow\gamma} q'$ for pushing a stack symbol if $\langle^\delta(q) = (q', \gamma)$ and closing rules $q \xrightarrow{\rangle\uparrow\gamma} q'$ popping a stack symbol if $\rangle^\delta(q, \gamma) = q'$. Any $dNWA^\infty$ defines a pushdown machine that runs on words with parenthesis. A configuration of this machine is a pair in $\mathcal{K} = \mathcal{Q} \times \Gamma^*$ of a state and a stack. For any word $v \in \hat{\Omega}^*$, we define a streaming transition $[\![v]\!]^\delta_{str} = [\![v]\!]_{str} : \mathcal{K} \hookrightarrow \mathcal{K}$ such that for all $q \in \mathcal{Q}$, stacks $s \in \Gamma^*$ and $\gamma \in \Gamma$:

$$
\begin{array}{ll}
[\![a]\!]_{str}(q, s) = (a^\delta(q), s) & [\![\rangle]\!]_{str}(q, s \cdot \gamma) = (\rangle^\delta(q, \gamma), s) \\
[\![\varepsilon]\!]_{str}(q, s) = (q, s) & [\![\langle]\!]_{str}(q, s) = (q', s \cdot \gamma) \text{ where } (q', \gamma) = \langle^\delta(q) \\
[\![v \cdot v']\!]_{str}(q, s) = [\![v']\!]_{str}(q', s') \text{ where } (q', s') = [\![v]\!]_{str}(q, s) &
\end{array}
$$

A word $v \in \hat{\Omega}^*$ is accepted if $[\![nw(h)]\!]_{str}(q_{init}, \varepsilon)$ yields a final configuration in $F \times \{\varepsilon\}$: The language of a $dNWA^\infty$ is the set of nested words that it accepts: $\mathcal{L}(B) = \{v \in \hat{\Omega}^* \mid [\![v]\!]_{str}(q_{init}, \varepsilon) \in F \times \{\varepsilon\}\}$. Since the initial and final stack are required to be empty it follows that any word $\mathcal{L}(B)$ is well-nested.

For any dSHA $A = (\Omega, \mathcal{Q}, \delta, q_{init}, F)$ we define the dNWA $A^{nwa} = (\Omega, \mathcal{Q}, \Gamma, \delta^{nwa}, q_{init}, F)$ with $\Gamma = \mathcal{Q}$ such that δ^{nwa} contains for all $a \in \Omega$ and $q, p \in \mathcal{Q}$ the rules $q \xrightarrow{a} a^\delta(q)$, $q \xrightarrow{\langle\downarrow q} \langle\rangle^\delta$, and $q \xrightarrow{\rangle\uparrow p} p@^\delta q$. In Fig. 5 we draw a run of the dNWA A^{nwa} for the dSHA in Fig. 1. The graph of this dNWA is in Fig. 6.

Lemma 3. $\mathcal{L}(A^{nwa}) = nw(\mathcal{L}(A))$.

Intuitively, the lemma follows since the run of any dSHA A can be identified with the run of the dNWA A^{nwa}. To illustrate this, we reconsider the run of the dSHA in Fig. 2 and compare it to the run of corresponding dNWA in Fig. 5. The transition $5 \dashrightarrow 1$ of A^{nwa} is justified by the opening rule $5 \xrightarrow{\langle\downarrow 5} 1$: when entering the subtree, the current state 5 is pushed to the stack and the computation is continued in state 1. The edges $5 \longrightarrow 9$ and $3 \dashrightarrow$ for the apply rule $5@3 \to 9$ of the dSHA are mimicked by the closing rule $3 \xrightarrow{\rangle\uparrow 5} 9$ of the dNWA: when going up from the subtree in state 3, the state 5 is popped from the stack and the computation continues in state 9.

5 Earliest Membership

A late streaming evaluator of a dSHA A on hedges $h \in \mathcal{H}_\Omega$ can be obtained by evaluating the dNWA A^{nwa} in streaming mode on the nested word of h, i.e., by testing $[\![nw(h)]\!]^{\delta^{nwa}}_{str}(q_{init}, \varepsilon) \in F \times \{\varepsilon\}$. In this manner, h is never loaded into the memory. Instead, only a state and a stack are stored at any event, i.e. at any prefix of $nw(h)$. The memory costs thus only depend on the depth of the hedge.

The decision of whether membership holds, however, is taken at the very end of the stream. Instead, we want to decide language membership at the earliest event when it becomes certain. We consider Σ-certain membership to languages $L \subseteq \mathcal{H}_\Omega$ where $\Omega \supseteq \Sigma$ as needed for certain query answering later on.

Definition 4. *Let $\Sigma \subseteq \Omega$ and $L \subseteq \mathcal{H}_\Omega$. A nested word prefix v with letter in Ω satisfies cert-mem$^L_\Sigma(v)$ if $\forall h \in \mathcal{H}_\Omega.\ (\exists w \in \hat{\Sigma}^*.\ v \cdot w = nw(h)) \to h \in L$.*

In other words, a nested word prefix v is Σ-certain for membership in $L \subseteq \mathcal{H}_\Omega$, if any completion of v with letters from Σ to a hedge in $\mathcal{H}_\Omega$ belongs to L. For instance, if $\Sigma = \{a\}$ then the prefix $v = aaax\langle a$ is Σ-certain for the language of the dSHA A with signature $\Omega = \{a, x\}$ in Fig. 1, since any completion of v without further x'es will be accepted by A.

Since certain membership is a universality property, we need to consider universal automata states. Given a state $q \in \mathcal{Q}$ let $A[q_{init}/q] = (\Omega, \mathcal{Q}, \delta, q, F)$ be obtained from A by replacing its initial state by q. We define:

$$q \in universal^A_\Sigma \Leftrightarrow \mathcal{H}_\Sigma \subseteq L(A[q_{init}/q])$$

In order to characterize universality by accessibility, we define for all $\mathcal{Q} \subseteq \mathcal{Q}$:

$$safe^\delta(\mathcal{Q}) = \{q \in \mathcal{Q} \mid acc^\delta(q) \subseteq \mathcal{Q}\}$$

If δ is complete then $safe^\delta(\mathcal{Q}) = \mathcal{Q} \setminus invacc^\delta(\mathcal{Q} \setminus \mathcal{Q})$, so it can be computed in $O(m)$. For any $\Sigma \subseteq \Omega$, let $\delta|_\Sigma$ be the restriction of δ to the letters of Σ, i.e., $\delta|_\Sigma = ((a^\delta)_{a\in\Sigma}, \langle\rangle^\delta, @^\delta)$.

Lemma 5. *Let $A = (\Omega, \mathcal{Q}, \delta, q_{init}, F)$ be a dSHA and $\Sigma \subseteq \Omega$ such that $\delta|_\Sigma$ is complete, and $q \in \mathcal{Q}$. Then: $q \in universal^A_\Sigma \Leftrightarrow q \in safe^{\delta|_\Sigma}(F)$.*

Safety can be used to detect certain language membership. For this we define for any $Q \subseteq \mathcal{Q}$ and $q \in \mathcal{Q}$ such that $q@^\delta p$ is well-defined for some $p \in \mathcal{Q}$:

$$d^\delta(q, Q) = \mathit{safe}^\delta(\mathit{dn}_{@^\delta}(q, Q)) \quad \mathit{where}\ \mathit{dn}_{@^\delta}(q, Q) = \{p \in \mathcal{Q} \mid\ q@^\delta p \in Q\}.$$

Note that if $q@^\delta p$ is undefined for all p then $d^\delta(q, Q)$ remains undefined too. We define the dNWA A^c_Σ for testing certain Σ-membership to $L(A)$ as follows.

$$\mathcal{Q}^c_\Sigma = \mathcal{Q} \times 2^{\mathcal{Q}} \ = \ \Gamma^c_\Sigma, \qquad {q_{init}}^c_\Sigma = (q_{init}, \mathit{safe}^{\delta|_\Sigma}(F)).$$

The transition rules in δ^c_Σ allow for all $S \subseteq \mathcal{Q}$, $q, p \in \mathcal{Q}$, and $a \in \Omega$:

$$(q, S) \xrightarrow{\langle\downarrow(q,S)} (\langle\rangle^\delta, d^{\delta|_\Sigma}(q, S)), \quad p \xrightarrow{\rangle\uparrow(q,S)} (q@^\delta p, S), \quad (q, S) \xrightarrow{a} (a^\delta(q), S).$$

Finally, let $A^c_\Sigma = (\Omega, \mathcal{Q}^c_\Sigma, \Gamma^c_\Sigma, \delta^c_\Sigma, {q_{init}}^c_\Sigma, F^c_\Sigma)$ where $F^c_\Sigma = F \times 2^{\mathcal{Q}}$. In the first component A^c_Σ behaves like A^{nwa}, while in the second component it computes safety information. Therefore, $L(A) = L(A^{nwa}) = L(A^c_\Sigma)$. We next show that the streaming evaluator of A^c_Σ detects certain membership at any time.

Proposition 6. *Let $A = (\Omega, \mathcal{Q}, \delta, q_{init}, \_)$ be a dSHA, $v \in \mathit{nwprefs}_\Omega$ a nested word prefix, and $\Sigma \subseteq \Omega$ such that $\delta|_\Sigma$ is complete. If $q \in \mathcal{Q}$ and $S \subseteq \mathcal{Q}$ such that $((q, S), \_) = [\![v]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$ then: $\mathit{cert\text{-}mem}^{\mathcal{L}(A)}_\Sigma(v) \Leftrightarrow q \in S$.*

We illustrate Proposition 6 in Fig. 7 at the dSHA A from Fig. 1. Recall that it has signature $\Omega = \Sigma^x$ where $\Sigma = \{a\}$. Given that $\delta|_\Sigma$ is complete, Σ-certain membership of $aaax\langle a\rangle a$ to $\mathcal{L}(A)$ can be detected at the earliest event $aaax\langle a$, by running the streaming evaluator of earliest automaton A^c_Σ. Note that the earliest automaton is a dNWA passing safety information top-down (while dSHAs cannot pass any information top-down). We have $\mathit{safe}^{\delta|_\Sigma}(\{9\}) = \{9\}$ and $d^{\delta|_\Sigma}(5, \{9\}) = \{3\}$. Hence $[\![aaax\langle]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma) = ((1, \{3\}), s)$ where the stack is $s = (5, \{9\})$. Since $1 \notin \{3\}$, membership is not yet Σ-certain. Indeed, the Σ-completion $aaax\langle\rangle$ is not accepted. After reading the next letter a, we have $[\![aaax\langle a]\!]^{\delta^c_\Sigma}_{str} = ((3, \{3\}), s)$. Since the current state 3 belongs to the current set of safe states $\{3\}$, membership is Σ-certain, i.e., membership of all completions without further x'es.

6 Late Monadic Query Answering

We study the problem of how to answer monadic queries on hedges in streaming mode, while selecting query answers lately at end of the stream.

Our algorithm will generate candidates $[x/\pi]$ binding the selection variable x to positions π of the input hedge. We want to formulate the streaming algorithm without fixing the input hedge a priori, thus we consider the infinite set of candidates $\mathit{Cands} = \{\alpha \mid \alpha : \{x\} \hookrightarrow \mathbb{N}\}$. Given a dSHA A with signature Σ^x and a hedge $h \in \mathcal{H}_\Sigma$, our algorithm computes the answer set $\mathit{qry}_{\mathcal{L}(A)}(h)$ in streaming mode. For this, we compile A to the late dNWA^∞ A^l and run the streaming

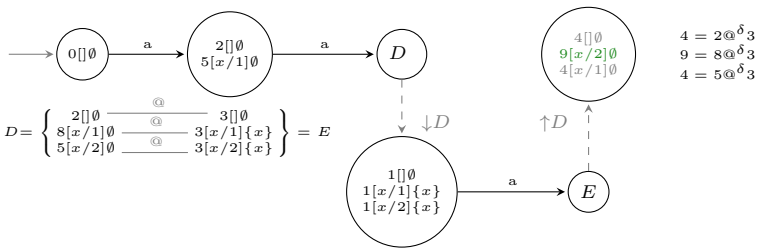


Fig. 3: The run of the late dNWA A^l for the dSHA A in Fig. 1 on $aa\langle a\rangle$.

evaluator of A^l on the nested word $\widetilde{nw(h)}$. The aphabet of A^l is $\Sigma \cup \mathbb{N}$. It has the state set $\mathcal{Q}^l = 2^{\mathcal{Q}\times Cands\times 2^{\{x\}}} = \Gamma^l$ and the initial state $q^l_{init} = \{(q_{init}, [], \emptyset)\}$. If the evaluator goes into some state $D \in \mathcal{Q}^l$, then $(q, \alpha, V) \in D$ means that the candidate α is in state q with A^{nwa} while the variables in $V \subseteq dom(\alpha)$ were bound in the context, i.e., before the last dangling opening parenthesis (so a preceding node that is not a preceding sibling). The set δ^l contains the following transition rules for all $D, E \in \mathcal{Q}^l$, $a \in \Sigma$, $V \subseteq \{x\}$, and $\pi \in \mathbb{N}$:

$$
\begin{aligned}
D &\xrightarrow{a} \{(a^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\}\\
D &\xrightarrow{\pi} \{(x^\delta(q), [x/\pi], \emptyset) \mid (q, [], \emptyset) \in D\} \cup D\\
D &\xrightarrow{\langle\downarrow D} \{(\langle\rangle^\delta, \alpha, dom(\alpha)) \mid (q, \alpha, V) \in D\}\\
E &\xrightarrow{\rangle\uparrow D} \{(q@^\delta p, \alpha', V) \mid (q, \alpha, V) \in D, (p, \alpha', dom(\alpha)) \in E, \alpha \in \{[], \alpha'\}\}
\end{aligned}
$$

When reading a position $\pi \in \mathbb{N}$ in a state D that contains a triple with the empty candidate $(q, [], \emptyset)$, a new candidate $[x/\pi]$ is created, and the triple $(x^\delta(q), [x/\pi], \emptyset)$ is added to D. At opening events, the current state D of A^l is pushed onto the stack. It is also updated for continuation: if D contains a triple with candidate α, then the next state of A^l contains $(\langle\rangle^\delta, \alpha, dom(\alpha))$. At closing events, the state D of the parent hedge is popped from the stack. Let E be the current state. Any $(q, \alpha, V) \in D$ must be matched with some $(p, \alpha', dom(\alpha)) \in E$, so that A^l can continue in $q@^\delta p$. Matching here means that either $\alpha = \alpha'$ or, $\alpha' = [x/\pi]$ and $\alpha = []$. This is expressed by the condition $\alpha \in \{[], \alpha'\}$. Note that if $\alpha = []$ matches $\alpha' = [x/\pi]$ then $dom(\alpha) = \emptyset$ so that π was not bound in the context. This is where the knowledge on the context is needed.

An example run of A^l on the hedge $aa\langle a\rangle$ is given in Fig. 3, where A is the dSHA A from Fig. 1. The tuples are written there without commas and parenthesis. The run of A^l first consumes aa and goes into state $D = \{(2, [], \emptyset), (8, [x/1], \emptyset), (5, [x/2], \emptyset)\}$. It contains the candidates $[x/1]$ and $[x/2]$ for the two leading a position, plus the empty candidate $[]$. After the following open parenthesis $\langle$, the run goes into the set $\{(1, [], \emptyset), (1, [x/1], \{x\}), (1, [x/2], \{x\})\}$. The state of each of the candidates got set to $\langle\rangle^\delta = 1$. Furthermore, the set memoizes that the candidates $[x/1]$ and $[x/2]$ were bound in the context. It then consumes the letter a and reaches $E = \{(3, [], \emptyset), (3, [x/1], \{x\}), (3, [x/2], \{x\})\}$.

When reading the closing parenthesis D is popped from the stack, its tuples in state q are matched with tuples in state p those from with E as illustrated in the figure, so that one can apply the apply rules $q@^\delta p$ of A. The tuple in state 5 of D, for instance, matches the tuple in state 3 of E, so A^l continues the candidate $[x/2]$ in state $9 = 5@^\delta 3$. Since $9 \in F^A$, position 2 is selected, i.e. $2 \in qry_{\mathcal{L}(A)}(aa\langle a\rangle)$.

Proposition 7 (Correctness of the late streaming evaluator). *If* $(D, \varepsilon) = [\![nw(\tilde{h})]\!]^{\delta^l}_{str}(q^l_{init}, \varepsilon)$ *then* $qry_{\mathcal{L}(A)}(h) = \{\pi \mid (q, [x/\pi], \emptyset) \in D,\ q \in F\}$.

7 Certain Answers and Earliest Query Answering

In order to justify early selection, we need the concept of certain answers. Let $\mathbf{Q}$ be a monadic query on $\mathcal{H}_\Sigma$ and $v \in nwprefs_\Sigma$ a nested word prefix.

Definition 8. *A position* $\pi \in \mathbb{N}$ *is a certain answer of* $\boldsymbol{Q}$ *at prefix* v *– written* $\pi \in CA^{\mathbf{Q}}(v)$ *– if* $\pi \in pos(v)\ \wedge\ \forall h \in \mathcal{H}_\Sigma.\ v \in prefs(nw(h)) \rightarrow \pi \in \boldsymbol{Q}(h)$.

A position π is thus a certain answer of query $\mathbf{Q}$ at prefix v of the stream if it answers the query for all completions h of v. Certain answers can be safely selected however the stream continues. For instance, the position 3 is a certain answer on the prefix $aaa\langle a$ for the query defined by the dSHA in Fig. 1.

In analogy we can define that π is certainly a nonanswer of $\mathbf{Q}$ at v, and denote this by $\pi \in CNA^{\mathbf{Q}}(v)$. Once π becomes a certain nonanswer then it can be safely rejected. The positions $1, ..., n-1$, for instance, are certain nonanswers on our example query on $a^n\langle a$. We call a position π alive for $\mathbf{Q}$ at v it it is neither a certain answer nor a certain nonanswer of $\mathbf{Q}$ at v. The *concurrency* c of $\mathbf{Q}$ at v is its number of alive candidates. For the shorter prefix $aaa\langle$, for instance, all n positions $1, \ldots, n$ are alive, so the concurrency is n.

We next want to link certain answers to certain Σ-membership. For this we need to annotate nested word prefixes with positions and variables, similarly as for hedges. Given a word $v \in \hat{\Sigma}$, the set of positions of v then is $pos(v) = \{1, \ldots, \#_{\Sigma\cup\{\langle\}}(v)\}$. We can define the annotation of v with its positions as a word $\widetilde{v} \in (\widehat{\Sigma} \cup pos(v))^*$. For any variable assignment $\alpha : \{x\} \hookrightarrow pos(v)$ we define an annotated word $v * \alpha \in \widehat{\Sigma^x}^*$ in analogy as for hedges.

Lemma 9. *For any prefix* $v \in prefs(nw(\mathcal{H}_\Sigma))$*, language* $L \subseteq \mathcal{H}_{\Sigma^x}$ *and candidate* $\alpha = [x/\pi]$ *with* $\pi \in pos(v)$*: cert-mem*$^L_\Sigma(v * \alpha) \Leftrightarrow \pi \in CA^{qry_L}(v)$.

Proposition 10 (Corollary of Proposition 6 and Lemma 9). *Let* $A = (\Sigma^x, \_, \delta, q_{init}, \_)$ *be a d*SHA *such that* $\delta|_\Sigma$ *is complete. For any* $v \in nwprefs_\Sigma$, $\pi \in pos(v)$, *and* $((q, S), \_) = [\![v * [x/\pi]]\!]^{\delta^c_\Sigma}_{str}(q_{init}{}^c_\Sigma, \varepsilon)$*:* $\pi \in CA^{qry_{\mathcal{L}(A)}}(v) \Leftrightarrow q \in S$.

For any dSHA A over Σ^x, we construct the earliest *dNWA*$^\infty$ A^e with alphabet $\Sigma \cup \mathbb{N}$, testing for all candidates $[x/\pi]$ on prefixes $\widetilde{v}$, whether π is certain for selection. For this A^e simulates the runs of A^c_Σ on all $v * [x/\pi]$. It has the states $\mathcal{Q}^e = \mathcal{Q} \times Cands \times 2^{\{x\}} \times 2^{\mathcal{Q}} = \Gamma^e$ and $q^e_{init} = \{(q_{init}, [], \emptyset, safe^{\delta|_\Sigma}(F))\}$. Initially,

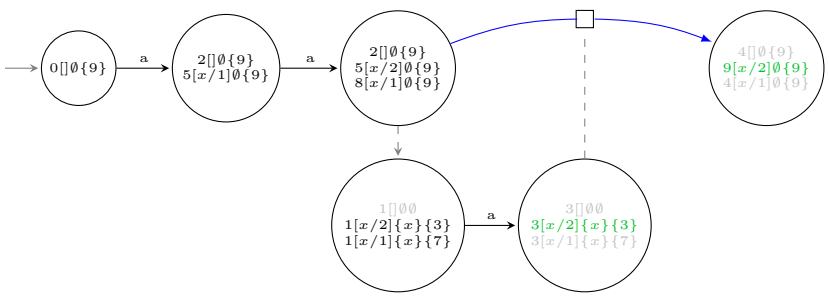


Fig. 4: A run of the earliest automaton A^e for the dSHA A in Fig. 1.

all states in $safe^{\delta|_\Sigma}(F)$ are safe for selection. Let δ^e contain the following rules for all $M, N \in \mathcal{Q}^e$, $a \in \Sigma$, $\pi \in \mathbb{N}$, $S \subseteq \mathcal{Q}$ and $q \in \mathcal{Q}$:

$$
\begin{array}{ll}
M & \xrightarrow{a} \ \{(a^\delta(q), \alpha, V, S) \mid (q, \alpha, V, S) \in M\} \\
M & \xrightarrow{\pi} \ \{(x^\delta(q), [x/\pi], \emptyset, S) \mid (q, [], \emptyset, S) \in M\} \cup M \\
M & \xrightarrow{\langle \downarrow M} \{(\langle\rangle^\delta, \alpha, dom(\alpha), d^{\delta|_\Sigma}(q, S)) \mid (q, \alpha, V, S) \in M\} \\
N & \xrightarrow{\rangle \uparrow M} (\{(q @^\delta p, \alpha', V, S) \mid (q, \alpha, V, S) \in M,\ (p, \alpha', dom(\alpha), S') \in N,\ \alpha \in \{[], \alpha'\}\}
\end{array}
$$

For the dSHA A in Fig. 1, for instance, a run of A^e is given in Fig. 4. It satisfies $[\![a1a2\langle 3a4]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon) = (\{(3, [x/2], \{x\}, \{3, 4\}), (3, [x/1], \{x\}, \{4, 9\})\}, \_)$. The certain answer $[x/2]$ is in a safe state now, while the other candidate $[x/1]$ can be seen to be safe for rejection so it could be removed.

Proposition 11. *Let A be a d*SHA *with signature Σ^x such that $\delta|_\Sigma$ is complete. For any nested word prefix $v \in nwprefs_\Sigma$ with $[\![\widetilde{v}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon) = (M, \_)$:*

$$CA^{qry_{\mathcal{L}(A)}}(v) = \{\alpha(x) \mid q \in S \wedge (q, \alpha, \emptyset, S) \in M \wedge dom(\alpha) = \{x\}\}$$

We can thus obtain an *EQA* algorithm by running the streaming evaluator of the earliest automaton A^e. Without removing candidates that are certainly nonanswers, however, it would maintain and update many candidates that are no more alive, leading to quadratic time in $O(m^2)$ even for bounded concurrency.

Theorem 1. *EQA for monadic d*SHA *queries can be done in time $O(c\ m)$ per event, where c is the concurrency of the query at the event.*

This complexity for dSHAs improves on Gauwin et al. [6] for dNWAs, which required time $O(c\ n^2)$ per event after $O(n^3)$ preprocessing time. Note also that *EQA* for monadic queries can also be used to detect certain membership for language of dSHAs. In this case, we have $c = 1$ so the time per event is reduced to $O(m)$. Finally, for monadic queries where c is bounded for all events and input hedges, the complexity per event is also reduced to $O(m)$.

8 Experimentation

We present experimental results of our *EQA* algorithm that we implemented in the AStream tool, with current version 3.36. The objective is to relate the theoretical complexity to practical efficiency. We note that we implemented AStream in Scala while using Java's abc-datalog for safety computation.

First, we consider a collection of 77 regular XPath queries [1] that was selected from the larger collection of XPath queries harvested by Lick and Schmitz [10] from real-world XSLT program. The queries are listed in Table 2. A single matching XML document per XSLT program of size less than 2MB was provided too. We used the dSHAs for these XPath queries from [1] as inputs (so the time for the automaton construction is ignored here). We could correctly answer all the 77 queries, yielding the same answer set as with Saxon. The overall time for computing the 77 answer sets was in 1:50 minute on a Macbook pro Apple M1 laptop with 16GB of RAM. With Saxon in-memory evaluation it required 0:45 minutes. The low running time of AStream reflects the low concurrency of all the queries on all these documents according to Theorem 1. There are 12 queries with concurrency 1, 47 with concurrency 2, 6 with concurrency 3, and 12 with concurrency 4. Our efficiency results for AStream thus show for the first time, that *EQA* is indeed feasible in practical scenarios with queries of low concurrency.

Second, we compare AStream to existing streaming tools for regular XPath queries with large coverage. We focus on the most efficient streaming evaluator called QuiXPath [4]. A detailed comparison to the many alternative tools is given there too. We note that QuiXPath is not always earliest, but still earliest in most cases. As done there, we consider the queries A1-A8 of the XPathMark collection [5], see Table 1. The other queries are either not regular or contain backward axis that our compiler to SHAs does not support. We also added the queries O1 and O2 from [4], in order to illustrate difficulties of non-earliest query answering and high concurrency. The XPathMark collection also provides a generator for scalable XML documents. We run AStream on XML documents of increasing size up to 1.2GB, but can also stream much larger documents >100GB. Up to 1GB we verified the correctness of the answer sets by comparison to Saxon's inmemory evaluator (which is limited to 1GB).

The running times on the scaling documents are reported in Fig. 8. The times grow linearly for all these queries except O1, given that their concurrency is bounded to 2. In contrast the time grows quadratically for query O1, since its concurrency grows linearly with the size of the document. The quadratic growth can be observed more clearly in Fig. 9 on smaller documents scaling from 27KB to 5MB. In average on the XPathMark queries A1-A8, AStream 3.36 is by a factor of 60 slower than QuiXPath, so requiring minutes instead of seconds. The main reason for this is the lack of streaming projection algorithms for dSHAs. In contrast, QuiXPath uses streaming projection for queries defined by dNWAs with selection states [18]. On the one hand side, QuiXPath cannot stream query O2 on large documents, since not being *earliest*. While the concurrency of O2 is 1, linearly many candidates are buffered by QuiXPath, until the buffer overflows

for documents larger than 5GB. For AStream's *EQA* algorithm, this query is not a problem since having low concurrency. On the other hand, QuiXPath can stream queries with high concurrency such as O1, where AStream 3.36 runs out of time for documents of 1MB. This is due to QuiXPath's state sharing, i.e. the sharing of the computations of all concurrent candidates in the same state.

Conclusion and Future Work. We introduced an *EQA* algorithm for regular monadic queries represented by dSHAs with a time complexity of $O(c\ m)$ per event. Its implementation in the AStream tool has demonstrated its efficiency on queries in practical scenarios with low concurrency. However, in order to compete with the current best non-earliest streaming algorithms, we need to develop streaming projection for dSHAs (as done previously for NWAs [18]), and to add some kind of factorization for candidates in the same state [4]. Additionally, we plan to extend our streaming algorithm to hyperstreaming, which involves handling multiple streams with references and holes [17].

References

1. A. Al Serhali and J. Niehren. A Benchmark Collection of Deterministic Automata for XPath Queries. In *XML Prague 2022*, Prague, Czech Republic, June 2022.
2. R. Alur. Marrying words and trees. *PODS 2007*.
3. M. Benedikt, A. Jeffrey, and R. Ley-Wild. Stream Firewalling of XML Constraints. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2008.
4. D. Debarbieux, O. Gauwin, J. Niehren, T. Sebastian, and M. Zergaoui. Early nested word automata for XPath query answering on XML streams. *TCS 2015*.
5. M. Franceschet. XPathMark performance test. `https://users.dimi.uniud.it/~massimo.franceschet/xpathmark/PTbench.html`. Accessed: 2020-10-25.
6. O. Gauwin, J. Niehren, and S. Tison. Earliest query answering for deterministic nested word automata. In *FCT 2009*.
7. A. Grez, C. Riveros, and M. Ugarte. A formal framework for complex event processing. *ICDT 2019*, pages 5:1–5:18, 2019.
8. M. Kay. Streaming in the saxon XSLT processor. In *XML Prague*, pages 81–101, 2014.
9. L. Libkin. How to define certain answers. In *IJCAI 2015*.
10. A. Lick and S. Sylvain. XPath Benchmark, Last visited April 13th 2022.
11. B. Mozafari, K. Zeng, and C. Zaniolo. High-performance complex event processing over XML streams. *SIGMOD Conference*. ACM, 2012.
12. M. Muñoz and C. Riveros. Streaming enumeration on nested documents. *ICDT 2022*.
13. J. Niehren and M. Sakho. Determinization and minimization of automata for nested words revisited. *Algorithms*, 14(3):68, 2021.
14. J. Niehren, M. Sakho, and A. A. Serhali. Schema-based automata determinization. *GandALF 2022*.
15. A. Okhotin and K. Salomaa. Complexity of input-driven pushdown automata. *SIGACT News*, 45(2):47–67, 2014.
16. D. Olteanu. SPEX: Streamed and progressive evaluation of XPath. *IEEE Trans. on Know. Data Eng.*, 19(7):934–949, 2007.
17. M. Sakho. *Certain Query Answering on Hyperstreams*. Theses, July 2020.
18. T. Sebastian and J. Niehren. Projection for nested word automata speeds up xpath evaluation on XML streams. In *SOFSEM 2016*.

A Proofs for Section 2 (Preliminaries)

Also, we define for any $n \in \mathbb{N}$ a hedge $\widetilde{h}^n \in \mathcal{H}_{\Sigma \cup \{n,\ldots,n+|h|-1\}}$ by induction on the structure of h:

$$\widetilde{\varepsilon}^n = \varepsilon, \quad \widetilde{a}^n = a \cdot n, \quad \widetilde{h \cdot h'}^n = \widetilde{h}^n \cdot \widetilde{h'}^{n+|h|}, \quad \widetilde{\langle h \rangle}^n = \langle n \cdot \widetilde{h}^{n+1} \cdot \rangle$$

We then set $\widetilde{h} = \widetilde{h}^1$. That is: $h * \alpha = \widetilde{h}[\alpha(x)/x][\pi/\varepsilon \mid \pi \neq \alpha(x)]$

B Proofs for Section 3 (Stepwise Hedge Automata)

Accessibility can be defined equivalently by the following inference rules:

$$\frac{q \xrightarrow{a} q' \quad a \in \Omega}{q' \in acc^\delta(q)} \qquad \frac{q_1 @^\delta p = q_2 \quad p \in acc^\delta(\langle\rangle^\delta) \quad q_1 \in acc^\delta(q)}{q_2 \in acc^\delta(q)}$$

For completing transition rules in linear time generally, we need to add else rules $q \xrightarrow{-} q'$ that represent a set of $|\Omega|$ internal rules and apply else rules $q \xrightarrow{-\!\!\!\rightarrow} q'$ that can represent a set of n apply rules. When doing so, $m = n^2 + |\Omega|\ n$ remains possible after completion.

C Proofs for Section 4 (Nested Word Automata)

Alternatively, $dNWAs^\infty$can also directly be run on hedges rather than on nested words. For this, we define transitions $[\![h]\!]^\delta = [\![h]\!] : \mathcal{Q} \hookrightarrow \mathcal{Q}$ for any hedge $h \in \mathcal{H}_\Sigma$, such that for all $q \in \mathcal{Q}$:

$$\begin{array}{ll} [\![\varepsilon]\!](q) = q & [\![h \cdot h']\!](q) = [\![h']\!]([\![h]\!](q)) \\ [\![a]\!](q) = a^\delta(q) & [\![\langle h \rangle]\!](q) = \rangle^\delta([\![h]\!](q'), \gamma) \text{ where } (q', \gamma) = \langle^\delta(q) \end{array}$$

Lemma 12. $[\![nw(h)]\!]_{str}(q, \varepsilon) = ([\![h]\!](q), \varepsilon)$.

Proof. First, we prove a more general equality for any $dNWA^\infty$ $B = (\Omega, \mathcal{Q}, \Gamma, \delta, q_{init}, F)$ and for all $h \in \mathcal{H}_\Omega$, $q \in \mathcal{Q}$ and $s \in \Gamma^*$ such that:

$$[\![nw(h)]\!]^\delta_{str}(q, s) = ([\![h]\!]^\delta(q), s)$$

The proof is done by induction on the structure of h. Let $s \in \Gamma^*$ and $q \in \mathcal{Q}$ be arbitrary, we then distinguish four cases of h:

Case $h = \varepsilon$**.** So $nw(h) = \varepsilon$, we then have:

$$[\![nw(h)]\!]_{str}(q, s) = [\![\varepsilon]\!]_{str}(q, s) = (q, s) = ([\![\varepsilon]\!](q), s) = ([\![h]\!](q), s)$$

Case $h = a$**.** With $nw(h) = a$, we have :

$$[\![nw(h)]\!]_{str}(q, s) = [\![a]\!]_{str}(q, s) = (a^\delta(q), s) = ([\![a]\!](q), s) = ([\![h]\!](q), s)$$

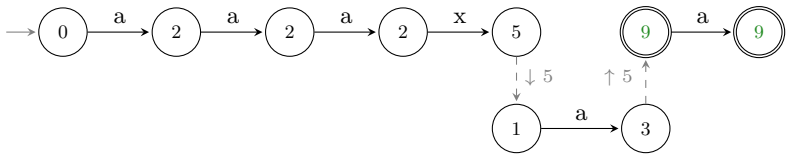


Fig. 5: A successful run of the dNWA A^{nwa} in Fig. 6 on $aaax\langle a\rangle a$.

Case $h = h_1 \cdot h_2$**.** Thus, $nw(h) = nw(h_1) \cdot nw(h_2)$. The induction hypothesis applied to h_1 yields $[\![nw(h_1)]\!]_{str}(q,s) = ([\![h_1]\!](q), s)$. Hence:

$$\begin{aligned}
[\![nw(h)]\!]_{str}(q,s) &= [\![nw(h_1)\cdot nw(h_2)]\!]_{str}(q,s) \\
&= [\![nw(h_2)]\!]_{str}([\![h_1]\!](q),s) && \text{streaming transition} \\
&= ([\![h_2]\!]([\![h_1]\!](q)),s) && \text{induction on } h_2 \\
&= ([\![h_1\cdot h_2]\!](q),s) && \text{NWA transition} \\
&= ([\![h]\!](q),s)
\end{aligned}$$

Case $h = \langle h_1\rangle$ **:** We have $nw(h) = \langle \cdot nw(h_1) \cdot \rangle$. Let $(q',\gamma) = \langle^{\delta}(q)$, thus:

$$\begin{aligned}
[\![nw(h)]\!]_{str}(q,s) &= [\![\rangle^{\delta}]\!]_{str}([\![nw(h_1)]\!]_{str}([\![\langle]\!]_{str}(q,s))) && \text{streaming transition} \\
&= [\![\rangle^{\delta}]\!]_{str}([\![nw(h_1)]\!]_{str}(q',s\cdot\gamma)) && \text{with } (q',\gamma) = \langle^{\delta}(q) \\
&= [\![\rangle^{\delta}]\!]_{str}([\![h_1]\!](q'),s\cdot\gamma) && \text{induction on } h_1 \\
&= (\rangle^{\delta}([\![h_1]\!](q'),\gamma),s) && \text{streaming transition} \\
&= ([\![\langle h_1\rangle]\!](q),s) && \text{NWA transition} \\
&= ([\![h]\!](q),s)
\end{aligned}$$

□

For $s = \varepsilon$, our lemma holds.

Lemma 13. *The language of nested words accepted by a dNWAs$^{\infty}$ satisfies:*

$$L(B) = \{nw(h) \mid h \in \mathcal{H}_\Omega,\ [\![h]\!](q_{init}) \in F\}$$

Proof. This follows immediately from Lemma 12.

Lemma 3. $\mathcal{L}(A^{nwa}) = nw(\mathcal{L}(A))$.

Proof. Based on Lemma 13 it is sufficient to show that:

$$\mathcal{L}(A) = \{h \in \mathcal{H}_\Omega \mid [\![h]\!]^{\delta^{nwa}}(q_{init}) \in F\}$$

This follows from $[\![h]\!]^{\delta}(q) = [\![h]\!]^{\delta^{nwa}}(q)$ for all hedge $h \in \mathcal{H}_\Omega$ and $q \in \mathcal{Q}$. So we prove this by induction on the structure of h.

Case $h = \varepsilon$**.** $[\![h]\!]^{\delta}(q) = [\![\varepsilon]\!]^{\delta}(q) = q = [\![\varepsilon]\!]^{\delta^{nwa}}(q) = [\![h]\!]^{\delta^{nwa}}(q)$
Case $h = a$**.** $[\![h]\!]^{\delta}(q) = [\![a]\!]^{\delta}(q) = a^{\delta}(q) = [\![a]\!]^{\delta^{nwa}}(q) = [\![h]\!]^{\delta^{nwa}}(q)$

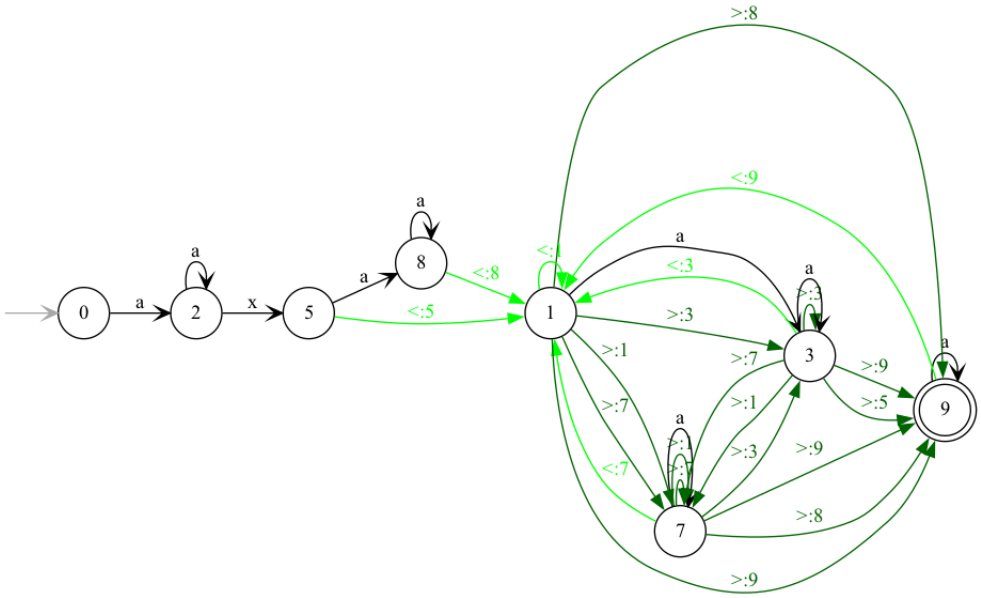


Fig. 6: The dNWA A^{nwa} for the dSHA in Fig. 1.

Case $h = h_1 \cdot h_2$**.** With the induction hypothesis applied to h_1 yields $[\![h_1]\!]^{\delta}(q) = [\![h_1]\!]^{\delta^{nwa}}(q)$:

$$\begin{aligned}
[\![h]\!]^{\delta}(q) &= [\![h_1 \cdot h_2]\!]^{\delta}(q) \\
&= [\![h_2]\!]^{\delta}([\![h_1]\!]^{\delta^{nwa}}(q)) && \text{SHA transition} \\
&= [\![h_2]\!]^{\delta^{nwa}}([\![h_1]\!]^{\delta^{nwa}}(q)) && \text{induction on } h_2 \\
&= [\![h_1 \cdot h_2]\!]^{\delta^{nwa}}(q) && \text{NWA transition} \\
&= [\![h]\!]^{\delta^{nwa}}(q)
\end{aligned}$$

Case $h = \langle h_1 \rangle$ **:** Let $(\langle\rangle^{\delta}, q) = \langle^{\delta}(q)$ by the rules of A^{nwa}, thus:

$$\begin{aligned}
[\![h]\!]^{\delta^{nwa}}(q) &= [\![\langle h_1 \rangle]\!]^{\delta^{nwa}}(q) \\
&= \rangle^{\delta}([\![h_1]\!]^{\delta^{nwa}}(\langle\rangle^{\delta}), q) && \text{NWA transition} \\
&= \rangle^{\delta}([\![h_1]\!]^{\delta}(\langle\rangle^{\delta}), q) && \text{induction on } h_1 \\
&= q@^{\delta}([\![h_1]\!]^{\delta}(\langle\rangle^{\delta})) && A^{nwa} \text{ rules} \\
&= [\![\langle h_1 \rangle]\!]^{\delta}(q) && \text{SHA transition} \\
&= [\![h]\!]^{\delta^{nwa}}(q)
\end{aligned}$$

□

D Proofs for Section 5 (Earliest Membership)

Lemma 5. *Let $A = (\Omega, \mathcal{Q}, \delta, q_{init}, F)$ be a dSHA and $\Sigma \subseteq \Omega$ such that $\delta|_{\Sigma}$ is complete, and $q \in \mathcal{Q}$. Then: $q \in universal^{A}_{\Sigma} \Leftrightarrow q \in safe^{\delta|_{\Sigma}}(F)$.*

Proof. Let $q \in \mathcal{Q}$ and $\Sigma \subseteq \Omega$. For the direction "$\Leftarrow$" we assume that $q \in safe^{\delta|_\Sigma}(F)$. To show $q \in universal^A_\Sigma$, we have to show for any $h \in \mathcal{H}_\Sigma$ that $h \in \mathcal{L}(A[q_{init}/q]))$. So let $h \in \mathcal{H}_\Sigma$ be arbitrary. Since $\delta|_\Sigma$ is complete, there exists $q' = [\![h]\!](q)$. Hence, $q' \in acc^\delta(q)$. Since $q \in safe^{\delta|_\Sigma}(F)$, this implies $q \in F$ and so $h \in \mathcal{L}(A[q_{init}/q]))$. For the direction "$\Rightarrow$" we assume that $q \in universal^A_\Sigma$. So $\mathcal{H}_\Sigma \subseteq \mathcal{L}(A[q_{init}/q])$, i.e., for any $h \in \mathcal{H}_\Sigma$ we have $[\![h]\!](q) \in F$. Hence $acc^\delta|_\Sigma(q) \subseteq F$, i.e., $q \in safe^{\delta|_\Sigma}(F)$. □

Proposition 6. *Let $A = (\Omega, \mathcal{Q}, \delta, q_{init}, \_)$ be a dSHA, $v \in nwprefs_\Omega$ a nested word prefix, and $\Sigma \subseteq \Omega$ such that $\delta|_\Sigma$ is complete. If $q \in \mathcal{Q}$ and $S \subseteq \mathcal{Q}$ such that $((q, S), \_) = [\![v]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$ then: cert-mem$^{\mathcal{L}(A)}_\Sigma(v) \Leftrightarrow q \in S$.*

Proof. Nested word prefixes may contain dangling opening parenthesis but no dangling closing parenthesis. We prove the equivalence for all v and A by induction on the number of dangling opening parenthesis in v. So let $((q, S), s) = [\![v]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$ for some stack $s \in (\Gamma^c_\Sigma)^*$.

In the base case, v does not have dangling opening parenthesis, so v is well-nested. The transition rules of δ^c_Σ change the states in the first component in the same manner than δ. Furthermore, the subset S of target states in the second component remains unchanged when reading letters in Ω. The same holds when processing trees $\langle w \rangle$ where w is well-nested: the target set S is pushed to the stack at the opening parenthesis and popped from the stack at the closing parenthesis for continuation. Therefore the assumption $[\![v]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon) = ((q, S), s)$ implies that $S = safe^{\delta|_\Sigma}(F)$, $s = \varepsilon$ and $[\![v]\!]^\delta(q_{init}) = q$. Hence, cert-mem$^{\mathcal{L}(A)}_\Sigma(v) \Leftrightarrow \mathcal{H}_\Omega \subseteq \mathcal{L}(A[q_{init}/q])$. By Lemma 5 and the completeness of $\delta|_\Sigma$, this is equivalent to $q \in safe^{\delta|_\Sigma}(F)$ and thus to $q \in S$.

For the induction step, we consider a nested word prefix v with at least one dangling opening parenthesis. We split v at the first dangling parenthesis such that $v = v_1 \cdot \langle \cdot v_2$ for some v_1 and v_2. This implies that $v_1 \in nw(\mathcal{H}_\Omega)$ in contrast to v_2. Let $((q_1, S_1), s_1) = [\![v_1]\!]^{\delta^c_\Sigma}_{str}(({q_{init}}^c_\Sigma, \varepsilon)$. Since the word v_1 is well-nested, we will reach the first dangling parenthesis with the empty stack, so $s_1 = \varepsilon$ and $S_1 = safe^{\delta|_\Sigma}(F)$. Reading the first dangling opening parenthesis in this configuration yields $[\![\langle]\!]^{\delta^c_\Sigma}_{str}((q_1, S_1), \varepsilon) = ((t, S_2), \gamma)$ where $S_2 = d^{\delta|_\Sigma}(q_1, S_1)$ and $\gamma = (q_1, S_1)$. Let $A_2 = A[q_{init}/\langle\rangle^\delta, F/F_2)]$ where $F_2 = dn_{@^\delta}(q_1, S_1)$. Let $I = (\langle\rangle^\delta, safe^{\delta|_\Sigma}(F_2))$ be the initial state of ${A_2}^c_\Sigma$. Note that $safe^{\delta|_\Sigma}(F_2) = d^{\delta|_\Sigma}(q_1, S_1)$. The assumption $[\![v]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon) = ((q, S), s)$ implies: $[\![v_2]\!]^{\delta^c_\Sigma}_{str}(I, \gamma) = ((q, S), s)$. Since the first dangling opening parenthesis of v will never be closed, the first stack symbol γ is never popped when reading v_2, so we have $s = \gamma \cdot s'$ for some stack s'. Therefore, γ can be canceled out, showing: $[\![v_2]\!]^{\delta^c_\Sigma}_{str}(I, \varepsilon) = ((q, S), s')$. Hence, cert-mem$^{\mathcal{L}(A)}_\Sigma(w)$ is equivalent to cert-mem$^{\mathcal{L}(A_2)}_\Sigma(w_2)$. By induction hypothesis applied to v_2 and A_2, this is equivalent to $q \in S$. □

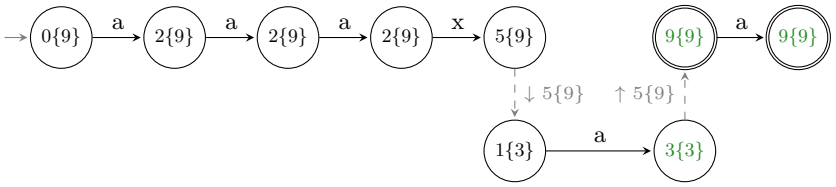


Fig. 7: A successful run of the dNWA A^c_Σ on $aaax\langle a\rangle a$.

E Proofs for Section 6 (Late Monadic Query Answering)

The correctness of the late automaton is stated by the following lemma.

Lemma 14. $[\![\widetilde{h}]\!]^{\delta^l}(q^l_{init}) = \{(q, \alpha, \emptyset) \mid q = [\![h * \alpha]\!]^\delta(q_{init}),\ \alpha : \{x\} \hookrightarrow pos(h)\}\}$

Proof. We call a state $D \in \mathcal{Q}^l$ admissible if any $(q, \alpha, V) \in D$ satisfies $V \subseteq dom(\alpha)$. Note that all states accessible from q^l_{init} via δ^l are admissible. For any $n \in \mathbb{N}$ and variable assignment $\alpha : \{x\} \hookrightarrow \{n, \ldots, n + |h| - 1\}$ we define a hedge $h *^n \alpha \in \mathcal{H}_{\Sigma^x}$ by substituting in $\widetilde{h}^n$ the position $\alpha(x)$ if defined by x and removing all other positions:

$$h *^n \alpha = \widetilde{h}^n[\alpha(x)/x][\pi/\varepsilon \mid \pi \neq \alpha(x)]$$

Clearly, $h * \alpha = h *^1 \alpha$. We call a state $D \in \mathcal{Q}^l$ applicable to h with offset n if $(\_, [x/\pi], V) \in D$ implies $\pi < n$. We will prove for all $h \in \mathcal{H}_\Sigma$, $n \in \mathbb{N}$, and admissible states $D \in \mathcal{Q}^l$ that are applicable with offset n:

$$[\![\widetilde{h}^n]\!]^{\delta^l}(D) = \begin{array}{l}\{([\![h *^n [x/\pi]]\!]^\delta(q), [x/\pi], \emptyset) \mid \pi \in \{n, \ldots, n + |h| - 1\},\ (q, [], \emptyset) \in D\} \\ \cup \{([\![h]\!]^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\}\end{array}$$

Applied with $D = q^l_{init}$ and $n = 0$ this claim yields exactly the lemma. The proof is by induction on the structure of h, using the facts that admissibility and applicability are maintained while the offsets are adapted.

So let $h \in \mathcal{H}_\Sigma$, $n \in \mathbb{N}$, and $D \in \mathcal{Q}^l$ an admissible state that is applicable with offset n. We distinguish cases according to all possible forms of h:

Case $h = a$**.** We have $\widetilde{h}^n = a \cdot n$ and $h *^n [x/n] = a \cdot x$.

$$\begin{aligned}
[\![\widetilde{h}^n]\!]^{\delta^l}(D) &= [\![a \cdot n]\!]^{\delta^l}(D) \\
&= \{(x^\delta(a^\delta(q)), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\
&\cup \{(a^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\} \\
&= \{([\![a \cdot n]\!]^\delta(q), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\
&\cup \{([\![a]\!]^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\} \\
&= \{([\![h *^n [x/n]]\!]^\delta(q), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\
&\cup \{([\![h]\!]^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\}
\end{aligned}$$

Case $h = \varepsilon$**.** We have $\widetilde{h}^n = \varepsilon$ and $h *^n [] = \varepsilon$.

$$\begin{aligned}
[\![\widetilde{h}^n]\!]^{\delta^l}(D) &= [\![\varepsilon]\!]^{\delta^l}(D) \\
&= D \\
&= \{([\![h *^n [x/\pi]]\!]^{\delta}(q), [x/\pi], \emptyset) \mid \pi \in \emptyset,\ (q, [], \emptyset) \in D\} \\
&\cup \{([\![h]\!]^{\delta}(q), \alpha, V) \mid (q, \alpha, V) \in D\}
\end{aligned}$$

Case $h = h_1 \cdot h_2$**.** Let $n_1 = n + |h_1|$. Then we have $\widetilde{h}^n = \widetilde{h_1}^n \cdot \widetilde{h_2}^{n_1}$. Hence:

$$[\![\widetilde{h}^n]\!]^{\delta^l}(D) = [\![\widetilde{h_2}^{n_1}]\!]^{\delta^l}([\![\widetilde{h_1}^n]\!]^{\delta^l}(D))$$

Let $D_1 = [\![\widetilde{h_1}^n]\!]^{\delta^l}(D)$, $D_2 = [\![\widetilde{h_2}^{n_1}]\!]^{\delta^l}(D_1)$ and $n_2 = n_1 + |h_2|$. The induction hypothesis applied to h_1, D, and n yields:

$$\begin{aligned}
D_1 = (&\{([\![h_1 *^n [x/\pi]]\!]^{\delta}(q), [x/\pi], \emptyset) \mid \pi \in \{n, \ldots, n_1 - 1\}, (q, [], \emptyset) \in D\} \\
&\cup \{([\![h_1]\!]^{\delta}(q), \alpha, V) \mid (q, \alpha, V) \in D\})
\end{aligned}$$

Note that D_1 is applicable with offset n_1. The induction hypothesis applied to h_2, D_1, and n_1 yields:

$$\begin{aligned}
D_2 = &\{([\![h_2 *^{n_1} [x/\pi]]\!]^{\delta}(q_1), [x/\pi], \emptyset) \mid \pi \in \{n_1, \ldots, n_2 - 1\}, (q_1, [], \emptyset) \in D_1\} \\
&\cup \{([\![h_2]\!]^{\delta}(q_1), \alpha, V) \mid (q_1, \alpha, V) \in D_1\}
\end{aligned}$$

Hence,

$$\begin{aligned}
&[\![\widetilde{h}^n]\!]^{\delta^l}(D) \\
&= \{([\![h_2 *^{n_1} [x/\pi]]\!]^{\delta}([\![h_1]\!]^{\delta}(q)), [x/\pi], \emptyset) \mid \pi \in \{n_1, \ldots, n_2 - 1\}, (q, [], \emptyset) \in D\} \\
&\cup \{([\![h_2]\!]^{\delta}([\![h_1 *^n [x/\pi]]\!]^{\delta}(q)), [x/\pi], \emptyset) \mid \pi \in \{n, \ldots, n_1 - 1\}, (q, [], \emptyset) \in D\} \\
&\cup \{([\![h_2]\!]^{\delta}([\![h_1]\!]^{\delta}(q)), \alpha, V) \mid (q, \alpha, V) \in D\}
\end{aligned}$$

Note that $[\![h_2 *^{n_1} [x/\pi]]\!]^{\delta}([\![h_1]\!]^{\delta}(q)) = ([\![h_1 \cdot h_2 *^n [x/\pi]]\!]^{\delta}(q)$ for any $\pi \in \{n, \ldots, n_2 - 1\}$, and similarly, $[\![h_2]\!]^{\delta}([\![h_1 *^n [x/\pi]]\!]^{\delta}(q)) = ([\![h_1 \cdot h_2 *^n [x/\pi]]\!]^{\delta}(q)$ for any $\pi \in \{n, \ldots, n_1 - 1\}$. Thus

$$\begin{aligned}
&[\![\widetilde{h}^n]\!]^{\delta^l}(D) \\
&\quad = \{([\![h_1 \cdot h_2 *^n [x/\pi]]\!]^{\delta}(q), [x/\pi], \emptyset) \mid \pi \in \{n, \ldots, n_2 - 1\}, (q, [], \emptyset) \in D\} \\
&\quad \cup \{([\![h_1 \cdot h_2]\!]^{\delta}(q), \alpha, V) \mid (q, \alpha, V) \in D\} \\
&\quad = \{([\![h *^n [x/\pi]]\!]^{\delta}(q), [x/\pi], \emptyset) \mid \pi \in \{n, \ldots, n + |h| - 1\}, (q, [], \emptyset) \in D\} \\
&\quad \cup \{([\![h]\!]^{\delta}(q), \alpha, V) \mid (q, \alpha, V) \in D\}
\end{aligned}$$

Case $h = \langle h_1 \rangle$**.** Thus $\widetilde{h}^n = \langle n \cdot \widetilde{h_1}^{n+1} \cdot \rangle$. Let:

$$D' = \{(\langle\rangle^{\delta}, \alpha, dom(\alpha)) \mid (q, \alpha, V) \in D\}$$

Thus $(D', D) = \langle^{\delta^l}(D)$ and:

$$\begin{aligned}
[\![\widetilde{h}^n]\!]^{\delta^l}(D) &= [\![\langle n \cdot \widetilde{h_1}^{n+1} \cdot \rangle]\!]^{\delta^l}(D) \\
&= \rangle^{\delta^l}([\![n \cdot \widetilde{h_1}^{n+1}]\!]^{\delta^l}(D'), D) \\
&= \rangle^{\delta^l}([\![\widetilde{h_1}^{n+1}]\!]^{\delta^l}([\![n]\!]^{\delta^l}(D')), D)
\end{aligned}$$

Let $D_1 = D' \cup \{(x^\delta(\langle\rangle^\delta), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\}$ and $D_2 = [\![\widetilde{h_1}^{n+1}]\!]^{\delta^l}(D_1)$. Then:

$$[\![\widetilde{h}^n]\!]^{\delta^l}(D) = \rangle^{\delta^l}([\![\widetilde{h_1}^{n+1}]\!]^{\delta^l}(D_1), D) = \rangle^{\delta^l}(D_2, D)$$

Let and $n_1 = n+|h_1|$.Note that D_1 is applicable with offset n_1. The induction hypothesis applied to h_1, D_1 and $n+1$ shows:

$$\begin{aligned} D_2 = {} & \{([\![h_1]\!]^\delta(q_1), \alpha, V) \mid (q_1, \alpha, V) \in D_1\} \\ & \cup \{([\![h_1 *^{n+1} [x/\pi]]\!]^\delta(q_1), [x/\pi], \emptyset) \mid \pi \in \{n+1, \ldots, n_1\}, (q_1, [], \emptyset) \in D_1\} \end{aligned}$$

Thereby and by the definition of D_1 we obtain: I put n as correction for x. Also $dom(\alpha)$ instead of V as for D'

$$\begin{aligned} D_2 = {} & \{([\![h_1]\!]^\delta(\langle\rangle^\delta)), \alpha, dom(\alpha)) \mid (q, \alpha, V) \in D\} \\ & \cup \{([\![h_1]\!]^\delta(n^\delta(\langle\rangle^\delta)), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\ & \cup \{([\![h_1 *^{n+1} [x/\pi]]\!]^\delta(\langle\rangle^\delta), [x/\pi], \emptyset) \mid \pi \in \{n+1, \ldots, n_1\}, (q, [], \emptyset) \in D\} \end{aligned}$$

With this equation, we can develop the right hand side of the equation $[\![\widetilde{h}^n]\!]^{\delta^l}(D) = \rangle^{\delta^l}(D_2, D)$ by applying the definition of the closing rule of δ^l: at closing time, we take back V, not $dom(\alpha)$

$$\begin{aligned} \rangle^{\delta^l}(D_2, D) \;=\; & \{(q@[\![h_1]\!]^\delta(\langle\rangle^\delta), \alpha, V) \mid (q, \alpha, V) \in D\} \\ & \cup \{(q@[\![n \cdot h_1]\!]^\delta(\langle\rangle^\delta), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\ & \cup \{(q@[\![h_1 *^{n+1} [x/\pi]]\!]^\delta(\langle\rangle^\delta), [x/\pi], \emptyset) \mid \pi \in \{n+1, \ldots, n_1\}, (q, [], \emptyset) \in D\} \end{aligned}$$

Note that the admissibility of D ensures above that $[x/n]$ cannot match with any $(q, [], \{x\}) \in D$. We next apply the definition of $[\![\langle . \rangle]\!]^\delta$ to show:

$$\begin{aligned} \rangle^{\delta^l}(D_2, D) \;=\; & \{([\![\langle h_1 \rangle]\!]^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\} \\ & \cup \{([\![\langle n \cdot h_1 \rangle]\!]^\delta(q), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\ & \cup \{([\![\langle h_1 *^{n+1} [x/\pi] \rangle]\!]^\delta(q), [x/\pi], \emptyset) \mid \pi \in \{n+1, \ldots, n_1\}, (q, [], \emptyset) \in D\} \end{aligned}$$

Since $h = \langle h_1 \rangle$ we get:

$$\begin{aligned} \rangle^{\delta^l}(D_2, D) \;=\; & \{([\![h]\!]^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\} \\ & \cup \{([\![h *^n [x/n]]\!]^\delta(q), [x/n], \emptyset) \mid (q, [], \emptyset) \in D\} \\ & \cup \{([\![h *^n [x/\pi]]\!]^\delta(q), [x/\pi], \emptyset) \mid \pi \in (n+1, \ldots, n_1), (q, [], \emptyset) \in D\} \\ \\ = {} & \{([\![h]\!]^\delta(q), \alpha, V) \mid (q, \alpha, V) \in D\} \\ & \cup \{([\![h *^n [x/\pi]]\!]^\delta(q), [x/\pi], \emptyset) \mid \pi \in (n, \ldots, |h|-1), (q, [], \emptyset) \in D\} \end{aligned}$$

Since, $[\![\widetilde{h}^n]\!]^{\delta^l}(D) = \rangle^{\delta^l}(D_2, D)$ the inductive statement follows.

We finally notice that the restriction of states D to be applicable with offset n is not strictly necessary. The claim remains true when omitting it. □

The late automaton thus computes the answer set of the query as follows:

Lemma 15. $qry_{\mathcal{L}(A)}(h) = \{\pi \mid (q, [x/\pi], \emptyset) \in [\![\tilde{h}]\!]^{\delta^l}(q^l_{init}),\ q \in F\}$

Proof. Let $\alpha = [x/\pi]$ for some $\pi \in \mathbb{N}$. By Lemma 14, we have that $q = [\![h * \alpha]\!]^{\delta}(q_{init})$ iff $(q, \alpha, \emptyset) \in [\![\widetilde{h}]\!]^{\delta^l}(q^l_{init})$. Hence:

$$\begin{aligned}\pi \in qry_{\mathcal{L}(A)}(h) &\text{ iff } \pi = \alpha(x) \text{ and } h * \alpha \in \mathcal{L}(A)\\ &\text{iff } \pi = \alpha(x) \text{ and } [\![h * \alpha]\!]^{\delta}(q_{init}) \in F\\ &\text{iff } \pi = \alpha(x) \text{ and } \exists q \in F.\ (q, \alpha, \emptyset) \in [\![\widetilde{h}]\!]^{\delta^l}(q^l_{init}).\end{aligned}$$ □

Proposition 7 (Correctness of the late streaming evaluator). *If* $(D, \varepsilon) = [\![nw(\tilde{h})]\!]^{\delta^l}_{str}(q^l_{init}, \varepsilon)$ *then* $qry_{\mathcal{L}(A)}(h) = \{\pi \mid (q, [x/\pi], \emptyset) \in D,\ q \in F\}$.

Proof. This is an immediate consequence of Lemma 15 and $[\![nw(\tilde{h})]\!]^{\delta^l}_{str}(q^l_{init}, \varepsilon) = ([\![\tilde{h}]\!]^{\delta^l}(q^l_{init}), \varepsilon)$ as shown in Lemma 12. □

For finalizing the definition of $A^l = (\Sigma \cup \mathbb{N}, \mathcal{Q}^l, \Gamma^l, \delta^l, q^l_{init}, F^l)$ we set $F^l = \{D \in \mathcal{Q}^l \mid (q, \alpha, \emptyset) \in D,\ q \in F, dom(\alpha) = \{x\}\}$.

Corollary 16. $qry_{\mathcal{L}(A)}(h) \neq \emptyset \Leftrightarrow nw(\tilde{h}) \in \mathcal{L}(A^l)$

Proof. By Proposition 7, $qry_{\mathcal{L}(A)}(h) \neq \emptyset$ if and only if there exists $q \in F$ and $\alpha : \{x\} \to \mathbb{N}$ such that $(q, \alpha, \emptyset) \in [\![\tilde{h}]\!]^{\delta^l}(q^l_{init})$. This is equivalent to $[\![\tilde{h}]\!]^{\delta^l}(q^l_{init}) \in F^l$ and thus to $\tilde{h} \in L(A^l)$. □

F Proofs for Section 7 (Certain Answers and Earliest Query Answering)

Definition 17. *A natural number* π *is certainly a nonanswer of a monadic query* $\boldsymbol{Q}$ *at nested word prefix* v*, denoted by* $\pi \in CNA^{\boldsymbol{Q}}(v)$*, if:*

$$\pi \in pos(v)\ \wedge\ \forall h \in \mathcal{H}_\Sigma.\ v \in prefs(nw(h)) \to \pi \notin \boldsymbol{Q}(h)$$

More generally, we define a position annotated word $\widetilde{v^n} \in (\Sigma \cup \{n, \ldots, n + |pos(v)| - 1\})^*$ for any $n \in \mathbb{N}$ such that $\widetilde{\varepsilon^n} = \varepsilon$ $\widetilde{a^n} = a \cdot n$, $\widetilde{v \cdot v'}^n = \widetilde{v^n} \cdot \widetilde{v'}^{n+|pos(v)|}$, $\widetilde{\langle}^n = \langle \cdot n$, and $\widetilde{\rangle}^n = \rangle$. By substituting in $\widetilde{v}$, the position $\alpha(x)$ by x if $x \in dom(\alpha)$, and removing all other positions: $v * \alpha = \widetilde{v}[\alpha(x)/x][\pi/\varepsilon \mid \pi \neq \alpha(x)]$.

Lemma 9. *For any prefix* $v \in prefs(nw(\mathcal{H}_\Sigma))$*, language* $L \subseteq \mathcal{H}_{\Sigma^x}$ *and candidate* $\alpha = [x/\pi]$ *with* $\pi \in pos(v)$*: cert-mem*$^L_\Sigma(v * \alpha) \Leftrightarrow \pi \in CA^{qry_L}(v)$.

Proof. For the forwards implication "⇒" we assume *cert-mem*$^L_\Sigma(v * \alpha)$. We fix $h \in \mathcal{H}_\Sigma$ arbitrarily such that $v \in prefs(nw(h))$. Then there exists $w \in \hat{\Sigma}^*$ such that $v \cdot w = nw(h)$. Let $h' = h * \alpha$. Since $h' = nw((v * \alpha) \cdot w)$, the certainty of Σ-membership of $v * \alpha$ for L yields $h' \in L$. For the backwards implication "⇐" we assume $\pi \in CA^{qry_L}(v)$. We fix $h' \in \mathcal{H}_{\Sigma^x}$ and $w \in \hat{\Sigma}^*$ arbitrarily such that $(v * \alpha) \cdot w \in nw(h')$. Let $h = nw(v \cdot w)$. Then $v \in prefs(nw(h))$, and since π is a certain answer of qry_L on v, it follows that $h * \alpha \in L$. Hence $h' = h * \alpha \in L$. □

Proposition 10 (Corollary of Proposition 6 and Lemma 9). *Let* $A = (\Sigma^x, \_, \delta, q_{init}, \_)$ *be a* d$\textsc{Sha}$ *such that* $\delta|_\Sigma$ *is complete. For any* $v \in nwprefs_\Sigma$, $\pi \in pos(v)$, *and* $((q,S),\_) = [\![v{*}[x/\pi]]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$: $\pi \in CA^{qry_{\mathcal{L}(A)}}(v) \Leftrightarrow q \in S$.

Proof. Let $\alpha = [x/\pi]$ and $((q,S),\_) = [\![v{*}\alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$. For the direction "$\Rightarrow$", we assume $\alpha(x) \in CA^{qry_{\mathcal{L}(A)}}(v)$. Lemma 9 yields $\textit{cert-mem}^{\mathcal{L}(A)}_\Sigma(v{*}\alpha)$. Since $\delta|_\Sigma$ is complete, Proposition 6 shows $q \in S$. For the other direction "$\Leftarrow$", we assume that $q \in S$. Since $\delta|_\Sigma$ is complete, Proposition 6 shows $\textit{cert-mem}^{\mathcal{L}(A)}_\Sigma(v * \alpha)$. Lemma 9 proves $\alpha(x) \in CA^{qry_{\mathcal{L}(A)}}(v)$. □

Lemma 18. *Let* A *be a* d$\textsc{Sha}$ *with signature* Σ^x *such that* $\delta|_\Sigma$ *is complete. For any nested word prefix* $v \in nwprefs_\Sigma$, *pair* $(M,\_) = [\![\widetilde{v}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$ *and variable assignment* $\alpha : \{x\} \to pos(v)$, $q \in \mathcal{Q}$ *and* $S \subseteq \mathcal{Q}$:

$$((q,S),\_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon) \Leftrightarrow (q, \alpha, \_, S) \in M$$

Proof. By induction on the structure of v. Assume $v \in nwprefs_\Sigma$, $(M,\_) = [\![\widetilde{v}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$, variable assignment $\alpha : \{x\} \to pos(v)$, $q \in \mathcal{Q}$ and $S \subseteq \mathcal{Q}$. In the base case, v is the empty word:

Case $v = \varepsilon$**.** Hence, $\widetilde{v} = \varepsilon$ and $[\![\widetilde{v}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon) = (q^e_{init}, \varepsilon)$. Also, $M = q^e_{init} = \{(q_{init}, [], \emptyset, \textit{safe}^{\delta|_\Sigma}(F))\}$. On the other hand, $[\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$ is defined only if $\alpha = []$, and in this case, it is equal to $({q_{init}}^c_\Sigma, \varepsilon) = ((q_{init}, \textit{safe}^{\delta|_\Sigma}(F)), \varepsilon)$ So $(q, \alpha, \_, S) \in M$ iff $q = q_{init}$, $\alpha = []$ and $S = \textit{safe}^{\delta|_\Sigma}(F)$ iff $((q,S),\_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$.

For the inductive step, there are three cases depending on the type of the last letter of v. Then, there are $a \in \Sigma$ or $v', v'' \in nwprefs_\Sigma$ such that:

Case $v = v' \cdot a$**.** So there exists $\pi \in \mathbb{N}$ such that $\widetilde{v} = \widetilde{v'} \cdot a \cdot \pi$. Let $(M',\_) = [\![\widetilde{v'}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$. We consider two cases depending on whether α binds x to the last position π or not:

- **Case** $\alpha = [x/\pi]$**.** We have $(q, \alpha, \_, S) \in M$ iff exists q' and q'' such that $(q', [], \emptyset, S) \in M'$ and $q' \xrightarrow{a} q'' \xrightarrow{x} q \in \delta$. By induction hypothesis, $(q', [], \emptyset, S) \in M'$ is equivalent to $((q',S),\_) = [\![v' * []]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$. Hence, $(q, \alpha, \_, S) \in M$ iff exists q' and q'' such that $((q',S),\_) = [\![v' * []]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$. and $q' \xrightarrow{a} q'' \xrightarrow{x} q \in \delta$. And this is equivalent to $((q,S),\_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$.
- **Case** $\alpha \neq [x/\pi]$**.** We have $(q, \alpha, \_, S) \in M$ iff exists q' such that $(q', \alpha, \_, S) \in M'$ and $q' \xrightarrow{a} q \in \delta$. By induction hypothesis, $(q', \alpha, \_, S) \in M'$ is equivalent to $((q',S),\_) = [\![v' * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$. Hence, $(q, \alpha, \_, S) \in M$ iff exists q' such that $((q',S),\_) = [\![v' * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$ and $q' \xrightarrow{a} q \in \delta$. And this is equivalent to $((q,S),\_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}({q_{init}}^c_\Sigma, \varepsilon)$.

Case $v = v' \cdot \langle$**.** Hence, $\widetilde{v} = \widetilde{v'} \cdot \langle$. Let $(M', \_) = [\![\widetilde{v'}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$. If α binds some position to x then it must be a position of $\widetilde{v'}$. So, $(q, \alpha, \_, S) \in M$ iff $q = \langle\rangle^\delta$ and there exist $q' \in \mathcal{Q}$ and $S' \subseteq \mathcal{Q}$ such that $(q', \alpha, \_, S') \in M'$ and $d^{\delta|_\Sigma}(q', S') = S$. By induction hypothesis, $(q', \alpha, \_, S') \in M'$ is equivalent to $((q', S'), \_) = [\![v' * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}{}^c_\Sigma, \varepsilon)$. Hence, $(q, \alpha, \_, S) \in M$ iff $q = \langle\rangle^\delta$ and there exist $q' \in \mathcal{Q}$ and $S' \subseteq \mathcal{Q}$ such that $((q', S'), \_) = [\![v' * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}{}^c_\Sigma, \varepsilon)$ and $d^{\delta|_\Sigma}(q', S') = S$. The latter is equivalent to $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}{}^c_\Sigma, \varepsilon)$.

Case $v = v' \cdot \langle \cdot\, v'' \cdot \rangle$ **and** v'' **is well-nested.** Let $n = \#_\Sigma(v)$. We have $\widetilde{v} = \widetilde{v'} \cdot \langle \cdot\, n \cdot \widetilde{v''}^{n+1} \rangle$. Let s be the stack in the second component of $(M, \_)$. The run of A^e leading to (M, s) must have the following form: Let $(M', s) = [\![\widetilde{v'}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$, and

$$M_0 = \{(q_{init}, \alpha', dom(\alpha'), d^{\delta|_\Sigma}(q', S')) \mid (q', \alpha', \_, S') \in M'\}.$$

Let $(M'', \_) = [\![n \cdot \widetilde{v''}]\!]^{\delta^e}_{str}(M_0, s \cdot M')$. We distinguish two cases, depending on whether α binds x to a position of v' or not.

Case $\alpha = [x/\pi]$ **where** $\pi \in pos(v')$**.** We have $v * \alpha = (v' * \alpha) \cdot \langle v'' \rangle$. We consider both implications independently.

"⇒" We suppose that $(q, \alpha, \_, S) \in M$ and have to show that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$. Since $(q, \alpha, \_, S) \in M$ there is $(q'', \alpha, \{x\}, S'') \in M''$, $(q', \alpha, \_, S) \in M'$ such that $S'' = d^{\delta|_\Sigma}(q', S)$ and $q = q' @^\delta q''$. By induction hypothesis, $(M', s) = [\![\widetilde{v'}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$ and $(q', \alpha, \_, S) \in M'$ imply that there exists s' such that $((q', S), s') = [\![v' * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$. The induction hypothesis applied to $(q'', \alpha, \{x\}, S'') \in M''$ implies that $((q'', S''), \_) = [\![v'']\!]^{\delta^c_\Sigma}_{str}(\langle\rangle^\delta, s' \cdot (q', S))$. Since v'' is well nested, the stack component of $((q'', S''), \_)$ must be unchanged, i.e. equal to $s' \cdot (q', S)$. Hence, we can apply the closing rule of A^c_Σ showing that $((q' @^\delta q'', S), s') = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$. Since $q' @^\delta q'' = q$ it follows that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$ as required.

"⇐" We suppose that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$ and have to show that $(q, \alpha, \_, S) \in M$. Let $((q', S'), \_) = [\![v' * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$. By induction hypothesis, we have $(q', \alpha, \_, S') \in M'$. Let $((q'', S''), \_) = [\![v'']\!]^{\delta^c_\Sigma}_{str}(q_{init}, s \cdot (q', S'))$. So $S'' = d^{\delta|_\Sigma}(q', S')$. By induction hypothesis, we have $(q'', [], \_, S'') \in M''$. Hence also $(q'', \alpha, \{x\}, S'') \in M''$. The closing rule permits to metch the tuples with α of M' and M'' yielding $(q' @^\delta q'', \alpha, \_, S') \in M$. Since α can only belong to a single tuple of M it follows that $q' @ q'' = q$ and $S' = S$. Hence, $(q, \alpha, \_, S) \in M$.

Case $\alpha = [x/n]$**.** So we have $v * \alpha = v' \cdot \langle x \cdot v'' \rangle$. We consider both implications independently.

"⇒" We suppose that $(q, \alpha, \_, S) \in M$ and have to show that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$. Since $(q, \alpha, \_, S)$ there are $(q'', \alpha, \{x\}, S'') \in M''$ and $(q', [], \emptyset, S) \in M'$ such that $S'' = d^{\delta|_\Sigma}(q', S)$ and $q = q' @^\delta q''$. By induction hypothesis, $(M', s) = [\![\widetilde{v'}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$ and $(q', [], \emptyset, S) \in M'$

imply that there exists s' such that $((q', S), s') = [\![v' * []]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$. The induction hypothesis applied to $(q'', \alpha, \{x\}, S'') \in M''$ implies that $((q'', S''), \_) = [\![x \cdot v'']\!]^{\delta^c_\Sigma}_{str}(\langle\rangle^\delta, s' \cdot (q', S))$. Since v'' is well nested, the stack component of $((q'', S''), \_)$ must be unchanged, i.e. equal to $s' \cdot (q', S)$. Hence, we can apply the closing rule of A^c_Σ showing that $((q' @^\delta q'', S), s') = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$. Since $q' @^\delta q'' = q$ it follows that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$ as required.

"⇐" We suppose that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$ and have to show that $(q, \alpha, \_, S) \in M$. Let $((q', S'), \_) = [\![v']\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$. By induction hypothesis, we have $(q', [], \_, S') \in M'$. Let $((q'', S''), \_) = [\![x \cdot v'']\!]^{\delta^c_\Sigma}_{str}(q_{init}, s \cdot (q', S'))$. So $S'' = d^{\delta|_\Sigma}(q', S')$. By induction hypothesis, we have $(q'', \alpha, \_, S'') \in M''$. The closing rule permits to match the tuple with α in M'' with the tuple with [] of M'. This yields $(q' @^\delta q'', \alpha, \_, S') \in M$. Since α can only belong to a single tuple of M it follows that $q' @ q'' = q$ and $S' = S$. Hence, $(q, \alpha, \_, S) \in M$.

Case else. So now either $\alpha = []$ or $\alpha = [x/\pi]$ with $\pi - n - 1 \in pos(v'')$ and we have $v * \alpha = v' * \cdot \langle v'' * \alpha \rangle$. We consider both implications independently.

"⇒" We suppose that $(q, \alpha, \_, S) \in M$ and have to show that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$. Since $(q, \alpha, \_, S)$ there are $(q'', \alpha, \emptyset, S'') \in M''$ and $(q', [], \emptyset, S) \in M'$ such that $S'' = d^{\delta|_\Sigma}(q', S)$ and $q = q' @^\delta q''$. By induction hypothesis, $(M', \_) = [\![\widetilde{v'}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon)$ and $(q', [], \_, S) \in M'$ imply that there exists s' such that $((q', S), s') = [\![v' * []]\!]^{\delta^c_\Sigma}_{str}(q_{init}, \varepsilon)$. The induction hypothesis applied to $(q'', \alpha, \_, S'') \in M''$ implies that $((q'', S''), \_) = [\![v'' * \alpha]\!]^{\delta^c_\Sigma}_{str}(\langle\rangle^\delta, s' \cdot (q', S))$. Since v'' is well nested, the stack component of $((q'', S''), \_)$ must be unchanged, i.e. equal to $s' \cdot (q', S)$. Hence, we can apply the closing rule of A^c_Σ showing that $((q' @^\delta q'', S), s') = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(init, s)$. Since $q' @^\delta q'' = q$ it follows that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(init, \varepsilon)$ as required.

"⇐" We suppose that $((q, S), \_) = [\![v * \alpha]\!]^{\delta^c_\Sigma}_{str}(init, s)$ and have to show that $(q, \alpha, \_, S) \in M$. Let $((q', S'), \_) = [\![v']\!]^{\delta^c_\Sigma}_{str}(q_{init}, s)$. By induction hypothesis, we have $(q', [], \_, S') \in M'$. Let $((q'', S''), \_) = [\![v'' * \alpha]\!]^{\delta^c_\Sigma}_{str}(q_{init}, s \cdot (q', S'))$. So $S'' = d^{\delta|_\Sigma}(q', S')$. By induction hypothesis, we have $(q'', \alpha, \_, S'') \in M''$. The closing rule permits to match the tuple with α in M'' with the tuple with [] of M'. This yields $(q' @^\delta q'', \alpha, \_, S') \in M$. Since α can only belong to a single tuple of M it follows that $q' @ q'' = q$ and $S' = S$. Hence, $(q, \alpha, \_, S) \in M$.

Proposition 11. *Let A be a d*SHA *with signature Σ^x such that $\delta|_\Sigma$ is complete. For any nested word prefix $v \in nwprefs_\Sigma$ with $[\![\widetilde{v}]\!]^{\delta^e}_{str}(q^e_{init}, \varepsilon) = (M, \_)$:*

$$CA^{qry_{\mathcal{L}(A)}}(v) = \{\alpha(x) \mid q \in S \wedge (q, \alpha, \emptyset, S) \in M \wedge dom(\alpha) = \{x\}\}$$

Proof. Let $\pi \in pos(v)$, $\alpha = [x/\pi]$, and $((q, S), \_) = [\![v * [x/\pi]]\!]^{\delta^c_\Sigma}_{str}(q_{init}{}^c_\Sigma, \varepsilon)$. By Lemma 18, we have $(q, \alpha, \_, S) \in M$. Proposition 10 shows that $\pi \in CA^{qry_{\mathcal{L}(A)}}(v)$

is equivalent to $q \in S$, and thus to $\pi \in \{\alpha(x) \mid q \in S \wedge (q, \alpha, \emptyset, S) \in M \wedge dom(\alpha) = \{x\}\}$. □

Certain nonanswers can be detected in analogy to certain answers. For this we can add a set of states that are safe for rejection. At the beginning, this is $safe^{\delta}(\mathcal{Q} \setminus F)$ lateron we have to use the function d^{δ} to update it when moving down into subtrees.

G Earliest Automaton with Safe Rejection

We next extend the earliest automaton with safe states for rejection. Given an dSHA A over Σ^x, we next construct the earliest automaton with rejection A^{e_r}. Its extends the states of A^e with a component with a subset of rejection states:

$$\mathcal{Q}^{e_r} = \mathcal{Q} \times \mathit{Cands} \times 2^{\{x\}} \times (2^{\mathcal{Q}})^2 = \Gamma^{e_r}$$
$$q^{e_r}_{init} = \{(\langle\rangle^{\delta}, [], \emptyset, \mathit{safe}^{\delta|_{\Sigma}}(F), \mathit{safe}^{\delta}(\mathcal{Q} \setminus F))\}.$$

At the beginning all states in $safe^{\delta}(\mathcal{Q} \setminus F)$ are safe for rejection. Let δ^{e_r} contain the following rules for all $M, N \in \mathcal{Q}^{e_r}$, $a \in \Sigma$, and $\pi \in \mathbb{N}$:

$$M \xrightarrow{a} \{(a^{\delta}(q), \alpha, V, S, R) \mid (q, \alpha, V, S, R) \in M\}$$
$$M \xrightarrow{\pi} \{(x^{\delta}(q), [x/\pi], \emptyset, S, R) \mid (q, [], \emptyset, S, R) \in M\} \cup M$$
$$M \xrightarrow{\langle\downarrow M} \{(\langle\rangle^{\delta}, \alpha, dom(\alpha), d^{\delta|_{\Sigma}}(q, S), d^{\delta}(q, R)) \mid (q, \alpha, V, S, R) \in M\}$$
$$N \xrightarrow{\rangle\uparrow M} (\{(q@^{\delta}p, \alpha', V, S, R) \mid (q, \alpha, V, S, R) \in M,\ (p, \alpha', dom(\alpha), S', R') \in N, \alpha \in \{[], \alpha'\}\}$$

Theorem 1. *EQA for monadic d*SHA *queries can be done in time $O(c\ m)$ per event, where c is the concurrency of the query at the event.*

Proof. We run the streaming evaluator of earliest automaton with safe rejection with two exceptions:

1. Whenever creating or updating a tuple (q, α, V, S, R) in the current set M, we test whether $x \in dom(\alpha)$ and $q \in S$, and if so we output $\alpha(x)$ and remove the tuple from M.
2. Whenever creating or updating a tuple (q, α, V, S, R) in the current set M, we test whether $q \in R$ and if so we remove the tuple from M.

This algorithm soundly outputs all certain query answers at the earliest time spoint by Proposition 11. It remains to argue the complexity per event. Since the candidates of all tuples in the current state M are pairwise distinct and alive, the cardinality of M is bounded by the number c of alive candidates of the query (not of the algorithm). For each tuple $(q, \alpha, V, S, R) \in M$, we need to compute $d^{\delta|_{\Sigma}}(q, S)$ and $d^{\delta}(q, R)$ or some transitions $a^{\delta}(q)$ and $x^{\delta}(q)$ where $a \in \Sigma$. The costs for this are in $O(m)$. □

H Proofs for Section 8 (Experimentation)

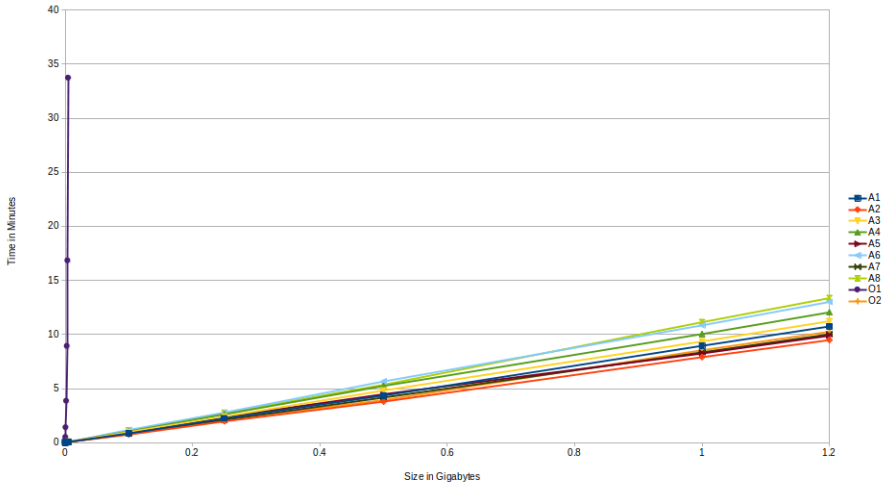


Fig. 8: Running times of AStream 3.36 for streaming XPathMark queries on XML documents whose size scale from 27 KB to 1.2GB. Only the running time of query O1 is not scaling up linearly, and thus running out of time for 128KB already. The concurrency of all other queries is bounded on these documents.

Table 1: XPathMark list of queries.

Id	XPath Query
A1:	`/site/closed_auctions/closed_auction/annotation/description/text/keyword`
A2:	`//closed_auction//keyword`
A3:	`/site/closed_auctions/closed_auction//keyword`
A4:	`/site/closed_auctions/closed_auction[annotation/description/text/keyword]/date`
A5:	`/site/closed_auctions/closed_auction[descendant::keyword]/date`
A6:	`/site/people/person[profile/gender and profile/age]/name`
A7:	`/site/people/person[phone or homepage]/name`
A8:	`/site/people/person[address and (phone or homepage) and (creditcard or profile)]/name`
O1:	`/site[closed auctions/closed auction/type]//item`
O2:	`/site[c or not(c)]//bidder`

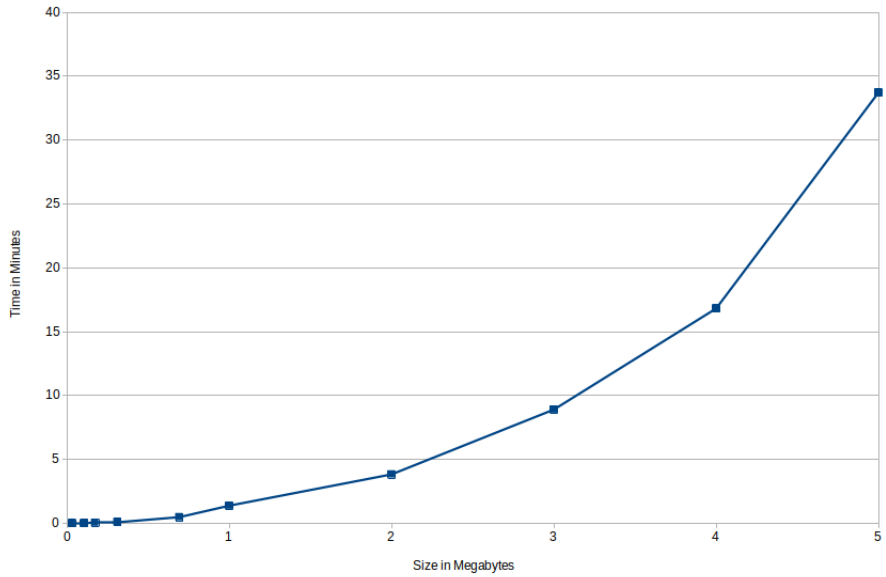


Fig. 9: The quadratic running time of AStream 3.36 on XPath query O1 with linear concurrency on XML documents ranging from 27 KB to 5 MB.

Table 2: The 77 queries of the XPath corpus of Lick and Schmitz without dublicates up to renaming.

Id	XPath Query
18330	/ descendant-or-self::node()/child::parts-of-speech
17914	/ descendant-or-self::node()/child::tei:back/descendant-or-self::node()/child::tei:interpGrp
10745	*//tei:imprint/tei:date[@type='access']
02091	* | .//refentry
00744	.//@id | .//@xml:id
12060	.//attDef
02762	.//authorgroup/author | .//author
06027	.//authorinitials | .//author
02909	.//bibliomisc[@role='serie']
06415	.//email | address/otheraddr/ulink
03257	.//equation[title or info/title]
05122	.//procedure[title]
09138	.//rng:ref | .//tei:elementRef | .//tei:classRef | .//tei:macroRef | .//tei:dataRef

05460	.//table//footnote | .//informaltable//footnote
12404	.//tei:dataRef[@name]
10337	.//tei:note[@place='end']
06639	.//tgroup//footnote
14340	//*
13804	//GAP/@DISP
13896	//HEADER//IDNO[@TYPE='evans citation']
02194	//annotation
06726	//doc:table | //doc:informaltable
13640	//equiv[@filter]
05735	//glossary[@role='auto']
15766	//h:body/h:section[@data-type='titlepage']
15524	//h:section[@data-type='titlepage']
06512	//refentry//text()
06176	//set | //book | //part | //reference | //preface | //chapter | //appendix | //article | //colophon | //refentry | //section | //sect1 | //sect2 | //sect3 | //sect4 | //sect5 | //indexterm | //glossary | //bibliography | //*[@id]
12539	//tei:elementSpec | //tei:classSpec[@type='atts']
11780	//tei:ref[@type='cite'] | //tei:ptr[@type='cite']
11478	//xhtml:p[@class]
11227	/tei:TEI/tei:text//tei:note[@type='action']
05684	@abbr | @align | @axis | @bgcolor | @border | @cellpadding | @cellspacing | @char | @charoff | @class | @dir | @frame | @headers | @height | @id | @lang | @nowrap | @onclick | @ondblclick | @onkeydown | @onkeypress | @onkeyup | @onmousedown | @onmousemove | @onmouseout | @onmouseover | @onmouseup | @rules | @scope | @style | @summary | @title | @valign | @valign | @width | @xml:id | @xml:lang
06947	anchor | areaset | audiodata | audioobject | beginpage | constraint | indexterm | itermset | keywordset | msg | doc:anchor | doc:areaset | doc:audiodata | doc:audioobject | doc:beginpage | doc:constraint | doc:indexterm | doc:itermset | doc:keywordset | doc:msg
06794	articleinfo | chapterinfo | bookinfo | doc:info | doc:articleinfo | doc:chapterinfo | doc:bookinfo
06169	article | preface | chapter | appendix | refentry | section | sect1 | glossary | bibliography
06924	authorblurb | formalpara | legalnotice | note | caution | warning | important | tip | doc:authorblurb | doc:formalpara | doc:legalnotice | doc:note | doc:caution | doc:warning | doc:important | doc:tip
11958	biblStruct//note
01705	book | article | part | reference | preface | chapter | bibliography | appendix | glossary | section | sect1 | sect2 | sect3 | sect4 | sect5 | refentry | colophon | bibliodiv[title] | setindex | index

02086	book | article | topic | part | reference | preface | chapter | bibliography | appendix | glossary | section | sect1 | sect2 | sect3 | sect4 | sect5 | refentry | colophon | bibliodiv[title] | setindex | index
02000	chapter | appendix | epigraph | warning | preface | index | colophon | glossary | biblioentry | bibliography | dedication | sidebar | footnote | glossterm | glossdef | bridgehead | part
02697	chapter | appendix | preface | reference | refentry | article | topic | index | glossary | bibliography
14183	content//rng:ref
07106	dbk:appendix | dbk:article | dbk:book | dbk:chapter | dbk:part | dbk:preface | dbk:section | dbk:sect1 | dbk:sect2 | dbk:sect3 | dbk:sect4 | dbk:sect5
05824	descendant-or-self::*
11368	descendant-or-self::tei:TEI/tei:text/tei:back
15848	descendant::*[@class='refname']
15462	descendant::h:span[@data-type='footnote']
04267	descendant::label
07113	following-sibling::*[self::dbk:appendix | self::dbk:article | self::dbk:book | self::dbk:chapter | self::dbk:part | self::dbk:preface | self::dbk:section | self::dbk:sect1 | self::dbk:sect2 | self::dbk:sect3 | self::dbk:sect4 | self::dbk:sect5] | following-sibling::dbk:para[@rnd:style = 'bibliography' or @rnd:style = 'bibliography-title' or @rnd:style = 'glossary' or @rnd:style = 'glossary-title' or @rnd:style = 'qandaset' or @rnd:style = 'qandaset-title']
03864	guibutton | guiicon | guilabel | guimenu | guimenuitem | guisubmenu | interface
15484	h:pre[@data-type='programlisting']//text()
15461	h:table[descendant::h:span[@data-type='footnote']]
11160	html:table | html:tr | html:thead | html:tbody | html:td | html:th | html:caption | html:li
06856	imageobject | imageobjectco | audioobject | videoobject | doc:imageobject | doc:imageobjectco | doc:audioobject | doc:videoobject
06458	info | refentryinfo | referenceinfo | refsynopsisdivinfo | refsectioninfo | refsect1info | refsect2info | refsect3info | setinfo | bookinfo | articleinfo | chapterinfo | sectioninfo | sect1info | sect2info | sect3info | sect4info | sect5info | partinfo | prefaceinfo | appendixinfo | docinfo
13710	persName | orgName | addName | nameLink | roleName | forename | surname | genName | country | placeName | geogName
06808	personname | surname | firstname | honorific | lineage | othername | contrib | doc:personname | doc:surname | doc:firstname | doc:honorific | doc:lineage | doc:othername | doc:contrib

04338	refsynopsisdiv/title | refsection/title | refsect1/title | refsect2/title | refsect3/title | refsynopsisdiv/info/title | refsection/info/title | refsect1/info/title | refsect2/info/title | refsect3/info/title
04358	section/title | simplesect/title | sect1/title | sect2/title | sect3/title | sect4/title | sect5/title | section/info/title | simplesect/info/title | sect1/info/title | sect2/info/title | sect3/info/title | sect4/info/title | sect5/info/title | section/sectioninfo/title | sect1/sect1info/title | sect2/sect2info/title | sect3/sect3info/title | sect4/sect4info/title | sect5/sect5info/title
13632	self::placeName | self::persName | self::district | self::settlement | self::region | self::country | self::bloc
01847	set | book | part | preface | chapter | appendix | article | reference | refentry | book/glossary | article/glossary | part/glossary | bibliography | colophon
05219	set | book | part | preface | chapter | appendix | article | topic | reference | refentry | book/glossary | article/glossary | part/glossary | book/bibliography | article/bibliography | part/bibliography | colophon
05226	set | book | part | preface | chapter | appendix | article | topic | reference | refentry | sect1 | sect2 | sect3 | sect4 | sect5 | section | book/glossary | article/glossary | part/glossary | book/bibliography | article/bibliography | part/bibliography | colophon
03325	set | book | part | reference | preface | chapter | appendix | article | topic | glossary | bibliography | index | setindex | refentry | sect1 | sect2 | sect3 | sect4 | sect5 | section
03410	set | book | part | reference | preface | chapter | appendix | article | topic | glossary | bibliography | index | setindex | refentry | refsynopsisdiv | refsect1 | refsect2 | refsect3 | refsection | sect1 | sect2 | sect3 | sect4 | sect5 | section
03407	set | book | part | reference | preface | chapter | appendix | article | glossary | bibliography | index | setindex | refentry | sect1 | sect2 | sect3 | sect4 | sect5 | section
04245	set | book | part | reference | preface | chapter | appendix | article | glossary | bibliography | index | setindex | refentry | refsynopsisdiv | refsect1 | refsect2 | refsect3 | refsection | sect1 | sect2 | sect3 | sect4 | sect5 | section
04953	set | book | part | reference | preface | chapter | appendix | article | glossary | bibliography | index | setindex | topic | refentry | refsynopsisdiv | refsect1 | refsect2 | refsect3 | refsection | sect1 | sect2 | sect3 | sect4 | sect5 | section
07095	sf:stylesheet | sf:stylesheet-ref | sf:container-hint | sf:page-start | sf:br | sf:selection-start | sf:selection-end | sf:insertion-point | sf:ghost-text | sf:attachments
05463	table//footnote | informaltable//footnote
12960	tei:classSpec/tei:attList//tei:attDef/tei:datatype/rng:ref

12961	tei:classSpec/tei:attList//tei:attDef/tei:datatype/tei:dataRef
09123	tei:content//rng:ref[@name = 'macro.anyXML']
12514	tei:content/tei:classRef | tei:content//tei:sequence/tei:classRef
12964	tei:dataSpec/tei:content//tei:dataRef
08632	tei:front//tei:titlePart/tei:title
10595	tei:label | tei:figure | tei:table | tei:item | tei:p | tei:title | tei:bibl | tei:anchor | tei:cell | tei:lg | tei:list | tei:sp
12962	tei:macroSpec/tei:content//rng:ref