Complete Subhedge Projection for Stepwise Hedge Automata

Antonio Al Serhali, Joachim Niehren

Article

Complete Subhedge Projection for Stepwise Hedge Automata

Antonio Al Serhali and Joachim Niehren

Inria Center of the University of Lille, France

Abstract: We show how to evaluate stepwise hedge automata (SHAs) with subhedge projection, while completely projecting irrelevant subhedges. Since this requires passing finite state information top-down, we introduce the notion of downward stepwise hedge automata. We use them to define in-memory and streaming evaluators with complete subhedge projection for SHAs. We then tune the evaluators so that they can decide membership at the earliest time point. We apply our algorithms to the problem of answering regular XPath queries on XML streams. Our experiments show that subhedge projection of SHAs can indeed speed up earliest query answering on XML streams.

Contents

Citation: Al Serhali, A.; Niehren, J. Complete Subhedge Projection. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Hedges are sequences of letters from some alphabet and trees $\langle h \rangle$ where h is a again a hedge. Hedges abstract from the syntactical details of XML or JSON documents while still being able to represent them. The linearization of a hedge is a nested word that can be written to a file. In this article, we study the problem of hedge pattern matching, so whether a given hedge h belong to a given regular hedge language L. Regular hedge languages can be defined in XPath or JSONPath in particular.

Projection is necessary for the efficiency of many algorithms on words, trees, hedges, or nested words, since it avoids to inspect irrelevant parts of the input structure. The relevance of projection for XML processing was already noticed by [1–4]. Saxon's in-memory evaluator, for instance, projects input XML document relative to an XSLT program, which contains a collection of XPath queries to be answered simultaneously [5]. The QuiXPath tool [4] evaluates XPath queries in streaming mode with subtree and descendant projection. Projection during the evaluation JSONPath queries on JSON documents in streaming mode is called fast-forwarding [6].

Projecting in-memory evaluation assumes that the full graph of the input hedge is constructed at beforehand. Nevertheless, projection may still save time, if one has to match several patterns on the same input hedge, or, if the graph was constructed for different reasons anyway. In streaming mode, the situation is similar: the whole input hedge on the stream needs to be parsed, but the evaluators needs to inspect only nodes that are not projected away. Given that pure parsing is by two or three orders of magnitude faster then pattern evaluation, the time gain of projection may be considerable.

The starting point of this article is the notion of subhedge irrelevance of positions of a hedge h with respect to a hedge language L that we introduce. These are positions of h where for all possible continuations inserting any subhedge does not affect membership to L. We contribute an algorithm for hedge pattern matching with complete subhedge projection. Our algorithm decides hedge pattern matching while completely projecting away the subhedges that are located at subhedge irrelevant positions. In other words, it decides whether a hedge h matches a pattern L without visiting any subhedge of h that is located at some position that is subhedge irrelevant with respect to L.

Regular hedge languages can be represented by nested regular expressions, which can be derived from regular XPath queries [7], or else by some kind of hedge automata, which can be compiled from nested regular expressions. We use stepwise hedge automata (SHAs) [8], a recent variant of standard hedge automata, which in turn go back to the sixties

[9,10]. SHAs mix up bottom-up processing of standard tree automata with the left-to-right processing of finite word automata (DFAs). They neither support top-down processing nor have an explicit stack, in contrast to nested word automata NWAs [11–15]. SHAs have a good notion of bottom-up and left-to-right determinism, avoiding the problematic notion of determinism for standard hedge automata, and the too costly notion of determinism for NWAs. Furthermore, compilers from nested regular expressions to SHAs are available [8], and determinization algorithms [16] producing small SHAs for all regular XPath queries in the existing practical benchmarks [7,17].

The motivation of the present work is to add subhedge projection to a recent streaming algorithm for deterministic SHAs that can evaluate regular XPath queries in an earliest manner based. Most alternative streaming approaches avoid earliest query answering all over, by considering sublanguages of regular queries without delays, so that node selection depends only on the past of the node but not on the future [18,19], or by admitting late selection [20]. In particular, it could be shown recently that earliest query answering for regular monadic queries defined by deterministic SHAs [21] has a lower worst case complexity than for deterministic NWAs [22]. Thereby, earliest query answering for regular queries became feasible in practice for the first time. On the other hand side, it is still slower experimentally than the best existing non-earliest approaches for general regular monadic queries on XML streams (see [23] for a comparison) since the latter support projection. How projection could be done for SHAs has not been studied before the conference version of the present article [24].

Consider the example of the XPath filter `[self::list][child::item]` that is satisfied by an XML document if its root is an `list` element that has some `item` child. When evaluating this filter on an XML document, it is sufficient to inspect its root for having the label `list` and then all its children until some `item` is found. The subhedges of these children can be projected away. However, one must memoize whether the level of the current node is 0, 1, or greater. This level information can be naturally updated in a top-down manner. The evaluators of SHAs, however, operate bottom-up and left-to-right exclusively. Therefore, projecting evaluators for SHAs need to be based on more general machines. We propose *downward stepwise hedge automata* (SHA$^{\downarrow}$s), a variant of SHAs that supports top-down processing in addition. They are basically Neumann and Seidl's pushdown forest automata [25], except that they apply to unlabeled hedges instead of labeled forests. NWAs are known to operate similarly on nested words [26], while allowing for more general visible pushdowns. We then distinguish subhedge projection states for SHA$^{\downarrow}$s, and show how to use them in order to evaluate SHAs with subhedge projection both in-memory and in streaming mode.

As a first contribution, we present the safe-no-change compiler from SHAs to SHA$^{\downarrow}$s that can distinguish appropriate subhedge projection states. The idea is that the safe-no-change SHA$^{\downarrow}$ can distinguish contexts in which the states of the SHA will safely not change. For instance, the XPath filter `[self::list][child::item]` can be defined by the deterministic SHA in Fig. 1, which our compiler maps to the SHA$^{\downarrow}$ in Fig. 2. The context information made explicit is about the levels of the states. This permits us to distinguish a projection state Π taken from level 2 on, and in which subhedges can be ignored. We prove the soundness of our compiler based on a nontrivial invariant that we establish. We note that the proof required an adaptation of the original compiler from FCT [24]. We also present a counter example showing that the safe-no-change projection is not complete for projection of all irrelevant subhedges. It shows that a subhedge may be irrelevant even though its state may still change.

As a second contribution, we propose the congruence projection algorithm. It again compiles SHAs to SHA$^{\downarrow}$s but relies on the congruence relations on automata states. We then prove that congruence projection yield not only a sound algorithm, but this algorithm is also complete for subhedge projection, i.e., all irrelevant subhedges are evaluated into some subhedge projection state. Congruence projection propagates congruence relations top-down,

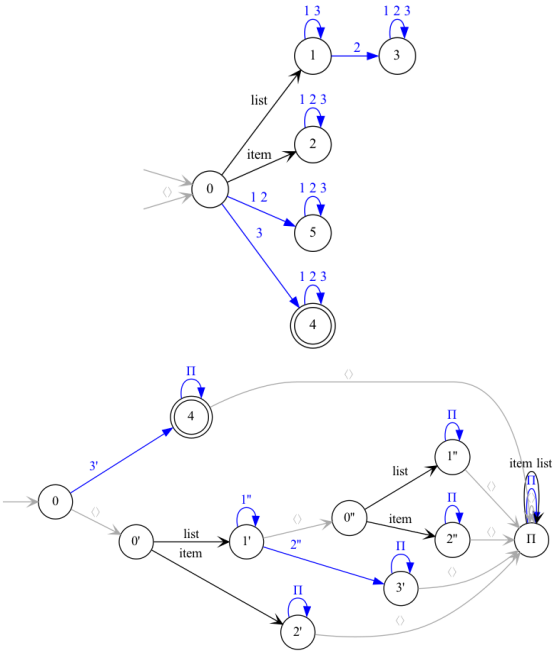


Figure 1. The unique minimal deterministic schema-complete SHA with initial state equal tree initial state for the XPath filter `[self::list][child::item]`.

Figure 2. The deterministic SHA$^{\downarrow}$ with subhedge projection state Π obtained by safe-no-change projection.

while starting with Myhill-Nerode's congruence known from automata minimization on top-level.

We show that both compilers may in the worst case increase the size of the automata exponentially. This is due to the fact that they must be able to convert deterministic automata on words that operate left to right into deterministic automata on words that operate right to left. This is since on hedge we need to turn bottom-up processing top-down. Due to the danger of an exponential explosion, we avoid constructing the SHA$^{\downarrow}$s statically in practice. Instead, we dynamically construct only the needed part of the SHA$^{\downarrow}$s on the fly when using it to evaluate some hedge with subhedge projection.

Our third contribution is a refinement of the two previous compilers, so that they can also distinguish safe states for selection at the earliest position. For this, we combine these compilers with a variant of the earliest membership tester of [21] that operates in-memory by compiling to SHA$^{\downarrow}$s instead of NWAs. Furthermore, membership failure is detected at the earliest position too. In this way, we obtain an earliest in-memory membership tester for deterministic SHAs.

Our fourth contribution is to show how to run the previous in-memory evaluators in streaming mode while reading the linearization of the hedge into a nested word as an input. Thereby, we can improve on the recent earliest streaming membership tester behind the AStream tool [21] by adding subhedge projection to it.

Our fifth and last contribution is an implementation and experimental evaluation of the streaming earliest query answering algorithm for dSHAs with safe-no-change subhedge projection, by introducing subhedge projection into the AStream tool [21]. For thism we have to lift our earliest membership tester with subhedge projection to an earliest query answering algorithm for monadic queries. For the experimentation, we start with the deterministic SHAs constructed with the compiler from [8] for the forward regular XPath queries of the XPathMark benchmark [27] and real-world XPath queries [7]. It turns out that we can reduce the running time for all regular XPath queries that contain only child axes considerably since large parts of the input hedges can be projected away. For such XPath queries, the earliest query answering algorithm of AStream with projection becomes competitive in efficiency with the best existing streaming algorithm from QuiXPath [23] (which is non-earliest on some queries though). The win is smaller for XPath queries with descendant axis, where only few subhedge projection is possible.

Outline. We start with preliminaries on hedges and nested words in Section 2. In Section 3, we recall the problem of hedge pattern matching and formally define irrelevant subhedges. In Section 4, we recall the definition of SHAs, introduce SHA$^{\downarrow}$s, discuss their in-memory evaluators, and define schema-completeness for them. In Section 5, we introduce the notion of subhedge projection states, define when an SHA$^{\downarrow}$ is complete for subhedge projection, and present an in-memory evaluator for SHA$^{\downarrow}$s with subhedge projection. In Section 6 we introduce safe-no-change projection and show its soundness and incompleteness. In Section 7, we introduce congruence projection, and prove it to be sound and complete for subhedge projection. Earliest in-memory evaluators for SHA$^{\downarrow}$s with subhedge projection follow in Section 8. Streaming variants of our in-memory evaluators are derived systematically in Section 9. Section 10 discusses how to lift our algorithms from membership testing to monadic queries. Section 11 discusses our practical experiments. Section 12 discusses further related work on automata notions related to SHAs and SHA$^{\downarrow}$s. Appendix O presents the reformulation of the earliest query answering algorithm from [21] based on SHA$^{\downarrow}$s and with schema restrictions.

Publication Comments. This original version of journal article was published at the FCT conference [24]. Compared to there, the following contributions are new:

- The definition of subhedge irrelevant prefixes of nested words and the definition of completeness for subhedge projection (Section 3).
- The addition of schema restrictions throughout the paper.
- The soundness proof of the safe-no-change projection (Section 6.2).
- The congruence projection-algorithm and the proof of its soundness and completeness for subhedge projection (Section 7).
- A systematic method to add subhedge projection to an earliest SHA$^{\downarrow}$ (Section 8).
- A systematic method to reduce the correctness of streaming automata evaluators to the corresponding in-memory evaluators (Section 9).
- A discussion of how to deal with monadic queries while exploiting the availability of schema restrictions (Section 10).
- A longer discussion on related work (Section 12).
- In Appendix O we also show how to obtain earliest dSHA$^{\downarrow}$s for dSHAs by adapting the compiler from dSHAs to dNWAs from [21] to dSHA$^{\downarrow}$s.

2. Preliminaries

Let A and B be sets and $r \subseteq A \times B$ a binary relation. The domain of r is $dom(r) = \{a \in A \mid \exists b \in B.\ (a,b) \in r\}$. We call r total if $dom(r) = A$. A partial function $f : A \hookrightarrow B$ is a relation $f \subseteq A \times B$ that is functional. A total function $f : A \to B$ is a partial function $f : A \hookrightarrow B$ that is total.

Words. Let $\mathbb{N}$ be the set of natural numbers including 0. Let the alphabet Σ be a set. The set of words over Σ is $\Sigma^* = \cup_{n\in\mathbb{N}}\Sigma^n$. A word $(a_1,\ldots,a_n) \in \Sigma^n$ where $n \in \mathbb{N}$ is written as $a_1 \ldots a_n$. We denote the empty word of length 0 by $\varepsilon \in \Sigma^0$ and by $v_1 \cdot v_2 \in \Sigma^*$ the concatenation of two words $v_1, v_2 \in \Sigma^*$. If $w = v \cdot v' \cdot v''$ is a word, then we call v a prefix of w and v' a factor of w and v'' a suffix of w.

If $\Sigma' \subseteq \Sigma$ then the projection of a word $v \in \Sigma^*$ to Σ' is the word $proj_{\Sigma'}(v)$ in $(\Sigma')^*$ that is obtained from v by removing all letters from $\Sigma \setminus \Sigma'$.

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

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

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 Σ^* as a hedge in $\mathcal{H}_\Sigma$, i.e., $\Sigma^* \ni aab = a \cdot a \cdot b \in \mathcal{H}_\Sigma$. Any hedge can be converted or stored as a graph. For the signature $\Sigma = \{list, item, A, \ldots, Z\}$, the graph of an example hedge in $\mathcal{H}_\Sigma$ is shown in Fig. 3.

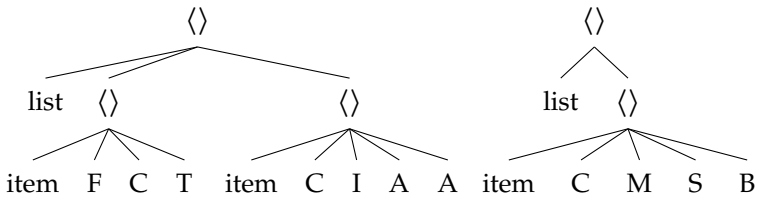


Figure 3. The graph of the hedge $\langle list \cdot \langle item \cdot F \cdot C \cdot T\rangle \cdot \langle item \cdot C \cdot I \cdot A \cdot A\rangle\rangle \cdot \langle list \cdot \langle item \cdot C \cdot M \cdot S \cdot B\rangle\rangle$ is a sequence of trees.

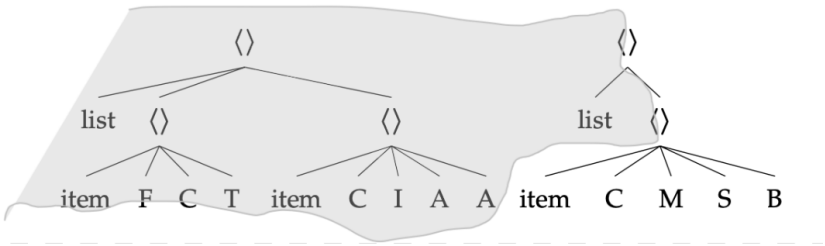


Figure 4. The part of the hedge distinguished by the nested word prefix $\langle list \cdot \langle item \cdot F \cdot C \cdot T\rangle \cdot \langle item \cdot C \cdot I \cdot A \cdot A\rangle\rangle \cdot \langle list \cdot \langle$.

Nested Words. A nested word over Σ is a word over the alphabet $\hat{\Sigma} = \Sigma \cup \{\langle, \rangle\}$ in which all parentheses are well-nested. Intuitively, this means that for any opening parenthesis there is a matching closing parenthesis and vice versa.

The set of nested words over Σ can be defined as the subsets of words over $\hat{\Sigma}$ that are a linearization of some hedge, where the linearization function $nw : \mathcal{H}_\Sigma \to (\Sigma \cup \{\langle, \rangle\})^*$ is defined as follows:

$$nw(\varepsilon) = \varepsilon, \quad nw(\langle h\rangle) = \langle \cdot nw(h) \cdot \rangle, \quad nw(a) = a, \quad nw(h \cdot h') = nw(h) \cdot nw(h').$$

So the set of all nested words over Σ is $nw(\mathcal{H}_\Sigma)$. Let hdg be the inverse of injective function nw on its image by hdg, so the mapping from nested words to hedges with $hdg(nw(h)) = h$ for all $h \in \mathcal{H}_\Sigma$.

Nested Word Prefixes. Prefixes, suffixes, and factors of nested words may not be nested words themselves. For instance, the hedge $h = a \cdot \langle b \cdot c\rangle$ has the linearization $nw(h) = a\langle bc\rangle$. Its prefix $a\langle b$ is not well-nested since it has a dangling opening parenthesis. Neither its suffix $c\rangle$ is well-nested since it has a dangling closing parenthesis.

Any algorithm that traverses a hedge h in a top-down and left-to-right manner inspects all the prefixes of $nw(h)$. Any prefix v of $nw(h)$ not only distinguishes some position of the hedge h but also specifies the part of hedge that is located before this position in the linearization $nw(h)$. An example is illustrated graphically in Fig. 4. This particularly holds for streaming algorithms that receive the nested word $nw(h)$ as an input on a stream, and may be inspected only once from the left to the right. But it holds equally for in-memory algorithms that receive the input hedge h as a hierarchical structure whose graph is stored in-memory, and then traverse the graph top-down and left-to-right. Any $\langle\rangle$ node will then be visited twice, once when reading an opening parenthesis $\langle$ and going down into a subhedge, and another time when going up to the closing parenthesis $\rangle$ after having processed the subhedge.

3. Problem

We introduce the hedge pattern matching problem that we want to solve and what parts of the hedges should not be visited.

3.1. Hedge Pattern Matching

We are interested in algorithms that match a hedge pattern against a hedge. We start with algorithms that test whether such a matching exists. This is sometimes called filtering, or alternatively Boolean query answering. In Section 10 we consider the more general problem to return the set of matchings too. This problem is sometimes called monadic query answering. Our experiments in Section 11 will apply to monadic query answering for regular XPath queries.

We start from hedge patterns that are nested regular expressions with letters in some signature Σ:

$$e, e' \in nRegExp_\Sigma ::= \varepsilon \mid a \mid e \cdot e' \mid e^* \mid \langle e \rangle \mid \mu a.e \qquad \text{where } a \in \Sigma$$

We adopt the usual restriction [28] that in any nested regular expression $\mu a.e$, all occurrences of a in e are guarded by some tree constructor $\langle . \rangle$. It is well known that such regular expressions can define all regular languages of hedges, i.e., they have the same expressiveness as hedge automata. Analogous results for tree regular expressions and tree automaton are folklore [29]. Semantically, any hedge pattern describes a hedge language $[\![e]\!] \subseteq \mathcal{H}_\Sigma$ that can be defined as usual. We may want to restrict the set of valid hedges by a schema $\mathbf{S} \subseteq \mathcal{H}_\Sigma$. For instance we might want to consider hedge encoding XML documents with two kinds of elements only, so $\Sigma = \{\text{list}, \text{item}\}$. An XML document then satisfies the following regular expression *doc*:

$$doc =_{\text{def}} tree^* \quad \text{where} \quad tree =_{\text{def}} \mu a.\, \langle (\text{list} + \text{item}) \cdot a^* \rangle$$

An XPath filter such as `[self::list/child::item]` can then be expressed by the nested regular expression:

$$filter =_{\text{def}} \langle \text{list} \cdot doc \cdot \langle \text{item} \cdot doc \rangle \cdot doc \rangle \cdot doc$$

General compilers from regular downward XPath filter queries to nested regular expressions where proposed in [8]. An implementation was developed in the context of the AStream tool [21].

The pattern matching problem for nested regular expressions with respect to a schema $\mathbf{S} \subseteq \mathcal{H}_\Sigma$ receives the following inputs:

- a nested regular expression $e \in nRegExp_\Sigma$ and
- a hedge $h \in \mathbf{S}$.

It then returns the truth value of the judgment $h \in [\![e]\!]$. The input hedge $h \in \mathbf{S}$ may either be given by a graph that resides in memory or by a stream containing the linearization $nw(h)$ of the input hedge and which can be read only once from left-to-right.

3.2. Irrelevant Subhedges

We define the concept of irrelevant occurrences of subhedges with respect to a given hedge pattern. What this means depends on the kind of algorithm that we will use for pattern matching. We will use algorithms that operate on the input hedge top-down, left-to-right, and bottom-up. This holds for streaming algorithms in particular.

Intuitively, when the pattern matching algorithm reaches a node top-down or left-to-right whose subsequent subhedge is irrelevant, then it can jump over it without inspecting its structure. What jumping means should be clear if the hedge is given by a graph that is stored in memory. Notice, that the full graph is to be constructed at before hand even though some parts of it may turn out irrelevant. Still one may win lots of time by jumping over irrelevant parts if either the graph was already constructed for other reason, or if many pattern matching problems are to be solved on the same hedge.

In the case of streams, the irrelevant subhedge needs still to be parsed, but it will not be analyzed otherwise. Most typically, the possible analysis are done by automata, that

may take 2 order of magnitudes more time than needed for the parsing. So therefore not having to do any analysis may considerably speed up a streaming algorithm.

Definition 1. *Let $\mathbf{S} \subseteq \mathcal{H}_\Sigma$ be a hedge schema and $L \subseteq \mathbf{S}$ a hedge language satisfying this schema. A nested word prefix v is called* subhedge irrelevant *for L with schema $\mathbf{S}$ if for all hedges $h \in \mathcal{H}_\Sigma$ and for all suffixes w of nested words such that $v \cdot w \in nw(\mathbf{S})$ and $v \cdot nw(h) \cdot w \in nw(\mathbf{S})$:*

$$v \cdot w \in nw(L) \Leftrightarrow v \cdot nw(h) \cdot w \in nw(L)$$

So language membership does not depend on hedge insertion at prefix v under the condition that schema membership is guaranteed. This definition generalizes over the case without schema restrictions where $\mathbf{S} = \mathcal{H}_\Sigma$. It is not clear, however, how one could reduce the general case to the schema-free case. Since we need schemas in our application to XPath, we thus have to deal with the general case.

Reconsider the schema $\mathbf{S} = [\![doc]\!]$ for hedges representing XML documents with signature $\Sigma = \{\text{list}, \text{item}\}$, and the regular hedge language $L = [\![filter]\!]$ where *filter* is the nested regular expression for the XPath filter `[self::list][child::item]`. Recall that this filter can be applied to hedges representing XML documents only. The nested word prefix $v = \langle\text{list}\langle\text{item}$ is subhedge irrelevant for the language L with schema $\mathbf{S}$. Note that its continuation $\langle\text{list}\langle\text{item}\rangle\rangle$ with the suffix $w = \rangle\rangle$ belongs to L. Hence, for any $h \in \mathcal{H}_\Sigma$ if the continutation $\langle\text{list}\langle\text{item} \cdot h\rangle\rangle$ belongs to $\mathbf{S} = [\![doc]\!]$ then it also belongs to L. Nevertheless, for the hedge $h_1 = \langle\rangle$ the continuation $\langle\text{list}\langle\text{item} \cdot h_1\rangle\rangle$ does not belong to L, since it does not satisfy the schema $\mathbf{S} = [\![doc]\!]$.

The prefix $\langle\text{item}$ is also irrelevant for the language L even independently of the schema. This is since L does not contain any continuation of this prefix to some hedge. The prefix $\langle\text{list}$ is not irrelevant for the language L wrt $\mathbf{S}$. This can be seen with suffix $w = \rangle$, since $\langle\text{list}\rangle$ does not belong to L while the continuation $\langle\text{list} \cdot h\rangle$ with $h = \langle\text{item}\rangle$ does belong to L and both continuations satisfy the schema $\mathbf{S} = [\![doc]\!]$.

4. Hedge Automata

We recall the notion of stepwise hedge automata (SHAs), introduce their downward extension (SHA$^\downarrow$), discuss schema-completeness for both kinds of automata, define subhedge projection states for SHA$^\downarrow$s, and show how to use them for evaluating SHA$^\downarrow$s with subhedge projection. We limit ourselves to in-memory evaluation in this section, but will discuss streaming evaluation in Section 9. The relation to the many closely related notions of automata on hedges forests or nested words are discussed in some more detail in Section 12.

4.1. Stepwise Hedge Automata

SHAs are a recent notion of automata for hedges [8] that mix up bottom-up tree and left-to-right word automata in a natural manner. They extend on stepwise tree automata [30] in that they operate on hedges rather than unranked trees, i.e., on sequences of letters and trees containing hedges. They differ from nested word automata (NWAs) [11,15] in that they process hedges directly rather than their linearizations to nested words.

Definition 2. *A stepwise hedge automaton (*SHA*) is a tuple $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ where Σ and $\mathcal{Q}$ are finite sets, $I, F \subseteq \mathcal{Q}$, and $\Delta = ((a^\Delta)_{a\in\Sigma}, \langle\rangle^\Delta, @^\Delta)$ where: $a^\Delta \subseteq \mathcal{Q} \times \mathcal{Q}$, $\langle\rangle^\Delta \subseteq \mathcal{Q}$, and $@^\Delta \subseteq (\mathcal{Q} \times \mathcal{Q}) \times \mathcal{Q}$. A* SHA *is deterministic or equivalently a d*SHA *if I and $\langle\rangle^\Delta$ contain at most one element, and all relations $(a^\Delta)_{a\in\Sigma}$ and $@^\Delta$ are partial functions.*

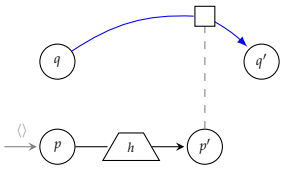
The set of state $\mathcal{Q}$ subsumes a subset I of initial state, a subset F of final states, and a subset $\langle\rangle^\Delta$ of tree initial states. The transition rules in Δ have three forms: If $(q, q') \in a^\Delta$ then we have a letter rule that we write as $q \xrightarrow{a} q'$ in Δ. If $(q, p, q') \in @^\Delta$ then we have an apply rule that we write as: $q@p \to q'$ in Δ. And if $q \in \langle\rangle^\Delta \in \mathcal{Q}$ then we have a tree initial

rule that we denote as $\xrightarrow{\langle\rangle} q$ in Δ. For any hedge $h \in \mathcal{H}_\Sigma$ we define the transition steps $q \xrightarrow{h} p$ wrt Δ such that for all $q, q', p, p' \in Q$, $a \in \Sigma$, and $h, h' \in \mathcal{H}_\Sigma$:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a} q' \text{ wrt } \Delta} \qquad \frac{q \xrightarrow{h} q' \text{ wrt } \Delta \quad q' \xrightarrow{h'} q'' \text{ wrt } \Delta}{q \xrightarrow{h \cdot h'} q'' \text{ wrt } \Delta}$$

$$\frac{true}{q \xrightarrow{\varepsilon} q \text{ wrt } \Delta} \qquad \frac{\xrightarrow{\langle\rangle} p \text{ in } \Delta \quad p \xrightarrow{h} p' \quad q@p' \to q' \text{ in } \Delta}{q \xrightarrow{\langle h \rangle} q' \text{ wrt } \Delta}$$

The transitions can be used to evaluate hedges nondeterministically bottom-up and left-to-right by SHAs. The first three rules state how to evaluate sequences of trees and letters such as a usual finite state automaton, while assuming that the trees were already evaluated to states. The last rule states how to evaluate a tree $\langle h \rangle$ from some state q to some state q'. This can be visualized as follows:



For any tree initial state $p \in \langle\rangle^\Delta$, one has to evaluate the subhedge h to some state p' nondeterministically. For each p' obtained, one then returns some state q' such that $q@p' \to q'$ in Δ nondeterministically.

A hedge is accepted by A if it permits a transition from some initial state to some final state. The language $Ł(A)$ is the set of all accepted hedges:

$$Ł(A) = \{h \in \mathcal{H}_\Sigma \mid q \xrightarrow{h} q' \text{ wrt } \Delta,\ q \in I,\ q' \in F\}$$

Runs can represent the whole history of a single choice of the nondeterministic evaluator on a hedge. For instance, the unique successful run of the dSHA in Fig. 1 on the hedge $h = \langle h' \rangle$ with subhedge $h' = list \cdot \langle item \rangle$ is the hedge $0 \cdot \langle 0 \cdot list \cdot 1 \cdot \langle 0 \cdot item \cdot 2 \rangle \cdot 3 \rangle \cdot 4$ whose computation is illustrated in Fig. 2. On the top level, the run justifies the transition $0 \xrightarrow{h} 4$ wrt Δ from the initial state 0 to a final state 4. When started in state 0, the subhedge h' needs to be evaluated first, so the run on the upper level needs to suspend until this is done. The subrun on h' eventually justifies the transition $0 \xrightarrow{h'} 3$ wrt Δ. At this time point the run of the upper level can be resumed and continue in state 4 since $0@3 \to 4$ in Δ.

When drawing runs, we illustrate any suspension/resumption mechanism by a box, as for instance by the box in the edge $0 \text{ —}\square\text{→ } 4$. The box stands for a future value computed by the lower level for which arrival the upper level suspends. Eventually, the box is filled by state 3 as illustrated by $3 \text{ - - }\square$, so that the computation can be resumed on the upper level.

We next define runs of SHAs on hedges formally. Whether a hedge with letters and states $R \in \mathcal{H}_{\Sigma \cup Q}$ is a run on a hedge $h \in \mathcal{H}_\Sigma$ – written $R \in run^\Delta(h)$ – is defined by the following rules:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \cdot a \cdot q' \in run^\Delta(a)} \qquad \frac{q \cdot R \cdot q' \in run^\Delta(h) \quad q' \cdot R' \cdot q'' \in run^\Delta(h')}{q \cdot R \cdot q' \cdot R' \cdot q'' \in run^\Delta(h \cdot h')}$$

$$\frac{true}{q \in run^\Delta(\varepsilon)} \qquad \frac{\xrightarrow{\langle\rangle} p \text{ in } \Delta \quad p \cdot R \cdot p' \in run^\Delta(h) \quad q@p' \to q' \text{ in } \Delta}{q \cdot \langle R \rangle \cdot q' \in run^\Delta(\langle h \rangle)}$$

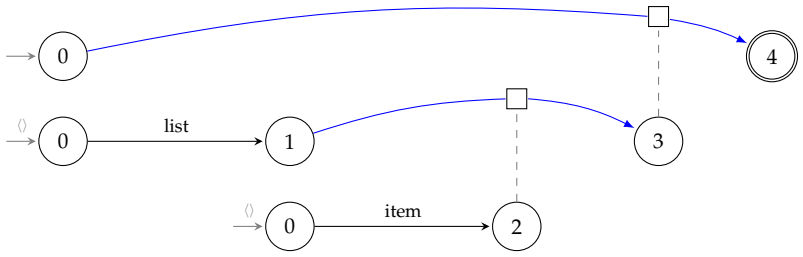


Figure 5. The unique successful run of the dSHA in Fig. 1 on the hedge $\langle list \cdot \langle item \rangle \rangle$ is the hedge $0 \cdot \langle 0 \cdot list \cdot 1 \cdot \langle 0 \cdot item \cdot 2 \rangle \cdot 3 \rangle \cdot 4$ computed as illustrated above.

Note that if $R \in run^{\Delta}(h)$ then h can be obtained by removing all states from R, i.e., $proj_{\Sigma}(R) = h$. Any run $R \in run^{\Delta}(h)$ can be identified with a mapping of prefixes of the nested word $nw(h)$ to states in $\mathcal{Q}$. Note that $nw(h) = proj_{\Sigma}(nw(R))$. For any prefix v of $nw(h)$ therefore there exists a unique prefix r of the nested word $nw(R)$ that ends with some state $q \in \mathcal{Q}$ and satisfies $proj_{\Sigma}(r) = v$. We then call q the state of R at prefix v. The run $0 \cdot \langle 0 \cdot list \cdot 1 \cdot \langle 0 \cdot item \cdot 2 \rangle \cdot 3 \rangle \cdot 4$ on hedge $\langle list \langle item \rangle \rangle$ from Fig. 5, for instance, assigns state 0 to the nested word prefixes ε and $\langle$ and state 1 to the nested word prefix $\langle \cdot$ list, etc.

A run is called a run of A if it starts with some state in I. It is called successful if it ends with some state in F.

Lemma 3. *$h \in Ł(A)$ iff there exists a successful run $R \in run^{\Delta}(h)$ for A.*

Proof. Straightforward by induction on h. □

Testing membership $h \in Ł(A)$ for a hedge h can be done in by computing the unique run of the determinization of A a bottom-up and left-to-right manner, and testing whether it is successful. Alternatively, one can compute the unique set $P \subseteq \mathcal{Q}$ such that $I \xrightarrow{h} P$ with respect to the determinization of A and test whether $P \cap F \neq \emptyset$. The computations are quasi the same, but the full history of the computation is lost when returning the transition only and not the run.

For determinizing SHAs we can rely on a variant of the usual subset construction that is well-known from finite state automata on words or standard tree automata (see e.g. [8,16]). When it comes to membership testing, on-the-fly determinization is sufficient, which creates only the part of the deterministic automaton that is used by the run on h. This part is of linear size $O(|h|)$ while the full determinization of A may be of exponential size $O(2^{|A|})$. In this way, membership $h \in Ł(A)$ can be tested in time $O(|h||A|)$.

For any set $P \subseteq \mathcal{Q}$ we define the set of states accessible from P via some hedge by:

$$acc^{\Delta}(P) = \{q' \in \mathcal{Q} \mid \exists\, q \in P.\ \exists h \in \mathcal{H}_{\Sigma}.\ q \xrightarrow{h} q' \text{ wrt } \Delta\}$$

We note that $acc^{\Delta}(P)$ can be computed in linear time in the size of Δ. This is since a state is hedge accessbile from P if and only if it is accessible for P in the graph of Δ. Clearly $Ł(A) = \emptyset$ if and only if $F \cap acc^{\Delta}(I) = \emptyset$. Therefore, emptiness can be decided in linear time for SHAs. This is in contrast to the more general notion of SHA$^{\downarrow}$s that we introduce next.

4.2. Downward Stepwise Hedge Automata

The evaluation of SHAs operates in a bottom-up and left-to-right manner, but cannot pass any information top-down. We next propose an extension of SHAs to SHA$^{\downarrow}$s, adding the ability to pass finite state information top-down.

SHA$^{\downarrow}$s are basically equal to Neumann and Seidl's pushdown forest automata [31] but lifted from (labeled) forests to (unlabeled) hedges. See Section 12 for a more detailed discussion on the relationship of SHA$^{\downarrow}$s to other automata notions and NWAs in particular.

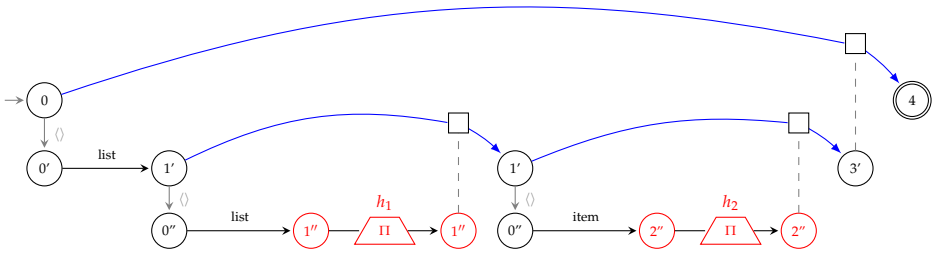


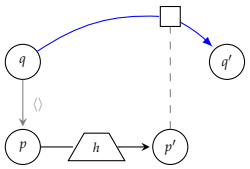
Figure 6. A successful run of the SHA$^{\downarrow}$ in Fig. 2 on the hedge $\langle list \cdot \langle \text{list} \cdot h_1 \rangle \cdot \langle \text{item} \cdot h_2 \rangle \rangle$.

Definition 4. *A downward stepwise hedge automaton (*SHA$^{\downarrow}$*) is a tuple* $A = (\Sigma, Q, \Delta, I, F)$ *where* Σ *and* Q *are finite sets,* $I, F \subseteq Q$*, and* $\Delta = ((a^{\Delta})_{a\in\Sigma}, \langle\rangle^{\Delta}, @^{\Delta})$*. Furthermore,* $a^{\Delta} \subseteq Q \times Q$*,* $\langle\rangle^{\Delta} \subseteq Q \times Q$*, and* $@^{\Delta} \subseteq (Q \times Q) \times Q$*. A* SHA$^{\downarrow}$ *is called deterministic or equivalently a d*SHA$^{\downarrow}$ *if* I *contains at most one element, and all relations* $\langle\rangle^{\Delta}$*,* a^{Δ}*, and* $@^{\Delta}$ *are partial functions.*

The only difference to SHAs is the form of the tree opening rules. If $(q, q') \in \langle\rangle^{\Delta} \in Q$ then we have a tree initial rule that we denote as: $q \xrightarrow{\langle\rangle} q'$ in Δ. So here the state q' where the evaluation of a subhedge starts depends on the state q of the parent. The definition of transition steps for SHA$^{\downarrow}$s differs from that of SHAs in the following rule:

$$\frac{q \xrightarrow{\langle\rangle} p \text{ in } \Delta \quad p \xrightarrow{h} p' \text{ wrt } \Delta \quad q@p' \to q' \text{ in } \Delta}{q \xrightarrow{\langle h\rangle} q' \text{ wrt } \Delta}$$

This means that the evaluation of the subhedge h starts in some state p such that $q \xrightarrow{\langle\rangle} p$ in Δ.



So the restart state p that is chosen may now depend on the state q above. This is how finite state information is passed top-down by SHA$^{\downarrow}$s. SHAs in contrast operate purely bottom-up and left-to-right.

The notion of runs can be adapted straightforwardly from SHAs to SHA$^{\downarrow}$s. When in state q, it is sufficient to restart the computation in subhedges on the state p such that $q \xrightarrow{\langle\rangle} p \in \Delta$. In this way, finite state information is passed down (while for SHAs some tree initial is to be chosen that is independent of q). The only rule of runs to be changed is the following:

$$\frac{q \xrightarrow{\langle\rangle} p \text{ in } \Delta \quad p \cdot R \cdot p' \in run^{\Delta}(h) \quad q@p' \to q' \text{ in } \Delta}{q \cdot \langle R\rangle \cdot q' \in run^{\Delta}(\langle h\rangle)}$$

An example of run of the *d*SHA$^{\downarrow}$ in Fig. 2 is shown in Fig. 6. Here, the information on the level of nodes is passed down. It is represented by the number of primes. For instance, when opening the first subtree labeled with `item` we use the transition rule $1' \xrightarrow{\langle\rangle} 0''$ stating that one moves from state 1 on the first level to state 0 on the second level.

One can see that all nodes of the subtrees h_1 and h_2 are evaluated to the projection state Π. So one can jump over these subtrees without reading them. This is justified by the fact that they are on level 2, the information that the evaluator of this SHA$^{\downarrow}$ was passing down.

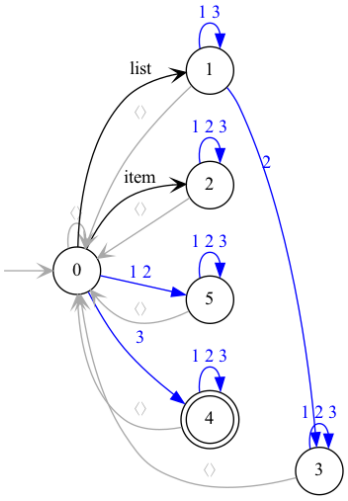


Figure 7. The SHA$^{\downarrow}$ A^{down} for the dSHA A for the filter `[self::list][child::item]` in Fig. 1.

Testing membership $h \in Ł(A)$ for a SHA$^{\downarrow}$s in memory can again be done by computing the unique run of the determinization of A. As before, the membership tester can be based on on-the-fly determinization. However, the determinization procedure for SHA$^{\downarrow}$s is more complicated then for SHAs, since SHA$^{\downarrow}$s can pass information top-down and not only bottom-up and left-to-right. Determinization therefore does not only rely on the pure subset construction, but also uses pairs of states in the subsets, basically in the same way as for nested word automata NWAs [14,15]. This is needed to deal with the stack of states that were seen above. Therefore, each determinization step may cost time $O(|A|^5)$ as stated for instance in [32]. The upper time bound for membership testing for SHA$^{\downarrow}$s is thus in time $O(|h||A|^5)$, and no more combined linear as for SHAs.

Any SHA $\mathcal{A} = (\Sigma, Q, \Delta, I, F)$ can be mapped to a SHA$^{\downarrow}$ $A^{down} = (\Sigma, Q, \Delta^{down}, I, F)$ while preserving the runs and determinism. The only change of Δ^{down} compared to Δ is described by the following rule:

$$\frac{\xrightarrow{\langle\rangle} p \text{ in } \Delta \quad q \in Q}{q \xrightarrow{\langle\rangle} p \text{ in } \Delta^{down}}$$

Independent of the current state $q \in Q$, the SHA$^{\downarrow}$ A^{down} can start the evaluation of the subhedge of a subtree in any open tree state $p \in \langle\rangle^{\Delta}$. For instance, the dSHA$^{\downarrow}$ A^{down} for dSHA A from Fig. 1 from the introduction is given in Fig. 7. Conversely, one can convert any dSHA$^{\downarrow}$ A into an equivalent SHA by introducing nondeterminism to eliminate top-down processing. See Section 12.2 for the construction.

Hedge accessibility can be defined for SHA$^{\downarrow}$s literally as for SHAs. However, the set $acc^{\Delta}(P)$ can no more be computed in linear time in the size of Δ – in contrast to SHAs. It can still be done in cubic time, similarly as for NWAs [22]. The problem is that hedge accessibility for SHA$^{\downarrow}$s does no more coincide with the accessibility notion of the automaton's graph.

For instance, in the SHA$^{\downarrow}$ from Fig. 2, we have $acc^{\Delta}(\{0\}) = \{4\}$. Note that $0'$ does not belong to this set, even though there is a transition rule $0 \xrightarrow{\langle\rangle} 0'$ in Δ, making node $0'$ accessible from node 0 in the graph of the automaton. This is since, $0'$ is not accessible over any hedge from 0, i.e., there does not exist any hedge h such that $0 \xrightarrow{h} 0'$ wrt Δ. Note that $0 \cdot \langle \cdot 0'$ is a factor of the nested word of some run of Δ, show that $0'$ is accessible from 0 over a nested word prefix nevertheless. Due to the cubic time of hedge accessibility for SHA$^{\downarrow}$s, we will use hedge accessibility only for SHAs.

4.3. Schema Completeness

Schemas are inputs of the membership problem that we consider, while completeness is a fundamental property of automata. In order to combine both aspects in an appropriate manner, we propose the notion of schema completeness, that we define uniformly for both SHAs and SHA$^{\downarrow}$s.

We call a set Δ of transition rules of a SHA$^{\downarrow}$ complete if all its relations $(a^{\Delta})_{a\in\Sigma}$, $\langle\rangle^{\Delta}$ and $@^{\Delta}$ are total. For SHA*s* we require that $\langle\rangle^{\Delta}$ is a nonempty set (instead of a total relation). A SHA or SHA$^{\downarrow}$ A is called complete if Δ is complete and $I \neq \emptyset$. We can complete any SHA or SHA$^{\downarrow}$ by adding a fresh sink state and also transition rules into the sink state for all left-hand side, for which no other transition rule exists. The sink state is made initial if and only if there was no initial state before. It is not final though. We denote the completion of A by $compl(A)$.

Definition 5. *A* partial run *r of A on a hedge h is a prefix of the some nested of some run of $compl(A)$ on h such that:*

- *r ends with some state of Q, and*
- *r does not contain the sink state of $compl(A)$.*

A partial run r of A on h is called blocking *if there does not exist any partial run r' of A on h such than r is a strict prefix of r'.*

For instance consider the hedge $h = \langle \text{list} \cdot \langle \text{item} \rangle \rangle$. The dSHA in Fig. 1 has the unique run $R = 0 \cdot \langle \cdot 0 \cdot \mathit{list} \cdot 1 \cdot \langle \cdot 0 \cdot \mathit{item} \cdot 2\rangle \cdot 3 \cdot \rangle \cdot 4$ on h that is illustrated in Fig. 5. The partial runs of h are thus all prefixes of $nw(R)$ that end with some state, i.e., $0, 0 \cdot \langle \cdot 0, 0 \cdot \langle \cdot 0 \cdot \text{list} \cdot 1$, etc. None of these partial runs is blocking.

Definition 6. *A schema for an automaton A is a hedge language $\mathbf{S} \subseteq \mathcal{H}_{\Sigma}$ such that $Ł(A) \subseteq \mathbf{S}$. We call A schema-complete for schema $\mathbf{S}$ if no hedge $h \in \mathbf{S}$ has some blocking partial run of A.*

Schemas are usually used to restrict the type of hedges that some automaton problem may accept as input. If A is schema-complete for the input schema $\mathbf{S}$ then no partial run of A on any input hedge from $\mathbf{S}$ may ever block.

The dSHA in Fig. 1 with signature $\Sigma = \{\text{item}, \text{list}\}$ is schema-complete for schema $[\![doc]\!]$. For making this happen, we added the state 5 to this automaton even though it cannot be used in any successful run. But still, this SHA is not complete. For completing it, we have to add some sink *sink* and many transition into it, such as $0@4 \to \mathit{sink}$ and $0@5 \to \mathit{sink}$, but also $3 \xrightarrow{\text{item}} \mathit{sink}$, etc. These transitions, however, are useless for evaluating hedges inside the schema $[\![doc]\!]$ (and would make subhedge projection with the safe-no-change algorithm fail on this example).

We notice that schemas may be a nonregular hedge languages. Schema-completeness for nonregular schemas still make good sense. Furthermore, if A is schema-complete for the universal schema $\mathbf{S} = \mathcal{H}_{\Sigma}$ and does not have inaccessible states then A is complete.

We call *A compatible with a schema $\mathbf{S}$* if for any hedge $h \in \mathbf{S}$ there exists a run of A in $run^{\Delta}(h)$. It is instructive to see that schema-completeness implies compatibility. In contrast, compatibility with a schema does not exclude the existence of blocking partial runs in the presence of nondeterminism.

To see this, suppose that A is schema-complete for $\mathbf{S}$. Wlog., we can assume $\mathbf{S} \neq \emptyset$. Note that if $I = \emptyset$, then ε would be a blocking partial run for any hedge in $\mathbf{S} \setminus \{\varepsilon\}$. Since $\mathbf{S} \neq \emptyset$ it follows that there exists $q_0 \in I$. So q_0 is a partial run for any hedge $h \in \mathbf{S} \setminus \{\varepsilon\}$. Since there exists no blocking partial runs by schema-completeness, this run can be extended step by step to a run on any $h \in \mathbf{S}$. So for any hedge $h \in \mathbf{S}$ there exists some run by A, showing compatibility.

Lemma 7. *If A is deterministic, then A is compatible with $\mathbf{S}$ iff A is schema-complete for $\mathbf{S}$.*

Proof. Let A be compatible with $\mathbf{S}$. We have to show that A is schema-complete for $\mathbf{S}$. Let v be a partial run of A on $h \in \mathbf{S}$. By compatibility, there exists a run $R \in run^{\Delta}(h)$ that starts in some initial state of A. By determinism, v must be a prefix of $nw(R)$. Thus, v is not blocking. □

Let B be a dSHA and $\mathbf{S} = \text{Ł}(B)$. Any SHA A with schema $\mathbf{S}$ can be made schema-complete for $\mathbf{S}$ as follows. First we note that B has $\mathbf{S}$ as schema too, and that B is compatible with $\mathbf{S}$. Since B is deterministic, Lemma 7 implies that B is schema-complete for $\mathbf{S}$. We then compute the completion of $compl(A)$. Clearly, $compl(A)$ is schema-complete for schema $\mathcal{H}_\Sigma$. The product $C = B \times compl(A)$ is thus schema-complete for schema $\mathbf{S}$ too. With $F^C = F^B \times F^A$ we have $\text{Ł}(C) = \text{Ł}(B) \cap \text{Ł}(A) = \text{Ł}(B) = \mathbf{S}$. Finally, let $F^S = F^B \times \mathcal{Q}^A$. We then have $F^C = F^B \times F^A \subseteq F^{\mathbf{S}}$ and $\text{Ł}(C[F^C/F^{\mathbf{S}}]) = \mathbf{S}$.

5. Subhedge Projection

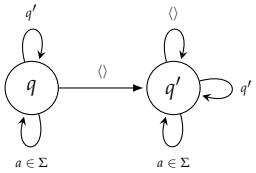
We next introduce the concept of subhedge projection states for SHA$^{\downarrow}$s in order to distinguish prefixes of hedges that are subhedge irrelevant, and to evaluate SHA$^{\downarrow}$s with subhedge projection.

5.1. Subhedge Projection States

We start with SHA$^{\downarrow}$s without any restrictions but will impose schema-completeness and determinism soon later on.

Definition 8. *We call a state $q \in \mathcal{Q}$ a* subhedge projection state of Δ *if there exists $q' \in \mathcal{Q}$ called the witness of q such that the set of transition rules of Δ containing q' or q on the leftmost position is included in the following set:*

$$\{q \xrightarrow{\langle\rangle} q',\ q@q' \to q,\ q' \xrightarrow{\langle\rangle} q',\ q'@q' \to q'\}$$
$$\cup\{q' \xrightarrow{a} q',\ q \xrightarrow{a} q \mid a \in \Sigma\}$$



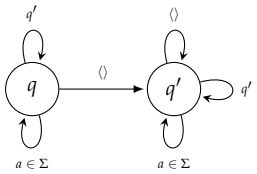
The set of all subhedge projection states of Δ is denoted by $\mathcal{Q}^{\Delta}_{shp}$.

For any complete SHA$^{\downarrow}$, the above set of transition rules must be equal to the set of all transition rules of Δ with q or q' on the leftmost position. But if the SHA$^{\downarrow}$ is only schema-complete for some schema $\mathbf{S}$ then not all these transitions must be present. Note also that a subhedge projection state q may be equal to its witness q'. Therefore any witness q' of some subhedge projection state is itself a subhedge projection state with witness q'.

In the example $d\text{SHA}^{\downarrow}$ $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ in Fig. 2, we have $\mathcal{Q}^{\Delta}_{shp} = \{\Pi, 4, 2', 3', 1'', 2''\}$. The witness of all these subhedge projection states is Π. Note that not all possible transitions are present for the states in $\mathcal{Q}^{\Delta}_{shp} \setminus \{\Pi\}$, given that this automaton is not complete (but still schema-complete for schema $[\![doc]\!]$).

Lemma 9. *If $q \in \mathcal{Q}^{\Delta}_{shp}$ is a subhedge projection state and $q \xrightarrow{h} p$ wrt Δ then $q = p$.*

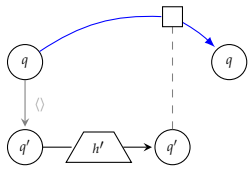
Proof. Suppose that $q \xrightarrow{h} p$ wrt Δ and that q is a subhedge projection state of Δ with witness q'. So the only possible transitions with q or q' on the left hand side are the following:



Suppose that $h = t_1 \cdot \ldots \cdot t_n$ is a sequence of trees or letters $t_i \in \langle \mathcal{H}_\Sigma \rangle \cup \Sigma$ where $1 \leq i \leq n$. Then there exists runs $R_i \in run^\Delta(t_i)$ and a run $R \in run^\Delta(h)$ that has the form $h = q_0 \cdot R_1 \cdot q_1 \cdot \ldots \cdot R_n \cdot q_n$ where $q_0 = q$ and $q_n = p$. We prove for all $0 \leq i \leq n$ that $q_i = q$ by induction on i. For $i = 0$, this is trivial. For $i > 0$, the induction hypothesis shows that $q_{i-1} = q$

Case $t_i = a \in \Sigma$**.** Then $q_{i-1} \xrightarrow{a} q_i$ in Δ Since $q_{i-1} = q$ is a subhedge projection state of Δ it follows that $q_i = q$ too.

Case $t_i = \langle h' \rangle$ **for some** $h' \in \mathcal{H}_\Sigma$**.** Since $q_{i-1} = q$ is a subhedge projection state of Δ with witness q', the subrun of R recognizing t_i must be justified by the following diagram:



So the subrun of R of h' may only contain the witness q' and furthermore, $q_i = q$.

□

5.2. Completeness for Subhedge Projection

Before defining when a SHA$^\downarrow$ is complete for subhedge projection, we show that subhedge projection states in runs of deterministic SHAs may only occur at irrelevant prefixes.

Proposition 10. *Let $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a dSHA$^\downarrow$ with schema $\mathbf{S}$, $R \in run^\Delta(h)$ the run of $compl(A)$ on some hedge $h \in \mathbf{S}$, and q a subhedge projection state for Δ. Then for any prefix $r \cdot q$ of $nw(R)$, the prefix $proj_\Sigma(r)$ of $nw(h)$ is subhedge irrelevant for $Ł(A)$ for schema $\mathbf{S}$.*

Proof. Let $R \in run^\Delta(h)$ be the unique run of $compl(A)$ on some hedge $h \in \mathbf{S}$. Suppose that $r \cdot q \cdot s = nw(R)$ for some subhedge projection state q of Δ and let $v = proj_\Sigma(r)$ and $w = proj_\Sigma(s)$. Since $v \cdot w = nw(h)$ and $h \in \mathbf{S}$ it follows that $v \cdot w \in nw(\mathbf{S})$. We fix some hedge $h' \in \mathcal{H}_\Sigma$ such that $v \cdot nw(h') \cdot w \in nw(\mathbf{S})$ arbitrarily. Let $\tilde{h} \in \mathcal{H}_\Sigma$ such that $nw(\tilde{h}) = v \cdot nw(h') \cdot w$. For the subhedge irrelevance of v, we have to show that:

$$h \in Ł(A) \Leftrightarrow \tilde{h} \in Ł(A)$$

"⇒" We suppose that $h \in Ł(A)$ and have to show that $\tilde{h} \in Ł(A)$. For the partial run $r \cdot q$ of A on $\tilde{h}$ there must exist R' and p such that $q \cdot R' \cdot p \in run^\Delta(h')$. Since q is a subhedge projection state of Δ, it follows by Lemma 9 that $q = p$. Hence $r \cdot q \cdot nw(R') \cdot q \cdot s$ is a run of A on $\tilde{h}$. Since R was successful for A, s must end in F, and thus the above run on $\tilde{h}$ is successful for A too. Hence $\tilde{h} \in Ł(A)$ as we had to show.

"⇐" We suppose that $\tilde{h} \in Ł(A)$ and have to show that $h \in Ł(A)$. Let $\tilde{R}$ be the unique run of $compl(A)$ on $\tilde{h}$. By determinism, $\tilde{R}$ must have the prefix $r \cdot q$. So there exist R', p', s such that $p \cdot R' \cdot q \in run^\Delta(h')$ and $r \cdot q \cdot R' \cdot p \cdot s = \tilde{R}$. Lemma 9 shows that $p = q$. Hence, $r \cdot q \cdot s$ is the unique run of $compl(A)$ on h and this run is accepting. So $h \in Ł(A)$.

□

We note that Proposition 10 would not hold without assuming determinism. To see this, we can add some sink to any dSHA as a second initial state. One can then always go to this sink, which is a subhedge projection state. Nevertheless, no prefix going to the sink may be subhedge irrelevant.

Proposition 10 shows that subhedge projection states permit to distinguish prefixes that are subhedge irrelevant for deterministic SHA$^{\downarrow}$s. An interesting question is whether all subhedge irrelevant prefixes can be found in this way.

Definition 11. *A dSHA$^{\downarrow}$ $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ that is schema-complete for $\mathbf{S}$ is called* complete for subhedge projection for schema $\mathbf{S}$ *if for all hedges $h \in \mathbf{S}$ and all prefixes v of $nw(h)$ that are subhedge irrelevant for Δ, the unique run of A on h goes to some subhedge projection state of Δ at v.*

The example dSHA$^{\downarrow}$ from Fig. 2 is indeed complete for subhedge projection with schema $[\![doc]\!]$. In general, however, this does not hold for all dSHA$^{\downarrow}$s obtained by safe-no-change projection as we will discuss in Section 6.3.

5.3. In-Memory Evaluator with Subhedge Projection

We next show how to refine the transitions for SHA$^{\downarrow}$s by subhedge projection. We define the transition relation with subhedge projection $\xrightarrow{h}_{shp} \subseteq \mathcal{Q} \times \mathcal{Q}$ with respect to Δ such that for all hedges $h, h' \in \mathcal{H}_\Sigma$ and letters $a \in \Sigma$:

$$\frac{q \in \mathcal{Q}^\Delta_{shp} \quad h \in \mathcal{H}_\Sigma}{q \xrightarrow{h}_{shp} q \text{ wrt } \Delta} \qquad \frac{q \notin \mathcal{Q}^\Delta_{shp} \quad q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a}_{shp} q' \text{ wrt } \Delta} \qquad \frac{q \notin \mathcal{Q}^\Delta_{shp}}{q \xrightarrow{\varepsilon}_{shp} q \text{ wrt } \Delta}$$

$$\frac{q \notin \mathcal{Q}^\Delta_{shp} \quad q \xrightarrow{h}_{shp} q' \text{ wrt } \Delta \quad q' \xrightarrow{h'}_{shp} q'' \text{ wrt } \Delta}{q \xrightarrow{h \cdot h'}_{shp} q'' \text{ wrt } \Delta}$$

$$\frac{q \notin \mathcal{Q}^\Delta_{shp} \quad q \xrightarrow{\langle\rangle} p \text{ in } \Delta \quad p \xrightarrow{h}_{shp} p' \quad q@p' \to q' \text{ in } \Delta}{q \xrightarrow{\langle h \rangle}_{shp} q' \text{ wrt } \Delta}$$

Subhedge projecting transitions stay in subhedge projection states until the end of the current subhedge is reached. This is correct by Lemma 9 under the condition that there are no blocking runs.

Proposition 12. *Let $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a* SHA$^{\downarrow}$ *that is schema-complete for $\mathbf{S}$. Then for all hedges $h \in \mathbf{S}$ and states $q, p \in \mathcal{Q}$:*

$$q \xrightarrow{h} p \text{ wrt } \Delta \text{ iff } q \xrightarrow{h}_{shp} p \text{ wrt } \Delta$$

Proof. We distinguish two cases:

Case $q \notin \mathcal{Q}^\Delta_{shp}$**.** Then for any $p \in \mathcal{Q}$, $q \xrightarrow{h} p$ wrt Δ iff $q \xrightarrow{h}_{shp} p$ wrt Δ by definition of the projecting transitions.

Case $q \in \mathcal{Q}^\Delta_{shp}$**.** Then $q \xrightarrow{h}_{shp} q$ wrt Δ. Since $h \in \mathbf{S}$ and A is schema-complete for $\mathbf{S}$ there exists some run of A on h, and thus some state q such that $q \xrightarrow{h} q'$ wrt Δ. Lemma 9 shows that $q' = q$ and thus $q \xrightarrow{h} q$ wrt Δ.

□

For evaluating a nondeterministic SHA$^{\downarrow}$ A on some hedge with subhedge projection, we again have to compute the transition of the determinization A^{det} on the hedge, where only the needed part of A^{det} is to be produced on-the-fly. But now when discovering a new

state P of $A^{\det}$, we have to test whether it is a subhedge projection state. For this we have to compute the state P' such that $P \xrightarrow{\langle\rangle} P'$ wrt $\Delta^{\det}$ and test whether only the transition rules allowed for subhedge projection states for P with witness P' are available wrt $\Delta^{\det}$.

6. Safe-No-Change Projection

We next introduce a compiler from SHAs to SHA$^{\downarrow}$s, that propagates information top-down by which to distinguish states that will safely no more be changed. Thereby, subhedge projection states are produced so that the evaluator for SHA$^{\downarrow}$s with subhedge projection from Section 5.3 can be applied. Note that our compiler works independently of any schema, while its soundness still requires the schema-completeness for some schema.

6.1. Algorithm

Let $A = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a SHA with schema $\mathbf{S}$. For any set $P \subseteq \mathcal{Q}$ we define the set

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

We note that $\mathit{safe}^{\Delta}(P)$ can be computed in linear time in the size of Δ, by using inverse hedge accessibility from P. We define the set of states that may no more change by:

$$\mathit{no\text{-}change}^{\Delta} = \{q \mid q \in \mathit{safe}^{\Delta}(\{q\})\}$$

Note that $q \in \mathit{no\text{-}change}^{\Delta}$ if and only if $\mathit{acc}^{\Delta}(\{q\}) \subseteq \{q\}$. In the example automaton from Fig. 1 we have $\mathit{no\text{-}change}^{\Delta} = \{2, 3, 4, 5\}$.

Lemma 13. *Let $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a SHA that is schema-complete for schema $\mathbf{S}$. For any hedge $h \in \mathbf{S}$ and state $q \in I \cap \mathit{no\text{-}change}^{\Delta}$ it then holds that $q \xrightarrow{h} q$ wrt Δ.*

Proof. The schema-completeness of A for $\mathbf{S}$ applied to $h \in \mathbf{S}$ and $q \in I$ yields the existence of some state $q' \in \mathcal{Q}$ such that $q \xrightarrow{h} q'$ wrt Δ. Let q' be any such state. Note that $q' \in \mathit{acc}^{\Delta}(q)$. Since $q \in \mathit{no\text{-}change}^{\Delta}$, we have $q \in \mathit{safe}^{\Delta}(\{q\})$ so that $q' \in \mathit{acc}^{\Delta}(q)$ implies $q' = q$. This proves $q \xrightarrow{h} q$ wrt Δ. □

For any state $q \in \mathcal{Q}$ and subset of states $Q \subseteq \mathcal{Q}$ we define:

$$\begin{aligned} \mathit{s\text{-}down}^{\Delta}(q, Q) &= \mathit{safe}^{\Delta}(\{p \in \mathcal{Q} \mid q@^{\Delta}p \subseteq Q\}) \\ \mathit{s\text{-}no\text{-}change}^{\Delta}(q) &= \mathit{s\text{-}down}^{\Delta}(q, \{q\}) \end{aligned}$$

A state belongs to $\mathit{s\text{-}down}^{\Delta}(q, Q)$ if all states $p \in \mathit{acc}^{\Delta}(q)$ satisfy $\emptyset \neq q@^{\Delta}p \subseteq Q$. So p is a state down that will safely go up to some state in Q.

We next compile the SHA A to a SHA$^{\downarrow}$ $A^{snc} = (\Sigma, \mathcal{Q}^{snc}, \Delta^{snc}, I^{snc}, F^{snc})$. For this let Π be a fresh symbol and consider the state set:

$$\mathcal{Q}^{snc} = \{\Pi\} \uplus (\mathcal{Q} \times 2^{\mathcal{Q}})$$

A pair (q, P) means that the evaluator in state q may project subhedges if $q \in P$ since these will no more lead to any relevant change. The sets of initial and final states are defined as follows:

$$I^{snc} = \{(q, \emptyset) \mid q \in I\} \qquad F^{snc} = \{(q, \emptyset) \mid q \in F\}$$

How to generate the transition rules of A^{snc} from those of A is described in Fig. 8.

When applied to the SHA in Fig. 1 for `[self::list][child::item]`, the construction yields the SHA$^{\downarrow}$ in Fig. 9 which is indeed equal to the SHA$^{\downarrow}$ from Fig. 2 up to state renaming. When run on the hedge $\langle \mathit{list} \cdot \langle \mathit{list} \cdot h_1 \rangle \cdot \langle \mathit{item} \cdot h_2 \rangle \rangle$ as shown in Fig. 6, it does not have to visit the subhedges h_1 nor h_2, since all of them will be reached starting from the projection state Π.

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{a} (q',P) \text{ in } \Delta^{snc}} \qquad \frac{a \in \Sigma \quad q \in P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{a} (q,P) \text{ in } \Delta^{snc}}$$

$$\frac{\xrightarrow{\langle\rangle} q' \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{\langle\rangle} (q', \textit{s-no-change}^{\Delta}(q)) \text{ in } \Delta^{snc}} \qquad \frac{q \in P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{snc}}$$

$$\frac{q@p \to q' \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P)@(p, \textit{s-no-change}^{\Delta}(q)) \to (q',P) \text{ in } \Delta^{snc}} \qquad \frac{q \in P \cup \textit{no-change}^{\Delta}}{(q,P)@\Pi \to (q,P) \text{ in } \Delta^{snc}}$$

$$\frac{a \in \Sigma}{\Pi \xrightarrow{a} \Pi \text{ in } \Delta^{snc}} \qquad \frac{\textit{true}}{\Pi@\Pi \to \Pi \text{ in } \Delta^{snc}} \qquad \frac{\textit{true}}{\Pi \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{snc}}$$

Figure 8. The transition rules of the SHA$^{\downarrow}$ A^{snc} inferred from those of the SHA A.

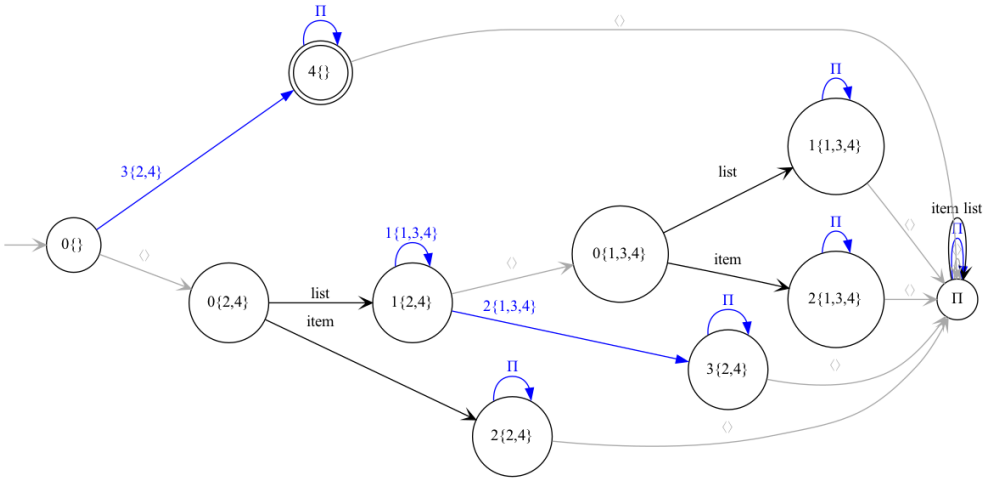


Figure 9. The dSHA$^{\downarrow}$ A^{snc} constructed from the dSHA A in Fig. 1 except for useless state transitions with respect to schema $[\![doc]\!]$. It is equal to the SHA$^{\downarrow}$ in Fig. 2 up to the state renaming $0 = (0,\{\})$, $0' = (0,\{2,4\})$, $0'' = (0,\{1,3,4\})$, $1' = (1,\{2,4\})$, $1'' = (1,\{1,3,4\})$, $2' = (2,\{2,4\})$, $2'' = (2,\{1,3,4\})$, $3' = (3,\{2,4\})$, $4 = (4,\{\})$. Recall that $\textit{no-change}^{\Delta} = \{2,3,4,5\}$.

6.2. Soundness

We next state and prove a soundness result for safe-no-change projection.

Theorem 1 (Soundness of Safe-No-Change Projection)**.** *If a* SHA A *is schema-complete for some schema* $\mathbf{S}$*, then safe-no-change projection for* A *preserves the language within this schema:* $Ł(A^{snc}) \cap \mathbf{S} = Ł(A)$.

Proof. We have to prove that no more changing states $q \in P \cup \textit{no-change}^{\Delta}$ is sound. If $q \in \textit{no-change}^{\Delta}$ this follows from the schema-completeness of Δ, so that one can neither block on any hedge from the schema nor change the state. In the case $q \in P$, the intuition is that the state on level above – say r – can no more change, since then $P = \textit{s-no-change}^{\Delta}(r)$. Neither can the automaton block on any hedge from the schema by schema-completeness.

We first prove the inclusion $Ł(A) \subseteq Ł(A^{snc})$. Since $Ł(A) \subseteq \mathbf{S}$ by definition of schemas, this implies $Ł(A) \subseteq Ł(A^{snc}) \cap \mathbf{S}$. The proof will be based on the following three Claims 1.1a, 1.2a, and 1.3a. Note that schema-completeness is not needed for this direction.

Claim 1.1a. $\Pi \xrightarrow{h} \Pi$ wrt Δ^{snc} for all hedges $h \in \mathcal{H}_{\Sigma}$.

The proof is straightforward by induction on the structure of h. It uses the last three transition rules of Δ^{snc} in Fig. 8 permitting to always stay in Π for whatever hedge follows.

Claim 1.2a. For all $h \in \mathcal{H}_\Sigma$, $q \in \mathcal{Q}$, and $P \subseteq \mathcal{Q}$ such that $q \in P \cup \mathit{no\text{-}change}^\Delta$:

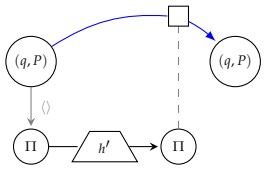
$$(q, P) \xrightarrow{h} (q, P) \text{ wrt } \Delta^{snc}$$

We prove Claim 1.2a by induction on the structure of h.

Case $h = \langle h' \rangle$**.** In this case, we can use Claim 1.1a to show $\Pi \xrightarrow{h'} \Pi$ wrt Δ^{snc} and the inference rules

$$\frac{q \in P \cup \mathit{no\text{-}change}^\Delta}{(q, P) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{snc}} \qquad \frac{q \in P \cup \mathit{no\text{-}change}^\Delta}{(q, P)@\Pi \to (q, P) \text{ in } \Delta^{snc}}$$

in order to close the following diagram with respect to Δ^{snc}:



This proves $(q, P) \xrightarrow{h} (q, P)$ wrt Δ^{snc} as required by the claim.

Case $h = a$**.** Since $q \in P \cup \mathit{no\text{-}change}^\Delta$ we can apply the inference rule:

$$\frac{a \in \Sigma \quad q \in P \cup \mathit{no\text{-}change}^\Delta}{(q, P) \xrightarrow{a} (q, P) \text{ in } \Delta^{snc}}$$

This proves this case of the claim.

Case $h = \varepsilon$**.** We trivially have $(q, P) \xrightarrow{\varepsilon} (q, P)$ wrt Δ^{snc}.

Case $h = h' \cdot h''$**.** By induction hypothesis applied to h' and h'' we have: $(q, P) \xrightarrow{h'} (q, P)$ and $(q, P) \xrightarrow{h''} (q, P)$ wrt Δ^{snc}. Hence $(q, P) \xrightarrow{h' \cdot h''} (q, P)$ wrt Δ^{snc}.

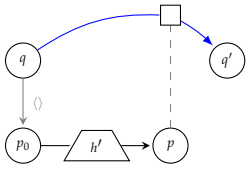
This ends the proof of Claim 1.2a. The next claim, in which the induction step is a little more tedious to prove, is the key of the soundness proof. We define the following predicate for all $q', q'' \in \mathcal{Q}$ and $P \subseteq \mathcal{Q}$:

$$q' \sim_P q'' \text{ iff } (q' = q'' \vee q', q'' \in P)$$

Claim 1.3a. Let $h \in \mathcal{H}_\Sigma$ a hedge, $q, q' \in \mathcal{Q}$ states and $P \subseteq \mathcal{Q}$ a subset of states such that $\mathit{acc}^\Delta(P) \subseteq P$ and $q \notin P \cup \mathit{no\text{-}change}^\Delta$. If $q \xrightarrow{h} q'$ wrt Δ then there exists q'' such that $(q, P) \xrightarrow{h} (q'', P)$ wrt Δ^{snc} and $q' \sim_P q''$.

Proof. By induction on the structure of h.

Case $h = \langle h' \rangle$**.** The assumption $q \xrightarrow{h} q'$ wrt Δ shows that there exists states $p_0 \in \langle\rangle^\Delta$ and $p \in \mathcal{Q}$ closing the following diagram:

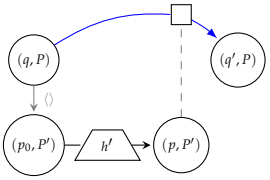


Let $P' = \textit{s-no-change}^{\Delta}(q)$ and note that $\textit{acc}^{\Delta}(P') \subseteq P'$. Since $q \notin P \cup \textit{no-change}^{\Delta}$ we can infer:

$$\frac{\xrightarrow{\langle\rangle} p_0 \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{\langle\rangle} (p_0,P') \text{ in } \Delta^{snc}} \qquad \frac{q@p \to q' \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P)@(p,P') \to (q',P) \text{ in } \Delta^{snc}}$$

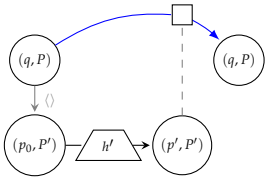
Subcase $p_0 \notin P' \cup \textit{no-change}^{\Delta}$**.** The induction hypothesis applies to h' shows that there exists p' such that $(p_0,P') \xrightarrow{h'} (p',P')$ wrt Δ^{snc} and $p \sim_P p'$. We distinguish the two cases justifying the latter predicate:

Subsubcase $p' = p$**.** Hence $(p_0,P') \xrightarrow{h'} (p,P')$ wrt Δ^{snc}, so we can close the diagram as follows:



This shows that $(q,P) \xrightarrow{h} (q',P)$ wrt Δ^{snc}, and thus the claim since $q' \sim_P q'$.

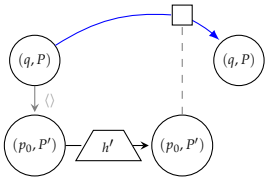
Subsubcase $p, p' \in P'$**.** Since $p' \in P'$ and $P' = \textit{s-no-change}^{\Delta}(q)$ we have $q@p' \to q$ in Δ. Hence we can close the diagram as follows:



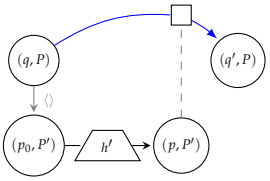
Since $p \in P'$ and $q@p \to q'$ in Δ we have $q = q'$ by definition of $P' = \textit{s-no-change}^{\Delta}(q)$. This shows that $(q,P) \xrightarrow{h} (q,P)$ wrt Δ^{snc}. Since $q \sim_P q$ the claim follows.

Subcase $p_0 \in P' \cup \textit{no-change}^{\Delta}$**.** Claim 1.2a then shows that $(p_0,P') \xrightarrow{h'} (p_0,P')$ wrt Δ^{snc}.

Subsubcase $p_0 \in P'$**.** Since $p \in \textit{acc}^{\Delta}(p_0)$ and $\textit{acc}^{\Delta}(P') \subseteq P'$ it follows that $p \in P'$ too. By definition $P' = \textit{s-no-change}^{\Delta}(q)$ and the completeness of Δ, the memberships $p_0 \in P'$ and $p \in P'$ imply that $q@^{\Delta}p_0 = \{q\} = q@^{\Delta}p$. We can now close the diagram below as follows:



Subsubcase $p_0 \in \textit{no-change}^{\Delta}$**.** In this case $p_0 = p$ so that $q' \in q@^{\Delta}p_0$. Hence:



Case $h = a$**.** Since $q \notin P \cup \textit{no-change}^{\Delta}$ we can apply the inference rule:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{a} (q',P) \text{ in } \Delta^{snc}}$$

This shows that $(q,P) \xrightarrow{h} (q',P)$ validating the claim since $q' \sim_P q'$.

Case $h = \varepsilon$**.** In this case we have $q = q'$ and $(q,P) \xrightarrow{\varepsilon} (q,P)$, so the claim holds.

Case $h = h_1 \cdot h_2$**.** Since $q \xrightarrow{h} q'$ wrt Δ, there exists $q_1 \in Q$ such that $q \xrightarrow{h_1} q_1$ wrt Δ and $q_1 \xrightarrow{h_2} q'$ wrt Δ. Since $q \notin P \cup \mathit{no\text{-}change}^\Delta$, we apply the induction hypothesis on h_1. This implies that there exists q_1' such that:

$$(q,P) \xrightarrow{h_1} (q_1',P) \text{ wrt } \Delta^{snc} \text{ and } q_1 \sim_P q_1'$$

We distinguish the two cases of $q_1 \sim_P q_1'$:

Subcase $q_1 = q_1'$**.** We also distinguish two subcases here:

Subsubcase $q_1 \notin P$**.** The induction hypothesis applied to h_2 yields:

$$\exists q''.\ (q_1,P) \xrightarrow{h_2} (q'',P) \text{ wrt } \Delta^{snc} \wedge q' \sim_P q''$$

Hence

$$\exists q''.\ (q,P) \xrightarrow{h} (q'',P) \text{ wrt } \Delta^{snc} \wedge q' \sim_P q''$$

Subsubcase $q_1 \in P$**.** By Claim 1.2a, we have $(q_1,P) \xrightarrow{h_2} (q_1,P)$. We also have $q' \in \mathit{acc}(\{q_1\})$ and since we assume $\mathit{acc}(P) \subseteq P$, this implies $q' \in P$. Hence $(q,P) \xrightarrow{h} (q_1,P)$ and $q', q_1 \in P$ implying the claim with $q' \sim_P q_1$.

Subcase $q_1, q_1' \in P$**.** Since $q_1' \in P$, Claim 1.2a, implies $(q_1',P) \xrightarrow{h_2} (q_1',P)$ wrt Δ^{snc}. Thus $(q,P) \xrightarrow{h} (q_1',P)$ wrt Δ^{snc}. Since $q' \in \mathit{acc}^\Delta(\{q_1\})$ and $q_1 \in P$ it follows that $q' \in \mathit{acc}^\Delta(P) \subseteq P$. Here we used as in the previous subsubcase that $\mathit{acc}(P) \subseteq P$ is assumed by the claim. Let $q'' = q_1'$. Then we have $(q,P) \xrightarrow{h} (q'',P)$ wrt Δ^{snc} and $q', q'' \in P$ showing the claim.

This ends the proof of Claim 1.3a.

Proof of inclusion $\text{Ł}(A) \subseteq \text{Ł}(A^{snc})$**.** Let $h \in \text{Ł}(A)$. Then there exists $q_0 \in I$ and $q \in F$ such that $q_0 \xrightarrow{h} q$. We distinguish two cases:

Case $q_0 \in \mathit{no\text{-}change}^\Delta$**.** By definition of $\mathit{no\text{-}change}^\Delta$ and since $q \in \mathit{acc}^\Delta(q_0)$ we have $q_0 = q$. Claim 1.2a shows that $(q_0,\emptyset) \xrightarrow{h} (q_0,\emptyset)$ wrt Δ^{snc} and thus $(q_0,\emptyset) \xrightarrow{h} (q,\emptyset)$ so that $h \in \text{Ł}(A^{snc})$.

Case $q_0 \notin \mathit{no\text{-}change}^\Delta$**.** Claim 1.3a with $P = \emptyset$ shows that $(q_0,\emptyset) \xrightarrow{h} (q,\emptyset)$ wrt Δ^{snc} and hence $h \in \text{Ł}(A^{snc})$.

This ends the proof of the first inclusion. We next want to show the inverse inclusion $\text{Ł}(A^{snc}) \cap \mathbf{S} \subseteq \text{Ł}(A)$. It will eventually follow from the following three Claims 1.1b, 1.2b, and 1.3b.

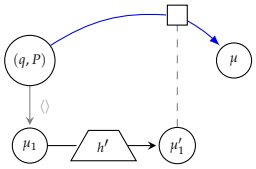
Claim 1.1b. For any hedge h and state $\mu \in Q^{snc}$, if $\Pi \xrightarrow{h} \mu$ wrt Δ^{snc} then $\mu = \Pi$.

The proof is straightforward by induction on the structure of h: the only transitions rules of Δ^{snc} with Π on the left hand side are inferred by the last three rules in Fig. 8. These require to stay in Π whatever hedge follows.

Claim 1.2b. For any hedge h, set $P \subseteq Q$, state $q \in P \cup \mathit{no\text{-}change}^\Delta$, and state $\mu \in Q^{snc}$: if $(q,P) \xrightarrow{h} \mu$ wrt Δ^{snc} then $\mu = (q,P)$.

Proof. By induction on the structure of h. Suppose that $(q,P) \xrightarrow{h} \mu$ wrt Δ^{snc}.

Case $h = \langle h' \rangle$**.** There must exist states $\mu_1, \mu_1' \in \mathcal{Q}^{snc}$ closing the following diagram:



Since $q \in P \cup \textit{no-change}^{\Delta}$, the following rule must have been applied to infer $(q,P) \xrightarrow{\langle\rangle} \mu_1$ wrt Δ^{snc}:

$$\frac{q \in P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{snc}}$$

Therefore $\mu_1 = \Pi$. Claim 1.1b shows that $\mu_1' = \Pi$ too. So μ must have been inferred by applying the rule:

$$\frac{q \in P \cup \textit{no-change}^{\Delta}}{(q,P)@\Pi \to (q,P) \text{ in } \Delta^{snc}}$$

So $\mu = (q,P)$ as required.

Case $h = a$**.** The following rule must have been applied:

$$\frac{q \in P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{a} (q,P) \text{ in } \Delta^{snc}}$$

Hence, $\mu = (q,P)$.

Case $h = \varepsilon$**.** Obvious.

Case $h = h_1 \cdot h_2$**.** There must exist μ_1 such that $(q,P) \xrightarrow{h_1} \mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{snc}. By induction hypothesis applied to h_1, we have $\mu_1 = (q,P)$. We can thus apply the induction hypothesis to h_2 to obtain $\mu_2 = (q,P)$.

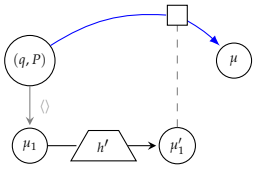
This ends the proof of Claim 1.2b. We next need an inverse of Claim 1.3a.

Claim 1.3b. Let $q \in \mathcal{Q}$, $P \subseteq \mathcal{Q}$ such that $q \notin P \cup \textit{no-change}^{\Delta}$ and $\textit{acc}^{\Delta}(P) \subseteq P$. For any $h \in \mathcal{H}_{\Sigma}$ such that $(q,P) \xrightarrow{h} \mu$ wrt Δ^{snc} for some $\mu \in \mathcal{Q}^{snc}$ and such that Δ does not have any blocking partial run on h starting from q, there exists q', q'' such that:

$$\mu = (q', P),\ q \xrightarrow{h} q'' \text{ wrt } \Delta, \text{ and } q' \sim_P q''.$$

Proof. By induction on the structure of $h \in \mathcal{H}_{\Sigma}$. We distinguish cases for all possible forms h.

Case $h = \langle h' \rangle$**.** By definition of $(q,P) \xrightarrow{h} \mu$ wrt Δ^{snc} there must exist $\mu_1, \mu_1' \in \mathcal{Q}^{snc}$ such that the following diagram can be closed:



Since $q \notin P \cup \textit{no-change}^{\Delta}$, the following rule got applied to infer $(q,P) \xrightarrow{\langle\rangle} \mu_1$ where $P' = \textit{s-no-change}^{\Delta}(q)$:

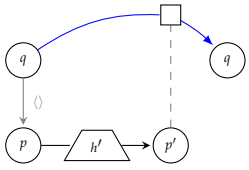
$$\frac{\xrightarrow{\langle\rangle} p \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q,P) \xrightarrow{\langle\rangle} (p,P') \text{ in } \Delta^{snc}}$$

Hence there exists $p \in \langle\rangle^{\Delta}$ such that $\mu_1 = (p, P')$. We fix such a state p arbitrarily. Since Δ does not have any blocking runs on h from q there exists $p', q'' \in Q$ such that $p \xrightarrow{h'} p'$ and $q@p' \to q''$ wrt Δ. Furthermore, Δ does not have any blocking partial run on h' starting from p.

Subcase $p \in P'$**.** In this case, we can apply Claim 1.2b to $(p, P') \xrightarrow{h'} \mu_1'$ wrt Δ in order to show that $\mu_1' = (p, P')$. Since $p \in acc^{\Delta}(\{p\})$ and $p \in P' = \textit{s-no-change}^{\Delta}(q)$ it follows that $q@^{\Delta}p = \{q\}$. Hence the following rule got applied to infer $(q, P) \xrightarrow{h} \mu$:

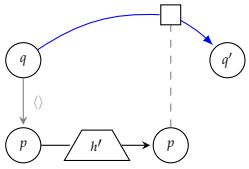
$$\frac{q@p \to q \text{ in } \Delta \quad q \notin P \cup \textit{no-change}^{\Delta}}{(q, P)@(p, P') \to (q, P) \text{ in } \Delta^{snc}}$$

This shows that $\mu = (q, P)$. Let $q' = q$ so that $\mu = (q', P)$. Since $p \xrightarrow{h'} p'$ wrt Δ we have $p' \in acc^{\Delta}(\{p\})$ so that $q@p' \to q$ wrt Δ by definition of $\textit{s-no-change}^{\Delta}(\{q\})$. Hence, we can close the following diagram:



Let $q'' = q$, so that $q' = q''$. It then holds that $q' \sim_P q''$ and $q \xrightarrow{\langle h' \rangle} q'$ wrt Δ, as required by the claim.

Subcase $p \in \textit{no-change}^{\Delta}$**.** In this case $p \xrightarrow{h'} p'$ wrt Δ implies that $p' = p$. Hence, we can close the following diagram:



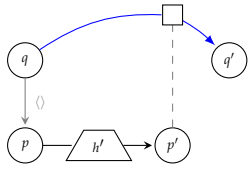
This shows that $q \xrightarrow{\langle h' \rangle} q'$ wrt Δ.

Subcase $p \notin P' \cup \textit{no-change}^{\Delta}$**.** By induction hypothesis applied to $(p, P') \xrightarrow{h'} \mu_1'$ wrt Δ there exists p'' such that:

$$\mu_1' = (p', P'),\ p \xrightarrow{h'} p'' \text{ wrt } \Delta, \text{ and } p' \sim_{P'} p''.$$

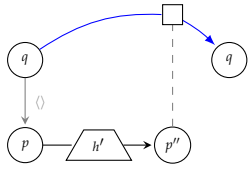
Since Δ does not have blocking runs on h starting with q there exist q'' such that $q@p'' \to q''$ wrt Δ. There are two ways to satisfy $p' \sim_{P'} p''$:

Subsubcase $p' = p''$**.** We then have:



This shows $q \xrightarrow{\langle h' \rangle} q'$ wrt Δ.

Subsubcase $p', p'' \in P$**.** By definition of P' it follows that $q' = q = q''$. Hence:



This shows $q \xrightarrow{\langle h' \rangle} q$ wrt Δ.

Case $h = a$**.** Since $q \notin P \cup \mathit{no\text{-}change}^{\Delta}$, the following inference rule must be used:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin P \cup \mathit{no\text{-}change}^{\Delta}}{(q,P) \xrightarrow{a} (q',P) \text{ in } \Delta^{snc}}$$

So $\mu = (q', P)$ and $q \xrightarrow{a} q'$ wrt Δ.

Case $h = \varepsilon$**.** Obvious.

Case $h = h_1 \cdot h_2$**.** The judgement $(q,P) \xrightarrow{h} \mu$ wrt Δ^{snc} shows that there exists μ_1 such that $(q,P) \xrightarrow{h_1} \mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{snc}. Since $q \notin P \cup \mathit{no\text{-}change}^{\Delta}$, we can apply the induction hypothesis to h_1. It shows that there exists q_1' and q_1'' such that $\mu_1 = (q_1', P)$, $q \xrightarrow{h_1} q_1''$ wrt Δ and $q_1' \sim_P q_1''$.

Subcase $q_1' = q_1''$**.** Hence $(q_1, P) \xrightarrow{h_2} \mu$ wrt Δ^{snc}.

Subsubcase $q_1' \notin P \cup \mathit{no\text{-}change}^{\Delta}$**.** In this case, we can apply the induction hypothesis to $(q_1', P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} showing the existence of q' such that $\mu = (q', P)$ and a state q'' such that $q_1' \xrightarrow{h_2} q''$ and $q' \sim_P q''$. Hence $\exists q''.\, q \xrightarrow{h} q''$ wrt Δ and $q' \sim_P q''$, so the claim holds.

Subsubcase $q_1' \in P \cup \mathit{no\text{-}change}^{\Delta}$**.** Claim 1.2b applied to $(q_1', P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} shows that $\mu = (q_1', P)$.

Subcase $q_1', q_1'' \in P$**.** Recall that $(q_1'', P) \xrightarrow{h_2} \mu$ wrt Δ^{snc} and $q_1'' \in P$. Claim 1.2b shows that $\mu = (q_1'', P)$ wrt Δ^{snc}. We also have $q \xrightarrow{h_1} q_1''$ wrt Δ. Since there are no blocking partial runs on h starting from q there exist a state q'' such that $q_1'' \xrightarrow{h_2} q''$ wrt Δ. Since $q_1' \in P$ and P is closed by accessibility, we have $q'' \in \mathit{acc}(\{q_1''\}) \subseteq \mathit{acc}(P) \subseteq P$. From $q \xrightarrow{h_1} q_1''$ wrt Δ, we get $q \xrightarrow{h} q''$ wrt Δ. Since $q_1'', q'' \in P$ it follows that $q_1'' \sim_P q''$ and thus the claim holds.

This ends the proof of Claim 1.3b.

Proof of inclusion $\text{Ł}(A^{snc}) \cap \mathbf{S} \subseteq \text{Ł}(A)$**.** Let $h \in \text{Ł}(A^{snc}) \cap \mathbf{S}$. Since $h \in \text{Ł}(A^{snc})$ then there exists $q_0 \in I$ and $q \in F$ such that $(q_0, \emptyset) \xrightarrow{h} (q, \emptyset)$ wrt Δ^{snc}. By Lemma 13 we have that $q_0 \xrightarrow{h} q$ wrt Δ.

We distinguish two cases:

Case $q_0 \in \mathit{no\text{-}change}^{\Delta}$**.** Claim 1.2b shows that $q = q_0$. Since A is schema-complete for $\mathbf{S}$, $h \in \mathbf{S}$, and $q_0 \in I \cap \mathit{no\text{-}change}^{\Delta}$, Lemma 13 shows that $q_0 \xrightarrow{h} q_0$ wrt Δ. Since $q = q_0$ this yields $h \in \text{Ł}(A)$.

Case $q_0 \notin \mathit{no\text{-}change}^{\Delta}$**.** Since A is schema-complete for $\mathbf{S}$ and $h \in \mathbf{S}$ there exist no blocking runs on h that start in q_0. Therefore, we can apply Claim 1.3b with $P = \emptyset$ to $(q_0, \emptyset) \xrightarrow{h} (q, \emptyset)$ wrt Δ^{snc}. This shows that $q_0 \xrightarrow{h} q$ wrt Δ and hence $h \in \text{Ł}(A)$.

This end the proof of the inverse inclusion, and thus of $L(A) = L(A^{snc})$. □

The projecting in-memory evaluator of A^{snc} will be more efficient than that non-projecting evaluator of A. Note, however, that the size of A^{snc} may be exponentially bigger than that of A. Therefore, for evaluating a dSHA A with subhedge projection on a given hedge h, we create only the needed part of A^{snc} on the fly. This part has size $O(|h|)$ and can be computed in time $O(|A|\,|h|)$, so the exponential preprocessing time is avoid.

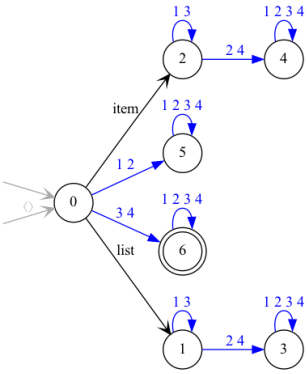


Figure 10. The unique minimal schema-complete dSHA with same initial and tree initial state for the XPath filter `[child::item]`. It is a counter example for the completeness of safe-no-change projection, see Fig. 11.

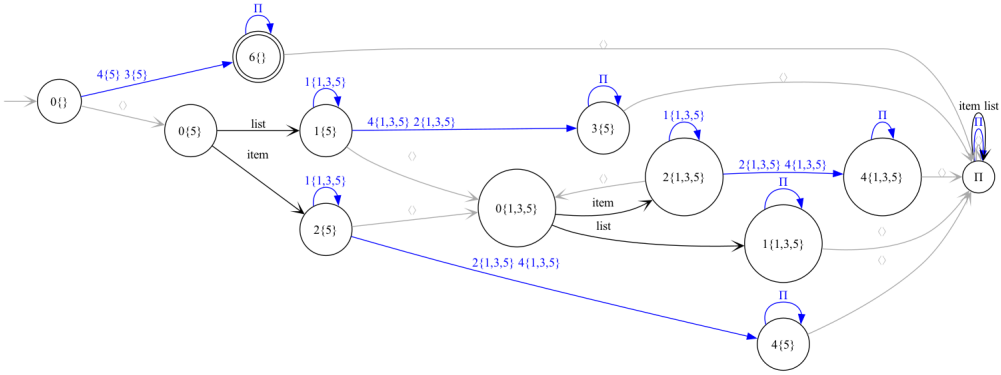


Figure 11. The safe-no-change projection $d\text{SHA}^{\downarrow}\ A^{snc}$ of the dSHA A in Fig. 10. It is incomplete for subhedge projection at the state $(2, \{1,3,5\})$. This state is not a subhedge projection state even though the prefixes $\langle \text{list} \langle \text{item}$ and $\langle \text{item} \langle \text{item}$ leading to it are subhedge irrelevant with schema $[\![doc]\!]$.

6.3. Incompleteness

Safe-no-change projection may be incomplete for subhedge projection, so that no all prefixes that are subhedge irrelevant are mappped to subhedge projection states. This is shown by the counter example in Fig. 10, where a dSHA A for the XPath filter `child::item` in Fig. 10 is given. The corresponding $d\text{SHA}^{\downarrow}\ A^{snc}$ obtained by safe-no-change projection is given in Fig. 11. Note that the prefix $\langle item . \langle item$ is subhedge irrelevant for the XPath filter `child::item`. Nevertheless its state $(2, \{1,3,4\})$ is not a subhedge projection state since $2 \notin \{1,3,4\}$. The problem is that this state may still be changed to $(4, \{1,3,4\})$, which is somehow equivalent with respect to the filter, but not equal.

Another incompleteness problem should be mentioned: Safe-no-change projection is sensible to automata completion. A state may belong to $no\text{-}change^{\Delta}$ before completion but no more afterwards, when adding a sink. But such states do never change on any tree satisfying the schema. This problem applies for instance in for the dSHA in Fig. 1. That is why we could not illustrate our safe-no-change algorithm with a complete automaton. Instead, we assumed schema-completeness only, and elaborated the whole paper for the more general case with schema-completeness (instead of general completeness).

7. Congruence Projection

For complete subhedge projection, we have to consider whether a state will remain equivalent when evaluating a subsequent subhedge. For safe-no-change projection, the equivalence relation chosen was state equality. In order to reach complete subhedge projection, we now consider richer equivalence relations. These will be congruences computed by the following congruence projection algorithm.

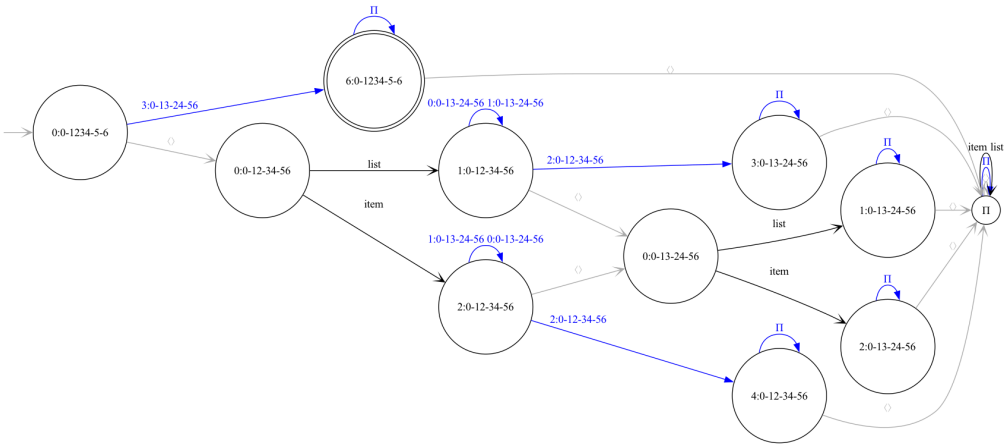


Figure 12. The congruence-projection $d\text{SHA}^{\downarrow}\ A^{cgr}_{[\![doc]\!]}$ constructed the counter example for safe-no-change projection, i.e., the dSHA A for the filter *child* :: *item* in Fig. 10. The schema final states are $F_{\mathbf{S}} = \{0, 5, 6\}$. The schema congruence $\sim^{\Delta}_{F_{\mathbf{S}}}$ has the equivalence classes $0 - 1234 - 56$. The start congruence $\sim^{\Delta}_{F_{\mathbf{S}} \to F}$ refines the schema congruence while splitting states 5 and 6, so it has the equivalence classes is $0 - 1234 - 5 - 6$. A state $(q, \sim^{\Delta}_{F_{\mathbf{S}} \to F})$ is a projection state if $q \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F}) = \mathcal{Q} \setminus \{0\}$. The equivalence relation $down^{\Delta}_0(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$ has the classes $056 - 12 - 34$, so that $\sim_0 = congr^{\Delta}(down^{\Delta}_0(\sim^{\Delta}_{F_{\mathbf{S}} \to F}))$ has the classes $0 - 12 - 34 - 56$ enabling projection in the states of $Prj^{\Delta}(\sim_0) = \{3, 4, 5, 6\}$. The equivalence relation $down^{\Delta}_1(\sim_0)$ has the classes $056 - 13 - 24$, so that $\sim_{01}$ has the classes $0 - 13 - 24 - 56$ allowing projection in states $Prj^{\Delta}(\sim_{01}) = \{1, 2, 3, 4, 5, 6\}$. The equivalence relation $down^{\Delta}_2(\sim_0)$ also has the classes $056 - 13 - 24$. Thus $\sim_{01} = \sim_{02}$ so that $Prj^{\Delta}(\sim_{02}) = \{1, 2, 3, 4, 5, 6\}$ too. Note that state $2 : 0 - 13 - 24 - 56$ reached over the prefixes $\langle \text{list} \langle \text{item}$ or $\langle \text{item} \langle \text{item}$ is a projection state.

Suppose that we are given a SHA $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ with schema $\mathbf{S} = \text{Ł}(A[F/F_{\mathbf{S}}])$ where $F \subseteq F_{\mathbf{S}} \subseteq \mathcal{Q}$. For stating the soundness later on, we will assume that A is schema-complete for $\mathbf{S}$, and for proving the completeness for subhedge projection, we will assume that A is deterministic and complete.

7.1. Algorithm

Let $\mathcal{E}_{\mathcal{Q}} \subseteq 2^{\mathcal{Q} \times \mathcal{Q}}$ be the set of all equivalence relations on states in $\mathcal{Q}$, i.e., the subset of reflexive and transitive relations.

Definition 14. *A precongruence wrt Δ is a reflexive and transitive relation $\lesssim \subseteq \mathcal{Q} \times \mathcal{Q}$ such that for all states $q_1, q_2, q_1' \in \mathcal{Q}$ and all hedges $h \in \mathcal{H}_{\Sigma}$:*

$$q_1 \lesssim q_2 \wedge q_1 \xrightarrow{h} q_1' \text{ wrt } \Delta \;\Rightarrow\; \exists q_2' \in \mathcal{Q}.\, q_2 \xrightarrow{h} q_2' \text{ wrt } \Delta \wedge q_1' \lesssim q_2'$$

A congruence wrt Δ is an equivalence relation $\approx \in \mathcal{E}_{\mathcal{Q}}$ that is a precongruence wrt Δ.

For any two subsets $P \subseteq P' \subseteq \mathcal{Q}$ we define a precongruence $\lesssim^{\Delta}_{P' \to P}$ such that for all states $q_1, q_2 \in \mathcal{Q}$ if for any hedge $h \in \mathcal{H}_{\Sigma}$ such that there is a transition from q_1 over h to some state of $q_1 \in P'$ there exists a transition from q_2 to some state $q_2' \in P'$ such that $q_1' \in P \Leftrightarrow q_2' \in P$:

$$q_1 \lesssim^{\Delta}_{P' \to P} q_2 \text{ iff } \begin{cases} \forall h \in \mathcal{H}_{\Sigma}. & \forall q_1' \in P'.\, (q_1 \xrightarrow{h} q_1' \text{ wrt } \Delta \\ \Rightarrow & \exists q_2' \in P'.\, q_2 \xrightarrow{h} q_2' \text{ wrt } \Delta \wedge (q_1' \in P \Leftrightarrow q_2' \in P)) \end{cases}$$

We can now define a congruence from this precongruence as follows:

$$\sim^{\Delta}_{P' \to P} \quad =_{\text{def}} \quad \lesssim^{\Delta}_{P' \to P} \cap \gtrsim^{\Delta}_{P' \to P}$$

Note that the congruence relation $\sim^{\Delta}_{P' \to P}$ can be computed from A, P, and P' in cubic time in the size of A. (For this one can derive a Datalog program for nonequivalence of two states from A, P, and P'.)

We also define $\sim^{\Delta}_{P} =_{\text{def}} \sim^{\Delta}_{P \to \mathcal{Q}}$.

The start congruence for the congruence projection is the congruence $\sim^{\Delta}_{F_{\mathbf{S}} \to F}$. The relation $\sim^{\Delta}_{F_{\mathbf{S}}}$ is called the schema equivalence. In the schema-less case where $\mathbf{S} = \mathcal{H}_{\Sigma}$, we assume that A is complete and $F_{\mathbf{S}} = \mathcal{Q}$. In this case, $\sim^{\Delta}_{F_{\mathbf{S}}} = \mathcal{Q} \times \mathcal{Q}$ is universal and thus trivial. The relation $\sim^{\Delta}_{F}$ is called the language equivalence. In case of deterministic finite state automata on words, note that the language equivalence is the usual Myhill-Nerode equivalence of automaton A as used for minimization.

The next lemma shows that the start congruence refines the schema equivalence.

Lemma 15. $\sim^{\Delta}_{F_{\mathbf{S}} \to F} \subseteq \sim^{\Delta}_{F_{\mathbf{S}}}$

Proof. Obvious □

In contrast, the start congruence does not always refine the language equivalence, since $\sim^{\Delta}_{F_{\mathbf{S}} \to F} \subseteq \sim^{\Delta}_{F}$ may go wrong (even though one may be tempted to run into this trap). The next lemma show how the start congruence refines the schema congruence. This yields a way to compute the start congruence from the schema equivalence too.

Lemma 16. *For any deterministic* SHA A*:*

$$q_1 \sim^{\Delta}_{F_{\mathbf{S}} \to F} q_2 \text{ iff } q_1 \sim^{\Delta}_{F_{\mathbf{S}}} q_2 \wedge \left\{ \begin{array}{l} \forall h \in \mathcal{H}_{\Sigma}. \forall q_1', q_2' \in F_{\mathbf{S}}. \\ \quad q_1 \xrightarrow{h} q_1' \text{ wrt } \Delta \wedge \\ \quad q_2 \xrightarrow{h} q_2' \text{ wrt } \Delta \end{array} \right\} \Rightarrow (q_1' \in F \Leftrightarrow q_2' \in F)$$

Proof. For the direction from the left to the right, we assume that $q_1 \sim^{\Delta}_{F_{\mathbf{S}} \to F} q_2$. Then we have $q_1 \sim^{\Delta}_{F_{\mathbf{S}}} q_2$ by Lemma 15. Let $h \in \mathcal{H}_{\Sigma}$ and $q_1', q_2' \in F_{\mathbf{S}}$ be arbitrary such that $q_1 \xrightarrow{h} q_1'$ wrt Δ and $q_2 \xrightarrow{h} q_2'$ wrt Δ. It then follows from $q_1 \lesssim^{\Delta}_{F_{\mathbf{S}} \to F} q_2$ that there exists $q_2'' \in F_{\mathbf{S}}$ such that $q_2 \xrightarrow{h} q_2''$ wrt $\Delta \wedge q_1' \in F \Leftrightarrow q_2'' \in F$. The determinism of A shows that $q_2' = q_2''$ and thus $q_1' \in F \Leftrightarrow q_2' \in F$.

For the other direction, we assume $q_1 \sim^{\Delta}_{F_{\mathbf{S}}} q_2$ and that for all hedges $h \in \mathcal{H}_{\Sigma}$ and states $q_1', q_2' \in F_{\mathbf{S}}$:

$$\left\{ \begin{array}{l} q_1 \xrightarrow{h} q_1' \text{ wrt } \Delta \wedge \\ q_2 \xrightarrow{h} q_2' \text{ wrt } \Delta \end{array} \right\} \Rightarrow (q_1' \in F \Leftrightarrow q_2' \in F)$$

We only show $q_1 \lesssim^{\Delta}_{F_{\mathbf{S}} \to F} q_2$ since $q_1 \gtrsim^{\Delta}_{F_{\mathbf{S}} \to F} q_2$ is symmetric. For this let $h \in \mathcal{H}_{\Sigma}$ and $q_1' \in F_{\mathbf{S}}$ such that $q_1 \xrightarrow{h} q_1'$ wrt Δ. By $q_1 \sim^{\Delta}_{F_{\mathbf{S}}} q_2$ it follows that there exists q_2'' such that $q_2 \xrightarrow{h} q_2''$ wrt Δ. By determinism, we have $q_2' = q_2''$. Hence our second assumption yields $q_1' \in F \Leftrightarrow q_2' \in F$. □

For any equivalence relation $\approx \in \mathcal{E}_{\mathcal{Q}}$ and state $q \in \mathcal{Q}$, we define another equivalence relation $down^{\Delta}_{q}(\approx) \in \mathcal{E}_{\mathcal{Q}}$ such that for all states $q_1, q_2 \in \mathcal{Q}$:

$$(q_1, q_2) \in down^{\Delta}_{q}(\approx) \text{ iff } q@^{\Delta}q_1 \approx q@^{\Delta}q_2$$

For any equivalence relation $\approx$ let $congr^{\Delta}(\approx)$ greatest congruence contained in $\approx$. We also define for any $q_1, q_2 \in \mathcal{Q}$, that $q_1 \approx^{\Delta} q_2$ if for all $h \in \mathcal{H}_{\Sigma}$:

$$\begin{aligned} & \forall q_1' \in \mathcal{Q}.\, q_1 \xrightarrow{h} q_1' \text{ wrt } \Delta \Rightarrow \exists q_2' \in \mathcal{Q}.\, (q_2 \xrightarrow{h} q_2' \text{ wrt } \Delta \wedge q_1' \approx q_2') \\ \wedge \quad & \forall q_2' \in \mathcal{Q}.\, q_2 \xrightarrow{h} q_2' \text{ wrt } \Delta \Rightarrow \exists q_1' \in \mathcal{Q}.\, (q_1 \xrightarrow{h} q_1' \text{ wrt } \Delta \wedge q_1' \approx q_2') \end{aligned}$$

Lemma 17. *$congr^{\Delta}(\approx) = \approx^{\Delta}$.*

Proof. It is not difficult to see that $\approx^{\Delta}$ is a congruence and that all other congruences contained in $\approx$ must contain $\approx^{\Delta}$. □

For any state $q \in \mathcal{Q}$ and equivalence relation $\approx \in \mathcal{E}_{\mathcal{Q}}$ we define a congruence as follows:

$$\approx_q = congr^{\Delta}(down_q^{\Delta}(\approx))$$

Reconsider the example dSHA in Fig. 1. The schema final states are $F_{\mathbf{S}} = \{0, 4, 5\}$. The equivalence classes of the schema congruence $\sim^{\Delta}_{F_{\mathbf{S}}}$ are $0 - 123 - 45$. The start congruence refines the schema congruence while distinguishing the states 4 and 5 in addition. Hence $\sim^{\Delta}_{F_{\mathbf{S}} \to F}$ has the equivalence classes $0 - 123 - 4 - 5$. Let $\approx = \sim^{\Delta}_{F_{\mathbf{S}} \to F}$. The equivalence relation $down_0^{\Delta}(\approx)$ has the classes $045 - 12 - 3$, and the congruence $\approx_0 = congr^{\Delta}(down_0^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F}))$ has the classes $0 - 1 - 2 - 3 - 45$. The equivalence relation $down_1^{\Delta}(\approx_0)$ has the classes $045 - 13 - 2$ and the congruence $\approx_{01}$ has the classes $0 - 13 - 2 - 45$.

Lemma 18. *If $p \approx_q p'$ then $(p, p') \in down_q^{\Delta}(\approx)$.*

Proof. By definition of operator $congr^{\Delta}$, the congruence $\approx_q = congr^{\Delta}(down_q^{\Delta}(\approx))$ is contained in equivalence relation $down_q^{\Delta}(\approx)$. (And it is the greatest congruence to do so.) □

For any equivalence relation $\approx$ in $\mathcal{E}_{\mathcal{Q}}$, we define a set of projection states by using hedge accessibility:

$$Prj^{\Delta}(\approx) = \{q \mid \forall q' \in acc^{\Delta}(q).\, q \approx q'\}$$

So a state q is a projection state with respect to $\approx$ if all states q' accessible from q over some hedge satisfy $q \approx q'$. This is $q \in Prj^{\Delta}(\approx) \Leftrightarrow q \in safe^{\Delta}(\{q' \mid q' \approx q\})$. In the example automaton, we have for instance, $2 \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$ since the only state accessible from state 2 is 2 itself. However, $0 \notin Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$, since 2 is accessible from state 0 but not $2 \sim^{\Delta}_{F_{\mathbf{S}} \to F} 0$.

Lemma 19. *If $\approx$ is a congruence, $q \in Prj^{\Delta}(\approx)$ and $q \approx p$ then $p \in Prj^{\Delta}(\approx)$.*

Proof. Let $p' \in acc^{\Delta}(p)$ be arbitrary. Then there exists $h \in \mathcal{H}_{\Sigma}$ such that $p \xrightarrow{h} p'$ wrt Δ. Since $\approx$ is a congruence and $q \approx p$, there exists q' such that $q \xrightarrow{h} q'$ wrt Δ and $q' \approx p'$. Hence $q' \in acc^{\Delta}(q)$ and since $q \in Prj^{\Delta}(\approx)$ it follows $q \approx q'$. Since $p' \approx q' \approx q \approx p$ by symmetry, transitivity yields $p' \approx p$. And since $p' \in acc^{\Delta}(p)$ was arbitrary, it follows that $p \in Prj^{\Delta}(\approx)$. □

Projection states for the initial congruence contain all states that are safe for selection or safe for rejection with respect to the schema:

Lemma 20. *$Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F}) \supseteq safe^{\Delta}(F) \cup safe^{\Delta}(F_{\mathbf{S}} \setminus F)$.*

Proof. We prove the two inclusions from the right to the left independently.

Case $q \in safe^{\Delta}(F)$**.** We have to show that $q \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$. Let $q' \in acc^{\Delta}(q)$ be arbitrary.

We have to show that $q \sim^{\Delta}_{F_{\mathbf{S}} \to F} q'$. Let $h \in \mathcal{H}_{\Sigma}$ and $\tilde{q}, \tilde{q}' \in \mathcal{Q}$ such that $q \xrightarrow{h} \tilde{q}$ wrt Δ

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin Prj^{\Delta}(\approx)}{(q,\approx) \xrightarrow{a} (q',\approx) \text{ in } \Delta^{cgr}} \qquad \frac{a \in \Sigma \quad q \in Prj^{\Delta}(\approx)}{(q,\approx) \xrightarrow{a} (q,\approx) \text{ in } \Delta^{cgr}}$$

$$\frac{\xrightarrow{\langle\rangle} q' \text{ in } \Delta \quad q \notin Prj^{\Delta}(\approx)}{(q,\approx) \xrightarrow{\langle\rangle} (\langle\rangle^{\Delta},\approx_q) \text{ in } \Delta^{cgr}} \qquad \frac{q \in Prj^{\Delta}(\approx)}{(q,\approx) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{cgr}}$$

$$\frac{q@q' \to q'' \text{ in } \Delta \quad q \notin Prj^{\Delta}(\approx)}{(q,\approx)@(q',\approx_q) \to (q'',\approx) \text{ in } \Delta^{cgr}} \qquad \frac{q \in Prj^{\Delta}(\approx)}{(q,\approx)@\Pi \to (q,\approx) \text{ in } \Delta^{cgr}}$$

$$\frac{a \in \Sigma}{\Pi \xrightarrow{a} \Pi \text{ in } \Delta^{cgr}} \qquad \frac{true}{\Pi@\Pi \to \Pi \text{ in } \Delta^{cgr}} \qquad \frac{true}{\Pi \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{cgr}}$$

Figure 13. The transitions rules Δ^{cgr} of the congruence projection $A^{cgr(\mathbf{S})}$.

and $q' \xrightarrow{h} \tilde{q}'$ wrt Δ. We have to show that $\tilde{q}' \in F \Leftrightarrow \tilde{q} \in F$ and $\tilde{q}' \in F_{\mathbf{S}} \Leftrightarrow \tilde{q} \in F_{\mathbf{S}}$. This follows since $q, q' \in safe^{\Delta}(F)$ so that $\tilde{q}, \tilde{q}' \in F \subseteq F_{\mathbf{S}}$.

Case $q \in safe^{\Delta}(F_{\mathbf{S}} \setminus F)$**.** We have to show that $q \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$. Let $q' \in acc^{\Delta}(q)$ be arbitrary. We have to show that $q \sim^{\Delta}_{F_{\mathbf{S}} \to F} q'$. Let $h \in \mathcal{H}_{\Sigma}$ and $\tilde{q}, \tilde{q}' \in \mathcal{Q}$ such that $q \xrightarrow{h} \tilde{q}$ wrt Δ and $q' \xrightarrow{h} \tilde{q}'$ wrt Δ. We have to show that $\tilde{q}' \in F \Leftrightarrow \tilde{q} \in F$ and $\tilde{q}' \in F_{\mathbf{S}} \Leftrightarrow \tilde{q} \in F_{\mathbf{S}}$. This follows since $q, q' \in safe^{\Delta}(F_{\mathbf{S}} \setminus F)$ so that $\tilde{q}, \tilde{q}' \in F_{\mathbf{S}} \setminus F$.

□

We now construct the congruence projection of a SHA$^{\downarrow}$ $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ with respect to schema $\mathbf{S} = Ł(A[F/F_{\mathbf{S}}])$ where $F \subseteq F_{\mathbf{S}} \subseteq \mathcal{Q}$:

$$A^{cgr(\mathbf{S})} = (\Sigma, \mathcal{Q}^{cgr}, \Delta^{cgr}, I^{cgr(\mathbf{S})}, F^{cgr(\mathbf{S})})$$

Let Π be a fresh state. The state set of the congruence projection automaton is:

$$\mathcal{Q}^{cgr} \subseteq \{\Pi\} \cup (\mathcal{Q} \times \mathcal{E}_{\mathcal{Q}})$$

The transition rules in Δ^{cgr} are given by the inference rules in Fig. 13 where $q, q', q'' \in \mathcal{Q}$, $\approx$ in $\mathcal{E}_{\mathcal{Q}}$ and $a \in \Sigma$. The sets of initial and final states of the projecting automaton are:

$$I^{cgr(\mathbf{S})} = \{(q, \sim^{\Delta}_{F_{\mathbf{S}} \to F}) \mid q \in I\} \qquad F^{cgr(\mathbf{S})} = \{(q, \sim^{\Delta}_{F_{\mathbf{S}} \to F}) \mid q \in F\}$$

For illustration, the congruence projection of the example dSHA in Fig. 1 from the introduction – defining the XPath filter `[self::list][child::item]` with schema $[\![doc]\!]$ – is given in Fig. 14. Up to state renaming this is the same automaton as obtained by safe-no-change projection in Fig. 2.

Next we reconsider the counter example dSHA in Fig. 10 with schema final states $F_{\mathbf{S}} = \{0, 5, 6\}$. The dSHA$^{\downarrow}$ resulting from congruence projection is shown in Fig. 12. We note with the prefix $\langle item.\langle item$ leads to state $2 : 0 - 13 - 24 - 5 - 6$. This is a projection state, since $2 \in Prj^{\Delta}(0 - 13 - 24 - 5 - 6) = \{1, 2, 3, 4, 5, 6\}$. In particular, note that $acc^{\Delta}(2) = \{2, 4\}$, so all accessible states from 2 are in the same congruence class. This means that the state 2 may still be changed to 4 but then remains equivalent now. This resolves the incompleteness issue with the safe-no-change projection on this example.

7.2. Soundness

We next the adapt the soundness result and proof from safe-no-change projection to congruence projection.

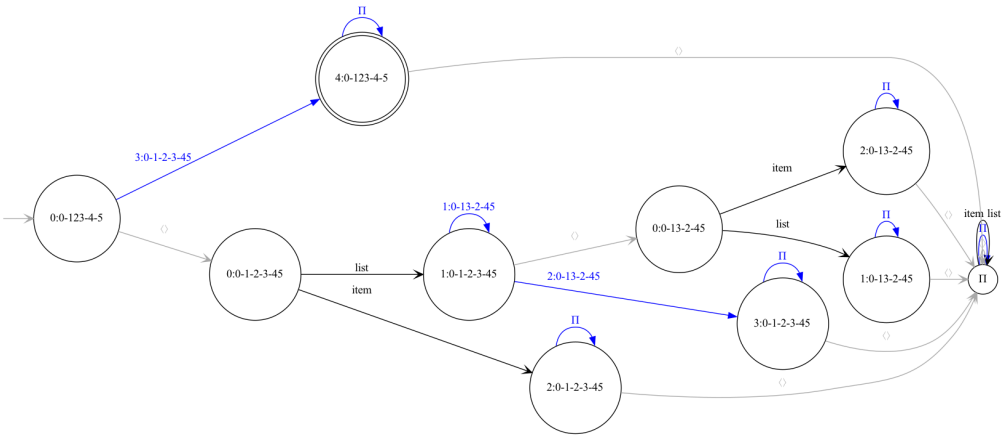


Figure 14. The congruence projection $d\text{SHA}^{\downarrow} A^{cgr([\![doc]\!])}$ constructed from the dSHA A for the XPath filter `[self::list][child::item]` with schema $[\![doc]\!]$ from the introduction in Fig. 1.

Theorem 2 (Soundness of Congruence Projection)**.** *If A is schema-complete for $\mathbf{S}$, then congruence projection preserves the language of A within schema $\mathbf{S}$, i.e.:* $Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S} = Ł(A)$.

Proof. The proof has the same structure as in the case of safe-no-change subhedge projection. We prove the first inclusion $Ł(A) \subseteq Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S}$ based on the following three Claims 2.1a, 2.2a, and 2.3a:

Claim 2.1a. $\Pi \xrightarrow{h} \Pi$ wrt Δ^{cgr} for all hedges $h \in \mathcal{H}_\Sigma$.

The proof is straightforward by induction on the structure of h. It uses the last three transition rules of Δ^{cgr} in Fig. 13 permitting to always stay in Π for whatever hedge follows.

Claim 2.2a. For all $h \in \mathcal{H}_\Sigma$, $q \in \mathcal{Q}$, and $\approx \in \mathcal{E}_\mathcal{Q}$ such that $q \in Prj^\Delta(\approx)$:

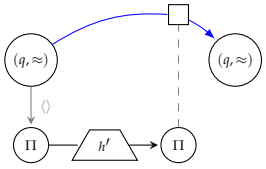
$$(q, \approx) \xrightarrow{h} (q, \approx) \text{ wrt } \Delta^{cgr}$$

We prove Claim 2.2a by induction on the structure of h.

Case $h = \langle h' \rangle$**.** In this case, we can use Claim 2.1a to show $\Pi \xrightarrow{h'} \Pi$ wrt Δ^{cgr} and the inference rules

$$\frac{q \in Prj^\Delta(\approx)}{(q, \approx) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{cgr}} \qquad \frac{q \in Prj^\Delta(\approx)}{(q, \approx)@\Pi \to (q, \approx) \text{ in } \Delta^{cgr}}$$

in order to close the following diagram with respect to Δ^{cgr}:



This proves $(q, \approx) \xrightarrow{h} (q, \approx)$ wrt Δ^{cgr} as required by the claim.

Case $h = a$**.** Since $q \in Prj^\Delta(\approx)$ we can apply the inference rule:

$$\frac{a \in \Sigma \quad q \in Prj^\Delta(\approx)}{(q, \approx) \xrightarrow{a} (q, \approx) \text{ in } \Delta^{cgr}}$$

This proves this case of the claim.

Case $h = \varepsilon$**.** We trivially have $(q, \approx) \xrightarrow{\varepsilon} (q, \approx)$ wrt Δ^{cgr}.

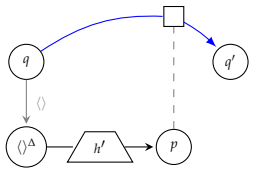
Case $h = h' \cdot h''$**.** By induction hypothesis applied to h' and h'' we have: $(q, \approx) \xrightarrow{h'} (q, \approx)$ and $(q, \approx) \xrightarrow{h''} (q, \approx)$ wrt Δ^{cgr}. Hence $(q, \approx) \xrightarrow{h' \cdot h''} (q, \approx)$ wrt Δ^{cgr}.

This ends the proof of Claim 2.2a. The next claim is the key of the soundness proof.

Claim 2.3a. Let $h \in \mathcal{H}_\Sigma$ a hedge, $q, q' \in \mathcal{Q}$ states and $\approx \in \mathcal{E}_\mathcal{Q}$ a congruence such that $q \notin Prj^\Delta(\approx)$. If $q \xrightarrow{h} q'$ wrt Δ then there exists $q'' \in \mathcal{Q}$ such that $(q, \approx) \xrightarrow{h} (q'', \approx)$ wrt Δ^{cgr} and $q'' \approx q'$.

Proof. By induction on the structure of h.

Case $h = \langle h' \rangle$**.** The assumption $q \xrightarrow{h} q'$ wrt Δ shows that there exists some state $p \in \mathcal{Q}$ closing the following diagram:



Let $\approx' = \approx_q$. Since $q \notin Prj^\Delta(\approx)$ we can infer:

$$\frac{\xrightarrow{\langle\rangle} \langle\rangle^\Delta \text{ in } \Delta \quad q \notin Prj^\Delta(\approx)}{(q, \approx) \xrightarrow{\langle\rangle} (\langle\rangle^\Delta, \approx') \text{ in } \Delta^{cgr}} \qquad \frac{q@p \to q' \text{ in } \Delta \quad q \notin Prj^\Delta(\approx)}{(q, \approx)@(p, \approx') \to (q', \approx) \text{ in } \Delta^{cgr}}$$

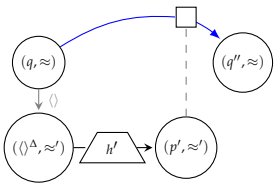
Subcase $\langle\rangle^\Delta \notin Prj^\Delta(\approx')$**.** The induction hypothesis applied to h' yields the existence of $p' \in \mathcal{Q}$ such that $p \approx' p'$ and:

$$(\langle\rangle^\Delta, \approx') \xrightarrow{h'} (p', \approx') \text{ wrt } \Delta^{cgr}$$

Since $p \approx_q p'$ we have $(p, p') \in down_q^\Delta(\approx)$ and thus $q@^\Delta p \approx q@^\Delta p'$. Since $q' \in q@^\Delta p$, there exists q'' such that $q' \approx q''$ and $q@p' \to q''$ wrt Δ. Therefore:

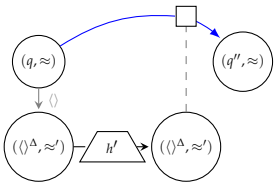
$$\frac{q@p' \to q' \text{ in } \Delta \quad q \notin Prj^\Delta(\approx)}{(q, \approx)@(p', \approx') \to (q'', \approx) \text{ in } \Delta^{cgr}}$$

Hence we can close the diagram as follows:



This shows that $(q, \approx) \xrightarrow{h} (q'', \approx)$ wrt Δ^{cgr}. The claim follows since $q' \approx q''$.

Subcase $\langle\rangle^\Delta \in Prj^\Delta(\approx')$**.** Claim 2.2a shows that $(\langle\rangle^\Delta, \approx') \xrightarrow{h'} (\langle\rangle^\Delta, \approx')$ wrt Δ^{cgr}. Since $p \in acc^\Delta(\langle\rangle^\Delta)$ the definition of $Prj^\Delta(\approx')$ implies that $\langle\rangle^\Delta \approx' p$. Hence $(\langle\rangle^\Delta, p) \in down_q^\Delta(\approx)$, so that $q@^\Delta\langle\rangle^\Delta \approx q@^\Delta p$. Therefore there exists q'' such that $q' \approx q''$ and $q@\langle\rangle^\Delta \to q''$ wrt Δ. We can thus close the diagram below as follows:



This shows that $(q,\approx) \xrightarrow{h} (q'',\approx)$ wrt Δ^{cgr}, so the claim follows since $q'\approx q''$.

Case $h = a$**.** Since $q \notin Prj^\Delta(\approx)$ we can apply the inference rule:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin Prj^\Delta(\approx)}{(q,\approx) \xrightarrow{a} (q',\approx) \text{ in } \Delta^{cgr}}$$

Hence $(q,\approx) \xrightarrow{h} (q',\approx)$ wrt Δ^{cgr}. Let q'' be equal to q'. Then the claim follows since $q'\approx q''$.

Case $h = \varepsilon$**.** In this case we have $q = q'$ and $(q,\approx) \xrightarrow{\varepsilon} (q,\approx)$ wrt Δ^{cgr}, so the claim holds.

Case $h = h_1 \cdot h_2$**.** Since $q \xrightarrow{h} q'$ wrt Δ there exists $q_1 \in \mathcal{Q}$ such that $q \xrightarrow{h_1} q_1$ wrt Δ and $q_1 \xrightarrow{h_2} q'$ wrt Δ. By induction hypothesis applied to h_1 there exists $q_1' \in \mathcal{Q}$ such that $(q,\approx) \xrightarrow{h_1} (q_1',\approx)$ wrt Δ^{cgr} and $q_1\approx q_1'$.

Subcase $q_1 \in Prj^\Delta(\approx)$**.** Claim 2.2a shows that $(q_1',\approx) \xrightarrow{h_2} (q_1',\approx)$ wrt Δ^{cgr} and therefore $(q,\approx) \xrightarrow{h} (q_1',\approx)$ wrt Δ^{cgr}. Since $q_1 \in Prj^\Delta(\approx)$ and $q' \in acc^\Delta(q_1)$, it follows that $q'\approx q_1$ and thus $q'\approx q_1'$. Hence, there exists q_1' such that $(q,\approx) \xrightarrow{h} (q_1',\approx)$ and $q'\approx q_1'$.

Subcase $q_1 \notin Prj^\Delta(\approx)$**.** Since $\approx$ is a congruence and $q_1 \xrightarrow{h_2} q'$ wrt Δ there exists $q'' \in \mathcal{Q}$ such that $q_1' \xrightarrow{h_2} q''$ wrt Δ and $q'\approx q''$. Since $q_1 \notin Prj^\Delta(\approx)$, we can apply the induction hypothesis to h_2 showing that there exists $q''' \in \mathcal{Q}$ such that $(q_1',\approx) \xrightarrow{h_2} (q''',\approx)$ wrt Δ^{cgr} and $q''\approx q'''$. By transitivity, we have $q'\approx q'''$. Furthermore, we have $(q_1,\approx) \xrightarrow{h} (q''',\approx)$ wrt Δ^{cgr} which shows the claim.

This ends the proof of Claim 2.3a.

Proof of inclusion $Ł(A) \subseteq Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S}$**.** Since $\mathbf{S}$ is a schema for A, we have $Ł(A) \subseteq \mathbf{S}$ so that it is sufficient to show $Ł(A) \subseteq Ł(A^{cgr(\mathbf{S})})$. Let $h \in Ł(A)$. Then there exists $q_0 \in I$ and $q \in F$ such that $q_0 \xrightarrow{h} q$ wrt Δ. Since $q_0 \in I$ it follows that $(q_0, \sim^\Delta_{F_\mathbf{S}\to F}) \in I^{cgr(\mathbf{S})}$. We distinguish two cases:

Case $q_0 \in Prj^\Delta(\sim^\Delta_{F_\mathbf{S}\to F})$**.** By definition of $Prj^\Delta(\sim^\Delta_{F_\mathbf{S}\to F})$ and $q \in acc^\Delta(\{q_0\})$, it follows that $q_0 \sim^\Delta_{F_\mathbf{S}\to F} q$. Thus, and since $q \in F$ it follows that $q_0 \in F$ too. Claim 2.2a shows that $(q_0, \sim^\Delta_{F_\mathbf{S}\to F}) \xrightarrow{h} (q_0, \sim^\Delta_{F_\mathbf{S}\to F})$ wrt Δ^{cgr}. Since $q_0 \in F$ it follows that $(q_0, \sim^\Delta_{F_\mathbf{S}\to F}) \in F^{cgr(\mathbf{S})}$. Hence $h \in Ł(A^{cgr(\mathbf{S})})$.

Case $q_0 \notin Prj^\Delta(\sim^\Delta_{F_\mathbf{S}\to F})$**.** Claim 2.3a shows that that there exists $q' \in \mathcal{Q}$ such that $q' \sim^\Delta_{F_\mathbf{S}\to F} q$ and $(q_0, \sim^\Delta_{F_\mathbf{S}\to F}) \xrightarrow{h} (q', \sim^\Delta_{F_\mathbf{S}\to F})$ wrt Δ^{cgr}. Since $q \in F$ it follows that $q' \in F$. Hence, $(q', \sim^\Delta_{F_\mathbf{S}\to F}) \in F^{cgr(\mathbf{S})}$ and thus $h \in Ł(A^{cgr(\mathbf{S})})$.

This ends the proof of the first inclusion.

We next want to show the inverse inclusion $Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S} \subseteq Ł(A)$. It will eventually follow from the following three Claims 2.1b, 2.2b, and 2.3b.

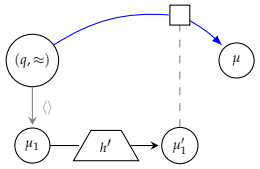
Claim 2.1b. For any hedge $h \in \mathcal{H}_\Sigma$ and state $\mu \in \mathcal{Q}^{cgr}$, if $\Pi \xrightarrow{h} \mu$ wrt Δ^{cgr} then $\mu = \Pi$.

The proof is straightforward by induction on the structure of h: the only transitions rules of Δ^{cgr} with Π on the left hand side are inferred by the last three rules in Fig. 13. These require to stay in Π whatever hedge follows.

Claim 2.2b. For any hedge $h \in \mathcal{H}_\Sigma$, equivalence relation $\approx \subseteq \mathcal{E}_\mathcal{Q}$, projection state $q \in Prj^\Delta(\approx)$ and state $\mu \in \mathcal{Q}^{cgr}$: if $(q, \approx) \xrightarrow{h} \mu$ wrt Δ^{cgr} then $\mu = (q, \approx)$.

Proof. By induction on the structure of $h \in \mathcal{H}_\Sigma$. Suppose that $(q, \approx) \xrightarrow{h} \mu$ wrt Δ^{cgr}.

Case $h = \langle h' \rangle$**.** There must exist states $\mu_1, \mu_1' \in \mathcal{Q}^{cgr}$ closing the following diagram:



Since $q \in Prj^\Delta(\approx)$, the following rule must have been applied to infer $(q, \approx) \xrightarrow{\langle\rangle} \mu_1$ wrt Δ^{cgr}:

$$\frac{q \in Prj^\Delta(\approx)}{(q, \approx) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{cgr}}$$

Therefore $\mu_1 = \Pi$. Claim 2.2a. shows that $\mu_1' = \Pi$ too. So μ must have been inferred by applying the rule:

$$\frac{q \in Prj^\Delta(\approx)}{(q, \approx)@\Pi \to (q, \approx) \text{ in } \Delta^{cgr}}$$

Hence, $\mu = (q, \approx)$ as required.

Case $h = a$**.** The following rule must have been applied:

$$\frac{a \in \Sigma \quad q \in Prj^\Delta(\approx)}{(q, \approx) \xrightarrow{a} (q, \approx) \text{ in } \Delta^{cgr}}$$

Hence, $\mu = (q, \approx)$.

Case $h = \varepsilon$**.** Obvious.

Case $h = h_1 \cdot h_2$**.** There must exist μ_1 such that $(q, \approx) \xrightarrow{h_1} \mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{cgr}. By induction hypothesis applied to h_1, we have $\mu_1 = (q, \approx)$. We can thus apply the induction hypothesis to h_2 to obtain $\mu_2 = (q, \approx)$.

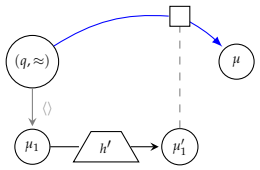
This ends the proof of Claim 2.2b. We next need an inverse of Claim 2.3a.

Claim 2.3b. Let $q \in \mathcal{Q}$ be a state and $\approx \in \mathcal{E}_\mathcal{Q}$ a congruence such that $q \notin Prj^\Delta(\approx)$. For any hedge $h \in \mathcal{H}_\Sigma$ with $(q, \approx) \xrightarrow{h} \mu$ wrt Δ^{cgr} for some state $\mu \in \mathcal{Q}^{cgr}$ such that Δ does not have any blocking partial run on h starting with q, there exists $q', q'' \in \mathcal{Q}$ such that:

$$\mu = (q', \approx),\ q \xrightarrow{h} q'' \text{ wrt } \Delta, \text{ and } q' \approx q''.$$

Proof. By induction on the structure of $h \in \mathcal{H}_\Sigma$. We distinguish all possible forms of hedges.

Case $h = \langle h' \rangle$**.** By definition of $(q, P) \xrightarrow{h} \mu$ wrt Δ^{cgr} there must exist $\mu_1, \mu_1' \in \mathcal{Q}^{cgr}$ such that the following diagram can be closed:



Since $q \notin Prj^{\Delta}(\approx)$, the following inference rule got applied to infer $(q,\approx) \xrightarrow{\langle\rangle} \mu_1$ wrt Δ^{cgr}:

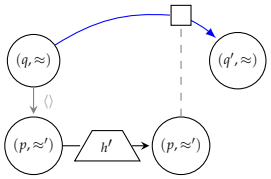
$$\frac{\xrightarrow{\langle\rangle} p \text{ in } \Delta \quad q \notin Prj^{\Delta}(\approx)}{(q,\approx) \xrightarrow{\langle\rangle} (p,\approx_q) \text{ in } \Delta^{cgr}}$$

Hence there exists $p \in \langle\rangle^{\Delta}$ such that $\mu_1 = (p,\approx')$ where $\approx' = \approx_q$. We fix such a state p arbitrarily.

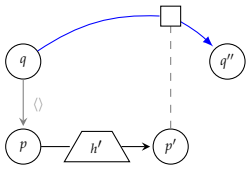
Subcase $p \in Prj^{\Delta}(\approx')$**.** In this case, Claim 2.2b applied to $(p,\approx') \xrightarrow{h'} \mu_1'$ wrt Δ yields that $\mu_1' = (p,\approx')$. The transition rule $(q,\approx)@\mu_1 \to \mu$ must thus be inferred as follows:

$$\frac{q@p \to q' \text{ in } \Delta \quad q \notin Prj^{\Delta}(\approx)}{(q,\approx)@(p,\approx') \to (q',\approx) \text{ in } \Delta^{cgr}}$$

This shows that $\mu = (q',\approx)$ for some $q' \in q@^{\Delta}p$. So we have the following diagram:

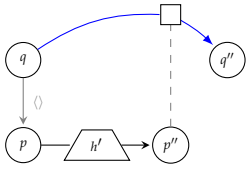


Since Δ does not have any blocking runs on h from q, it neither has any blocking run on h' from p, so there exists $p' \in \mathcal{Q}$ such that $p \xrightarrow{h'} p'$ wrt Δ. Since $p \in Prj^{\Delta}(\approx')$ and $p' \in acc^{\Delta}(p)$ it follows that $p \approx' p'$ by definition of Prj^{Δ}. Lemma 18 show that $(p,p') \in down^{\Delta}_{\approx}(q)$. This means $q@^{\Delta}p \approx q@^{\Delta}p'$. Since $q' \in q@^{\Delta}p$, it thus follows that there exists $q'' \in q@^{\Delta}p'$ such that $q' \approx q''$. With such a state q'', we can close the following diagram:



This shows that there exists $q'' \in \mathcal{Q}$ such that $q \xrightarrow{\langle h'\rangle} q''$ wrt Δ while $q' \approx q''$.

Subcase $p \notin Prj^{\Delta}(\approx')$**.** Since Δ does not have any blocking run on h from q, it neither has any blocking run on h' from p. Therefore, we can apply the induction hypothesis to $(p,\approx') \xrightarrow{h'} \mu_1'$ wrt Δ^{cgr}. It shows that there exists $p'' \in \mathcal{Q}$ such that $p' \approx' p''$, $\mu_1' = (p',\approx')$, and $p \xrightarrow{h'} p''$ wrt Δ. Since $p' \approx' p''$, Lemma 18 shows that $(p',p'') \in down^{\Delta}_{q}(\approx)$. That means $q@^{\Delta}p' \approx q@^{\Delta}p''$. Since $q' \in q@^{\Delta}p'$ it follows that there exists $q'' \in q@^{\Delta}p''$ such that $q' \approx q''$. Hence:



This shows that there exists $q'' \in \mathcal{Q}$ such that $q \xrightarrow{\langle h' \rangle} q''$ wrt Δ and $q' \approx q''$.

Case $h = a$**.** Since $q \notin Prj^{\Delta}(\approx)$, the following inference rule must be used:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q \notin Prj^{\Delta}(\approx)}{(q, \approx) \xrightarrow{a} (q', \approx) \text{ in } \Delta^{cgr}}$$

So $\mu = (q', \approx)$ and $q \xrightarrow{a} q'$ wrt Δ.

Case $h = \varepsilon$**.** Obvious.

Case $h = h_1 \cdot h_2$**.** Since $(q, \approx) \xrightarrow{h} \mu$ wrt Δ^{cgr} there exists $\mu_1 \in \mathcal{Q}^{cgr}$ such that $(q, \approx) \xrightarrow{h_1} \mu_1$ and $\mu_1 \xrightarrow{h_2} \mu$ wrt Δ^{cgr}. Furthermore $q \notin Prj^{\Delta}(\approx)$, so we can apply the induction hypothesis to h_1. Hence there exist states $q_1, q_1' \in \mathcal{Q}$ such that $\mu_1 = (q_1, \approx), q \xrightarrow{h_1} q_1'$ wrt Δ and $q_1 \approx q_1'$. We distinguish two cases:

Subcase $q_1 \in Prj^{\Delta}(\approx)$**.** Since $\mu_1 = (q_1, \approx) \xrightarrow{h_2} \mu$ wrt Δ^{cgr} and $q_1 \in Prj^{\Delta}(\approx)$, Claim 2.2b shows that $(q_1, \approx) \xrightarrow{h_2} (q_1, \approx)$ wrt Δ^{cgr}. In particular, $(q, \approx) \xrightarrow{h} (q_1, \approx)$ wrt Δ^{cgr}. Since h has no blocking partial run wrt Δ, h_2 may not have any blocking partial run with Δ starting from q_1'. Hence, there exists $q_2 \in \mathcal{Q}$ such that $q_1' \xrightarrow{h_2} q_2$ wrt Δ. In particular, $q \xrightarrow{h} q_2$ wrt Δ. It remains to show that $q_1 \approx q_2$. Since $q_1' \approx q_1$, $q_1 \in Prj^{\Delta}(\approx)$, and $\approx$ is a congruence, Lemma 19 shows that $q_1' \in Prj^{\Delta}(\approx)$ too. Furthermore, $q_2 \in acc^{\Delta}(q_1')$ so that $q_1' \in Prj^{\Delta}(\approx)$ implies $q_2 \approx q_1'$. The transitivity and symmetry of $\approx$ thus imply $q_1 \approx q_2$ as required.

Subcase $q_1 \notin Prj^{\Delta}(\approx)$**.** We can now apply the induction hypothesis to $(q_1, \approx) \xrightarrow{h_2} \mu$ showing the existence of $q_2, q_2' \in \mathcal{Q}$ such that $\mu = (q_2, \approx)$, $q_1 \xrightarrow{h_2} q_2'$ wrt Δ, and $q_2 \approx q_2'$. Since $q_1 \approx q_1'$ and $\approx$ is a congruence, there exists $q_2'' \in \mathcal{Q}$ such that $q_1' \xrightarrow{h_2} q_2''$ wrt Δ and $q_2' \approx q_2''$. In particular, $q \xrightarrow{h} q_2''$ and $q_2 \approx q_2''$ by transitivity. Therefore the claim holds.

This ends the proof of Claim 2.3b.

Proof of inclusion $Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S} \subseteq Ł(A)$**.** Let $h \in Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S}$. Since $h \in Ł(A^{cgr(\mathbf{S})})$ there exists $q_0 \in I$ and $q \in F$ such that $(q_0, \sim^{\Delta}_{F_{\mathbf{S}} \to F}) \xrightarrow{h} (q, \sim^{\Delta}_{F_{\mathbf{S}} \to F})$ wrt Δ^{cgr}. We distinguish two cases:

Case $q_0 \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$**.** Claim 2.2b shows that $q = q_0$, so $q_0 \in I \cap F$. Since $h \in \mathbf{S}$ and $q_0 \in I$ there exists $q' \in F_{\mathbf{S}}$ such that $q_0 \xrightarrow{h} q'$ wrt Δ. Note that $q' \in acc^{\Delta}(q_0)$. By definition of Prj^{Δ} we have $q_0 \sim^{\Delta}_{F_{\mathbf{S}} \to F} q'$, and thus $q_0 \lesssim^{\Delta}_{F_{\mathbf{S}} \to F} q'$. Since $q_0 \xrightarrow{\varepsilon} q_0$ wrt Δ and $q_0 \in F \subseteq F_{\mathbf{S}}$, there exists $q'' \in F_{\mathbf{S}}$ such that $q' \xrightarrow{\varepsilon} q''$ wrt Δ and $q_0 \in F \Leftrightarrow q'' \in F$. From $q' \xrightarrow{\varepsilon} q''$ wrt Δ it follows that $q' = q''$ and from $q_0 \in F$ and $(q_0 \in F \Leftrightarrow q'' \in F)$ that $q'' \in F$. Hence $q' \in F$, so that $q_0 \xrightarrow{h} q'$ wrt Δ yields $h \in Ł(A)$.

Case $q_0 \notin Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$**.** Since $h \in \mathbf{S}$ and A is schema-complete for $\mathbf{S}$ there exist no blocking runs of Δ on h starting in q_0. Therefore, we can apply Claim 2.3b to show that there exists q' such that $q_0 \xrightarrow{h} q'$ wrt Δ and $q \sim^{\Delta}_{F_{\mathbf{S}} \to F} q'$. Since $q \in F \subseteq F_{\mathbf{S}}$ and $q \xrightarrow{\varepsilon} q$ wrt Δ, $q \lesssim^{\Delta}_{F_{\mathbf{S}} \to F} q'$ implies that there exists $q'' \in F_{\mathbf{S}}$ such that $q' \xrightarrow{\varepsilon} q''$ wrt Δ and $q \in F \Leftrightarrow q'' \in F$. From $q' \xrightarrow{\varepsilon} q''$ wrt Δ it follows that $q' = q''$. From $q \in F$ and $(q \in F \Leftrightarrow q'' \in F)$ it follows that $q'' \in F$. Hence $q' \in F$ so that $q_0 \xrightarrow{h} q'$ wrt Δ yields $h \in Ł(A)$.

This ends the proof of the inverse inclusion, and thus of $Ł(A) = Ł(A^{cgr(\mathbf{S})}) \cap \mathbf{S}$. □

7.3. Completeness

We next show the completeness of congruence projection according to Definition 11. For this we eventually need to assume that the SHAs are deterministic and complete.

Let $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be a SHA with schema $\mathbf{S} = Ł(A[F/F_{\mathbf{S}}])$ where $F \subseteq F_{\mathbf{S}} \subseteq \mathcal{Q}$. We first show that all projection states of the congruence projection $A^{cgr(\mathbf{S})}$ are subhedge projection states according to Definition 8.

Lemma 21. *If $q \in Prj^{\Delta}(\approx)$ then $(q, \approx)$ is a subhedge projection state of Δ^{cgr} with witness Π.*

Proof. We assume that $q \in Prj^{\Delta}(\approx)$ and have to show that $(q, \approx)$ is a subhedge projection state of Δ^{cgr}. We have to show that all transition rules starting with $(q, \approx)$ and Π are permitted for a subhedge projection state $(q, \approx)$ with withness Π. For transitions starting of Δ^{cgr} with Π this is obvious, since all of them are generated by the following rules:

$$\frac{a \in \Sigma}{\Pi \xrightarrow{a} \Pi \text{ in } \Delta^{cgr}} \qquad \frac{true}{\Pi@\Pi \to \Pi \text{ in } \Delta^{cgr}} \qquad \frac{true}{\Pi \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{cgr}}$$

We have to consider all transition rules of Δ^{cgr} starting with $(q, \approx)$. We consider all forms of such transition rules. For this let $\mu, \mu' \in \mathcal{Q}^{cgr}$ and $a \in \Sigma$ be arbitrary.

Case $(q, \approx) \xrightarrow{a} \mu$ **wrt** Δ^{cgr}**.** By definition of subhedge projection states, we must show that $\mu = (q, \approx)$. Since $q \in Prj^{\Delta}(\approx)$ the above transition must have been inferred by the following rule:

$$\frac{a \in \Sigma \quad q \in Prj^{\Delta}(\approx)}{(q, \approx) \xrightarrow{a} (q, \approx) \text{ in } \Delta^{cgr}}$$

Hence $\mu = (q, \approx)$ as required.

Case $(q, \approx)@\mu \to \mu'$ **wrt** Δ^{cgr}**.** By definition of subhedge projection states with witness Π, we must show that $\mu = \Pi$ and $\mu' = (q, \approx)$. Since $q \in Prj^{\Delta}(\approx)$ the above transition must have been inferred by the following rule:

$$\frac{q \in Prj^{\Delta}(\approx)}{(q, \approx)@\Pi \to (q, \approx) \text{ in } \Delta^{cgr}}$$

This show indeed that $\mu = \Pi$ and $\mu' = (q, \approx)$.

Case $(q, \approx) \xrightarrow{\langle\rangle} \mu$ **wrt** Δ^{cgr}**.** By definition of subhedge projection states with witness Π, we must show that $\mu = \Pi$. Since $q \in Prj^{\Delta}(\approx)$ the above transition must have been inferred by the following rule:

$$\frac{q \in Prj^{\Delta}(\approx)}{(q, \approx) \xrightarrow{\langle\rangle} \Pi \text{ in } \Delta^{cgr}}$$

Hence, $\mu = \Pi$ as required.

□

Proposition 10 shows for any deterministic SHA$^{\downarrow}$ that subhedges starting at subhedge projection states are irrelevant. Hence by Lemma 21, if A is a dSHA with schema $\mathbf{S}$, then all subhedges starting at some projection state $(q, \approx)$ of A^{cgr} where $q \in Prj^{\Delta}(\approx)$ are irrelevant for $Ł(A)$ with respect to $\mathbf{S}$.

The induction step of the completeness proof (Proposition 26) requires a nontrivial generalization of the notion of subhedge irrelevance that we introduce next. For any state $q \in \mathcal{Q}$ let $jump_q$ be a fresh symbol. Let $\tilde{A} = (\tilde{\Sigma}, \mathcal{Q}, \tilde{\Delta}, I, F)$ be like A, except that the signature

$\tilde{\Sigma}$ adds all jump symbols for Σ and that $\tilde{\Delta}$ extends Δ with the following transition rules for all $p, q \in \mathcal{Q}$:

$$p \xrightarrow{jump_q} q \in \tilde{\Delta}$$

Lemma 22. *For any hedge $h \in \mathcal{H}_\Sigma$ and state $p \in \mathcal{Q}$: $[\![jump_{[\![h]\!](p)}]\!] = [\![jump_p \cdot h]\!]$.*

Proof. Obvious. □

For any nested word prefix v we define a binary relation $rel^A_{\mathbf{S}}(v) \subseteq \mathcal{Q} \times \mathcal{Q}$ so that for all states $p, p' \in \mathcal{Q}$:

$$(p, p') \in rel^A_{\mathbf{S}}(v) \Leftrightarrow \left\{ \forall w \in \hat{\Sigma}.\, v \cdot w \in nw(\mathcal{H}_\Sigma).\, \forall q_0 \in I.\, \forall q, q' \in F_{\mathbf{S}}. \left(\begin{array}{l} q_0 \xrightarrow{hdg(v \cdot jump_p \cdot w)} q \text{ wrt } \tilde{\Delta} \wedge \\ q_0 \xrightarrow{hdg(v \cdot jump_{p'} \cdot w)} q' \text{ wrt } \tilde{\Delta} \end{array} \right) \Rightarrow\ (q \in F \Leftrightarrow q' \in F) \right.$$

Note that the relation $rel^A_{\mathbf{S}}(v)$ is neither supposed to be transitive, nor an equivalence relation, nor a congruence. It mainly serves to formulate the induction invariant (see Claim 27 below).

Lemma 23. *Let the* SHA *be deterministic and v a nested word prefix that is subhedge irrelevant for $Ł(A)$ wrt $\mathbf{S}$. For any two states $p, p' \in \mathcal{Q}$ such that A has a partial run on v to p and $p' \in acc^\Delta(p)$ it holds that $(p, p') \in rel^A_{\mathbf{S}}(v)$.*

Proof. Let v be a nested word prefix such that A has a partial run on v from q_0 to p and $p' \in acc^\Delta(p)$. Then there exists a nested word u such that $p \xrightarrow{hdg(u)} p'$ wrt Δ. So A has a partial run on $v \cdot u$ to p'.

Suppose that v is subhedge irrelevant for $Ł(A)$ wrt $\mathbf{S}$. So for all words $w \in \hat{\Sigma}^*$ such that $v \cdot w \in nw(\mathbf{S})$ and $v \cdot u \cdot w \in nw(\mathbf{S})$ we have:

$$v \cdot w \in nw(Ł(A)) \Leftrightarrow v \cdot u \cdot w \in nw(Ł(A))$$

Since A is deterministic, this shows that for all $w \in \hat{\Sigma}^*$ and $q, q' \in F_{\mathbf{S}}$:

$$\left(\begin{array}{l} q_0 \xrightarrow{hdg(v \cdot w)} q \text{ wrt } \Delta \wedge \\ q_0 \xrightarrow{hdg(v \cdot u \cdot w)} q' \text{ wrt } \Delta \end{array} \right) \Rightarrow (q \in F \Leftrightarrow q' \in F)$$

Since A has a partial run on v from q_0 to p and a partial run on $v \cdot u$ from q_0 to p', this implies for all words $w \in \hat{\Sigma}^*$ such that $v \cdot w \in nw(\mathbf{S})$:

$$\left(\begin{array}{l} q_0 \xrightarrow{hdg(v \cdot jump_p \cdot w)} q \text{ wrt } \tilde{\Delta} \wedge \\ q_0 \xrightarrow{hdg(v \cdot jump_{p'} \cdot w)} q' \text{ wrt } \tilde{\Delta} \end{array} \right) \Rightarrow (q \in F \Leftrightarrow q' \in F)$$

Hence $(p, p') \in rel^A_{\mathbf{S}}(v)$. □

We next show that Lemmas 21 and 23 imply the completeness of congruence projection for subhedge irrelevance of nested words – but not yet prefixes thereof – and thus for DFAs on words in Σ^* in particular.

Lemma 24. *Let the* SHA *A be deterministic, v be a nested word, and $p, p' \in \mathcal{Q}$ such that A has a partial run on v to p and $p' \in acc^\Delta(p)$. The membership judgement $(p, p') \in rel^A_{\mathbf{S}}(v)$ then implies $p \sim^\Delta_{F_{\mathbf{S}} \to F} p'$.*

Proof. Since A is deterministic there exists a unique state $q_0 \in I$ such that $q_0 \xrightarrow{hdg(v)} p$ wrt Δ. Suppose that $(p, q') \in rel^A_{\mathbf{S}}(v)$. Then for all nested words w and $q, q' \in F_{\mathbf{S}}$:

$$\left(\begin{array}{l} q_0 \xrightarrow{hdg(v \cdot w)} q \text{ wrt } \Delta \wedge \\ q_0 \xrightarrow{hdg(v \cdot u \cdot w)} q' \text{ wrt } \Delta \end{array} \right) \Rightarrow (q \in F \Leftrightarrow q' \in F)$$

Since $p' \in acc^{\Delta}(p)$, there exists a nested word u such that $p \xrightarrow{hdg(u)} p'$. Since $q_0 \xrightarrow{hdg(v)} p$ wrt Δ, this u satisfies for all $w \in nw(\mathcal{H}_\Sigma)$ and $q, q' \in F_{\mathbf{S}}$:

$$\left(\begin{array}{l} p \xrightarrow{hdg(w)} q \text{ wrt } \Delta \wedge \\ p' \xrightarrow{hdg(w)} q' \text{ wrt } \Delta \end{array} \right) \Rightarrow \left(\begin{array}{l} q_0 \xrightarrow{hdg(v \cdot w)} q \text{ wrt } \Delta \wedge \\ q_0 \xrightarrow{hdg(v \cdot u \cdot w)} q' \text{ wrt } \Delta \end{array} \right)$$

Hence for all nested words $w \in nw(\mathcal{H}_\Sigma)$ and $q, q' \in F_{\mathbf{S}}$:

$$\left(\begin{array}{l} p \xrightarrow{hdg(w)} q \text{ wrt } \Delta \wedge \\ p' \xrightarrow{hdg(w)} q' \text{ wrt } \Delta \end{array} \right) \Rightarrow (q \in F \Leftrightarrow q' \in F)$$

This proves that $p \sim^{\Delta}_{F_{\mathbf{S}} \to F} p'$. □

Proposition 25. *Let the* SHA A *be deterministic and v a nested word such that $q_0 \xrightarrow{hdg(v)} p$ wrt Δ for some $q_0 \in I$ and $p \in \mathcal{Q}$. If v is subhedge irrelevant for $\text{Ł}(A)$ wrt $\mathbf{S}$ then $p \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$.*

Proof. Since v is subhedge irrelevant for $\text{Ł}(A)$ wrt $\mathbf{S}$ it follows by Lemma 23 for any $p' \in acc^{\Delta}(p)$ that $(p, p') \in rel^A_{\mathbf{S}}(v)$, and by Lemma 24 that $p \sim^{\Delta}_{F_{\mathbf{S}} \to F} p'$. That is $p \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})$. □

Corollary 3 (Completeness of congruence projection on words)**.** *If A is a* DFA *that is schema-complete for $\mathbf{S}$ then $A^{cgr(\mathbf{S})}$ is complete for subhedge projection.*

Proof. This follows from Proposition 25 and Lemma 21 since $I^{cgr} = \{(q, \sim^{\Delta}_{F_{\mathbf{S}} \to F} | \ q \in I)\}$ and $F^{cgr} = \{(q, \sim^{\Delta}_{F_{\mathbf{S}} \to F}) \mid q \in Prj^{\Delta}(\sim^{\Delta}_{F_{\mathbf{S}} \to F})\}$. □

Proposition 26. *Let $\mathcal{A}$ be a d*SHA *with schema $\mathbf{S}$, $v, w \in \hat{\Sigma}^*$ such that $v \cdot w \in nw(\mathcal{H}_\Sigma)$, and $u \in nw(\mathcal{H}_\Sigma)$. Suppose that there exist $p, p' \in \mathcal{Q}$ and a run of of A on $hdg(v \cdot w)$ that goes to p at v and a run on $hdg(v \cdot u \cdot w)$ that goes to p' at $v \cdot u$. Let $\approx \in \mathcal{E}_{\mathcal{Q}}$ be such that $(p, \approx)$ is the state of $A^{cgr(\mathbf{S})}$ at v. If $(p, p') \in rel^A_{\mathbf{S}}(v)$ then it follows that $p \approx p'$.*

Proof. Let v be a nested word prefix. So there exists a natural number $n \in \mathbb{N}$ and nested words $v_0, \ldots, v_n \in nw(\mathcal{H}_\Sigma)$ such that:

$$v = v_0 \cdot \langle \cdot v_1 \cdot \ldots \cdot \langle \cdot v_n$$

Let u be a nested word and $p, p' \in \mathcal{Q}$ such that there exist a partial run of A on v to p and on $v \cdot u$ to p'. For $0 \leq i \leq n-1$ let:

$$v^i = v_0 \cdot \langle \cdot \ldots \cdot \langle \cdot v_i$$

Note that $v^n = v$. Let q_i be the state of the partial run of A on the prefix v^i. Let $w \in \hat{\Sigma}^*$ be such that $v \cdot w \in nw(\mathcal{H}_\Sigma)$ and $u \in nw(\mathcal{H}_\Sigma)$, such that A has a run on $hdg(v \cdot w)$ that goes to

p at v and a run on $hdg(v \cdot u \cdot w)$ that goes to p' at $v \cdot u$. Since $v \cdot w$ is a nested word, there must exist nested words $w_0 \dots w_n \in nw(\mathcal{H}_\Sigma)$ with:

$$w = w_n \cdot \rangle \cdot \ldots \cdot \rangle \cdot w_0$$

For all $0 \leq i \leq n$ let $p_i, p'_i \in Q$ be the unique states such that:

$$p_i \in q_i @^\Delta [\![w_i]\!](q_{i+1} @^\Delta \dots q_{n-1} @^\Delta [\![w_n]\!](p))$$
$$p'_i \in q_i @^\Delta [\![w_i]\!](q_{i+1} @^\Delta \dots q_{n-1} @^\Delta [\![w_n]\!](p'))$$

These states exist, since there exists a runs of A on $v \cdot w$ and $v \cdot u \cdot w$, and they are unique since A is deterministic. Note that $p_n = p$ and $p'_n = p'$, while for all $0 \leq i < n$ it holds that $p_i \in q_i @^\Delta [\![w_i]\!](p_{i+1})$ and $p'_i \in q_i @^\Delta [\![w_i]\!](p'_{i+1})$.

Claim 27. *For all* $0 \leq i \leq n$: $(p_i, p'_i) \in rel^A_{\mathbf{S}}(v^i)$.

The proof is by induction on i from n to 1. The induction starts with $i = n$. Since $p_n = p$ and $p'_n = p'$, the assumption $(p, p') \in rel^A_{\mathbf{S}}(v)$ shows that $(p_n, p'_n) \in rel^A_{\mathbf{S}}(v^n)$. For the induction step, we consider $i < n$. The induction hypothesis ensures $(p_{i+1}, p'_{i+1}) \in rel^A_{\mathbf{S}}(v^{i+1})$. We have to show $(p_i, p'_i) \in rel^A_{\mathbf{S}}(v^i)$. Let $\tilde{w} \in \hat{\Sigma}^*$ and $r_i, r'_i \in F_{\mathbf{S}}$ be arbitrary so that:

$$q_0 \xrightarrow{hdg(v^i \cdot jump_{p_i} \cdot \tilde{w})} r_i \text{ wrt } \tilde{\Delta} \wedge$$
$$q_0 \xrightarrow{hdg(v^i \cdot jump_{p'_i} \cdot \tilde{w})} r'_i \text{ wrt } \tilde{\Delta}$$

Since $p_i \in q_i @^\Delta [\![w_i]\!](p_{i+1})$, and $v^{i+1} = v^i \cdot \langle \cdot v_{i+1}$, and since A maps v^i to q_i, the above any Lemma 22 imply:

$$q_0 \xrightarrow{hdg(v^{i+1} \cdot jump_{p_{i+1}} \cdot w_i \cdot \rangle \cdot \tilde{w})} r_i \text{ wrt } \tilde{\Delta} \wedge$$
$$q_0 \xrightarrow{hdg(v^{i+1} \cdot jump_{p'_{i+1}} \cdot w_i \cdot \rangle \cdot \tilde{w})} r'_i \text{ wrt } \tilde{\Delta}$$

Since we have $(p_{i+1}, p'_{i+1}) \in rel^A_{\mathbf{S}}(v^{i+1})$, the above yields $r_i \in F \Leftrightarrow r'_i \in F$. And since $\tilde{w}, r_i, r'_i$ were chosen arbitrarily, it follows that $(p_i, p'_i) \in rel^A_{\mathbf{S}}(v^i)$ as required for $i < n$. This ends the proof of Claim 27.

For all $0 \leq i \leq n$ let $\approx^i \in \mathcal{E}_Q$ be equivalence relation such that $A^{cgr(\mathbf{S})}$ maps v^i to $(q_i, \approx^i)$.

Claim 28. *For all* $0 \leq i \leq n$: $p_i \approx^i p'_i$.

The proof is by induction on i from 0 to n. First note that Claim 27 show that $(p_i, p'_i) \in rel^A_{\mathbf{S}}(v^i)$ for all $0 \leq i \leq n$. For induction base $i = 0$, the prefix v^0 is a nested word. Lemma 24 shows that $p_0 \sim^\Delta_{F_{\mathbf{S}} \to F} p'_0$. Furthermore, $\approx^0 = \sim^\Delta_{F_{\mathbf{S}} \to F}$ since $I^{cgr} = \{(p, \sim^\Delta_{F_{\mathbf{S}} \to F}) \mid p \in I\}$. Hence, $p_0 \approx^0 p'_0$. For the induction step from i to $i+1$, we note that $\approx^{i+1} = (\approx^i)_{q_i}$ by construction of Δ^{cgr}. The induction hypothesis yields $p_i \approx^i p'_i$. Furthermore $p_i \in q_i @^\Delta [\![w_{i+1}]\!](p_{i+1})$ and $p'_i \in q_i @^\Delta [\![w_{i+1}]\!](p'_{i+1})$. Since A is deterministic it follows that $q_i @^\Delta [\![w_{i+1}]\!](p_{i+1}) = \{p_i\}$ and $q_i @^\Delta [\![w_{i+1}]\!](p'_{i+1}) = \{p'_i\}$. Since $p_i \approx^i p'_i$, this yields $(q_i @^\Delta [\![w_{i+1}]\!](p_{i+1}), q_i @^\Delta [\![w_{i+1}]\!](p'_{i+1})) \in (\approx^i)_{q_i}$. That is $p_{i+1} \approx^{i+1} p'_{i+1}$ which ends the proof of Claim 28.

Claim 28 with $i = n$ shows $p_n \approx^n p'_n$. This is equivalent to $p \approx p'$ as stated by the Proposition. □

Theorem 4 (Completeness of congruence projection)**.** *For any complete d*SHA *A with schema $\mathbf{S}$ the congruence projection $A^{cgr(\mathbf{S})}$ is sound and complete for subhedge projection wrt to $\mathbf{S}$.*

Proof. The soundness of congruence projection was shown in Theorem 2. For showing the completeness, let $h \in \mathbf{S}$ and v a prefix of $nw(h)$ that is irrelevant for subhedge projection for $\text{Ł}(A)$ wrt $\mathbf{S}$. Since $h \in \mathbf{S} = \text{Ł}(A[F/F_\mathbf{S}])$, A has a run on h which is unique since A is deterministic, and ends in $F_\mathbf{S}$ since $h \in \mathbf{S}$. Let $w \in \hat{\Sigma}^*$ such that $v \cdot w = nw(h)$.

Let $p \in \mathcal{Q}$ be the state to which v is mapped by the unique run of A on h. Let $p' \in acc^\Delta(p)$. By Lemma 23 and since v is subhedge irrelevant for $\text{Ł}(A)$ wrt $\mathbf{S}$, it follows that $(p, p') \in rel^A_\mathbf{S}(v)$.

Since $p' \in acc^\Delta(p)$, there exists a hedge h' such that $q \xrightarrow{h'} q'$ wrt Δ. Let $u = nw(h')$. Since A is complete, it has a run on $hdg(v \cdot u \cdot w)$. By determinism, this run maps v to p and $v \cdot u$ to p'. Note that it may end up outside $F_\mathbf{S}$ though. By Proposition 26, the existence of runs of A on $h = hdg(v \cdot w)$ and on $hdg(v \cdot u \cdot w)$ leading to p at v and p' at $u \cdot v$ and $(p, p') \in rel^A_\mathbf{S}(v)$ show that $p \approx p'$ where $(p, \approx)$ is the state of $A^{cgr(\mathbf{S})}$ at v. Hence $p \in Prj^\Delta(\approx)$, so that Lemma 21 shows that $(p, \approx)$ is a subhedge projection state of $A^{cgr(\mathbf{S})}$. □

We note that the completeness of automaton A in this theorem is really needed for completeness in general. Schema-completeness may not be enough, since all states p' accessible from p are to considered. This may lead out of schema $\mathbf{S}$ when adding the closing nested word suffix w. In our examples, schema-completeness was good enough though.

7.4. In-Memory Complexity

We next discuss the complexity of membership testing with complete subhedge projection based on the in-memory evaluation of the input hedge by the congruence projection of the input automaton.

Lemma 29. *The number of states of $A^{cgr(\mathbf{S})}$ is in $O(2^{(n+1)\ \log(n)})$ where $n = |\mathcal{Q}|$.*

Proof. The states of $A^{cgr(\mathbf{S})}$ are pairs of states of $\mathcal{Q}$ and equivalence relations in $\mathcal{E}_\mathcal{Q}$. The number of equivalence relations in $\mathcal{E}_\mathcal{Q}$ is Bell's number $B_n \leq n^n = 2^{n \log(n)}$ where:

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$$

and $B_0 = 1$. See https://en.wikipedia.org/wiki/Bell_number for more information. So, the maximal number of states of the congruence projection is $|Q^{cgr}| = |Q|\ |\mathcal{E}_\mathcal{Q}| \leq n\ 2^{n\ \log(n)} = 2^{(n+1)\ \log(n)}$. □

Corollary 5. *Let the signature Σ be fixed. For any complete d*SHA *A with schema $\mathbf{S} \subseteq \mathcal{H}_\Sigma$, the membership of any hedge $h \in \mathbf{S}$ can be tested in-memory in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(2^{2(n+1)\log(n)})$ where n is the number of states of A.*

Proof. Since A is complete, Thereom 4 shows that $A^{cgr(\mathbf{S})}$ is complete for subhedge projection for schema $\mathbf{S}$. The in-memory evaluator of $A^{cgr(\mathbf{S})}$ can be run on any input hedge $h \in \mathbf{S}$ in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(n^3 + |A^{cgr(\mathbf{S})}| + n^3|\mathcal{Q}^{cgr}|)$. Since the signature Σ is fixed, this is in time $O(|\mathcal{Q}^{cgr}|^2)$ which by Lemma 29 is in $O(2^{2(n+1)\log(n)})$. □

Instead of precomputing $A^{cgr(\mathbf{S})}$ from the inputs A and $F_\mathbf{S}$ statically, we can compute only the part needed of $A^{cgr(\mathbf{S})}$ for evaluating the input hedge h on-the-fly. We note that the number of transition rules of the needed part is bounded by $O(|h|)$ but may be way smaller. The preprocessing time then goes down to $O(|A|)$. Since the full automaton $A^{cgr(\mathbf{S})}$ may be huge in the worst case, this may reduce the overall processing time.

On the positive side, the overall computation time goes down to $O(n^3\ |h|)$ while it was exponential before due to the preprocessing. On the down side, some steps may no more be in constant time $O(1)$ but in time $O(n^3)$. The number of these steps may be exponential in the worst case. But still, on-the-fly computation is the much better method in practice.

It shoud also be notice that the overall time of the safe-no-change projection algorithm is $O(|A|\ |h|)$ and thus smaller than the congruence projection algorithm. This is since the time per step of the safe-no-change algorithm is only in $O(|A|)$. This is the price to be payed for becoming complete.

Example 30. *In order to see how the exponential worst case may happen, we consider a family of regular languages, for which the minimal left-to-right* DFA *is exponentially bigger than the minimal right-to-left* DFA*. The classical example languages with this property are $L_n = \Sigma^*.a.\Sigma^n$ where $n \in \mathbb{N}$ and $\Sigma = \{a, b\}$. Intuitively, a word in Σ^* belongs to L_n if and only its $n+1$-th letter from the end is equal to "a". The minimal left-to-right* DFA *for L_n has 2^{n+1} many states, since needs to memoize a window of $n+1$-letters. In contrast, its minimal right-to-left* DFA *has only $n+1$ states; in this direction, it is sufficient to memoize the distance from the end modulo $n+1$.*

*We next consider the schema $\boldsymbol{S} = \mu c.\ \langle (a+b)\cdot c\rangle^*$ which contains all hedges whose nodes carry a single label from $\Sigma = \{a, b\}$. We the consider the family of hedge languages $H_n \in \mathcal{H}_\Sigma$ with schema $\boldsymbol{S}$ such that the sequence of labels of some root-to-leave path of h_n belongs to L_n. Note that H_n can be recognized in a bottom-up manner by the d*SHA *A_n with $O(n+1)$ states, which simulates the minimal deterministic* DFA *of L_n on all paths of the input hedge. For an evaluator with subhedge projection the situation is different. When moving top-down, it needs to memoize the sequence of labels of the $n+1$-last ancestors, possibly filled with $b's$, and there are 2^{n+1} of such sequences. If for some leaf, its sequence starts with an "a" then the following subhedges with the following leaves can be projected away. As a consequence, there cannot be any d*SHA$^\downarrow$ *recognizing H_n that projects away all irrelevant subhedges with less than 2^{n+1} states. In particular, the size of $A_n^{cgr(\boldsymbol{S})}$ must be exponential in the size of A_n.*

8. Earliest Membership with Subhedge Projection

We next enhance our compilers for introducing subhedge projection states such that they can also detect certain membership or nonmembership. By combining both aspects orthogonally, we obtain an earliest dSHA with subhedge projection, that can be inmemory or in streaming mode. The streaming version will enhance the previous earliest membership tester for dSHAs from [21] with subtree projection, while adding schemas to this approach.

8.1. Earliest Membership

We start with a semantic characterization of when membership to and nonmembership to hedge languages are certain for prefixes of hedges visited by top-down and left-to-right processing. Essentially the same characterizations were presented in the context of stream processing in [21,22].

Definition 31. *Let $\boldsymbol{S} \subseteq \mathcal{H}_\Sigma$ be a hedge schema and $L \subseteq \boldsymbol{S}$ a hedge language satisfying this schema. A nested word prefix v is* certain for membership *in L with schema $\boldsymbol{S}$ if for all suffixes w of nested words such that $v \cdot w \in nw(\boldsymbol{S})$ it holds that $v \cdot w \in nw(L)$. It is called* certain for nonmembership *in L with schema $\boldsymbol{S}$ if for all suffixes w of nested words such that $v \cdot w \in nw(\boldsymbol{S})$ it holds that $v \cdot w \notin nw(L)$.*

We are now interested in SHA$^\downarrow$ that are able to detect certain membership and nonmembership syntactically.

Definition 32. *A d*SHA$^\downarrow$ *$\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ with schema $\boldsymbol{S} \subseteq \mathcal{H}_\Sigma$ is called* earliest for schema $\boldsymbol{S}$ *if there exists a state $sel \in \mathcal{Q}$ such that for all hedges $h \in \boldsymbol{S}$ the unique run R of $compl(A)$ on h satisfies:*

- *it goes to the state sel for exactly all those prefixes of $nw(h)$ that are certain for membership in $Ł(A)$ wrt $\boldsymbol{S}$, and*
- *it goes to the sink for all prefixes of $nw(h)$ that are certain for nonmembership in $Ł(A)$ with $\boldsymbol{S}$.*

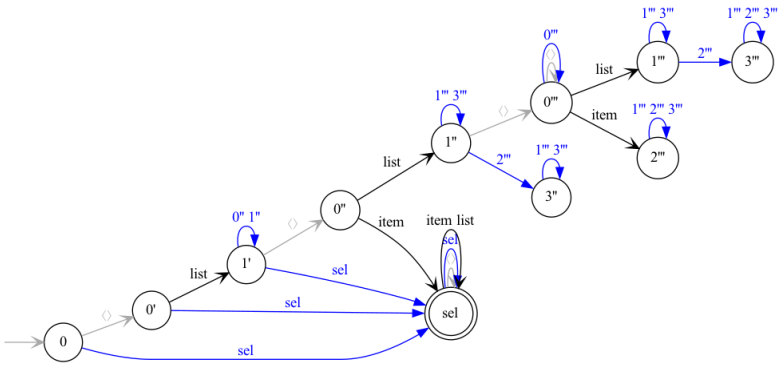


Figure 15. The earliest $d\text{SHA}^{\downarrow}$ $A_{e(\mathbf{S})}$ for the dSHA A in Fig. 1 with schema $\mathbf{S} = [\![doc]\!]$ for the XPath filter `[self::list][child::item]`.

For dSHA A that is schema-complete for schema $\mathbf{S} = \text{Ł}(A[F/F_{\mathbf{S}}])$, we can construct a $d\text{SHA}^{\downarrow}$ $A_{e(\mathbf{S})}$ that is earliest for $\mathbf{S}$, as shown in Appendix O. The idea is to adapt the earliest membership tester for deterministic SHAs from [21] so that it takes schema restrictions into account. Furthermore, we need to compile dSHAs to $d\text{SHA}^{\downarrow}$s rather than to dNWAs as used there, but this change is rather straightforward.

Given that the construction of $A_{e(\mathbf{S})}$ is not part of the core of the present paper, we only illustrate its outcome at our running example: For the dSHA A with schema $\mathbf{S} = [\![doc]\!]$ in Fig. 1 we obtain the $\text{SHA}^{\downarrow}$ $A_{e(\mathbf{S})}$ in Fig. 15. Note that on the prefix ⟨list⟨item it goes to state to *sel* showing that it is certain for membership. On prefix ⟨item it blocks showing that it is certain for nonmembership. However, on the prefix ⟨list⟨list it does not go into a subhedge projection state even though this prefix is subhedge irrelevant.

8.2. Adding Subhedge Projection

We now enhance any earliest membership for dSHAs with subhedge projection. The idea is to combine subhedge projection with earliest membership. So suppose that we are given a schema $\mathbf{S}$ and two automata A^{π} and A_e with schema $\mathbf{S}$ that recognize the same language up to the schema, so $\text{Ł}(A^{\pi}) \cap \text{Ł}(\mathbf{S}) = \text{Ł}(A_e) \cap \text{Ł}(\mathbf{S})$. Furthermore assume that A_e has a selection state *sel* indicating certain membership at the earliest possible prefix.

In order to obtain earliest membership with subhedge projection, we combine the two $\text{SHA}^{\downarrow}$s into a single $\text{SHA}^{\downarrow}$ $A^{\pi}_e = (\Sigma, Q^{\pi}_e, \Delta^{\pi}_e, I^{\pi}_e, F^{\pi}_e)$ basically running both automata in parallel, but under a shared control. The state set of A^{π}_e are as follows:

$$\begin{aligned} Q^{\pi}_e &= (Q^{\pi} \times (Q_e \setminus \{sel\})) \cup \{sel\} \\ I^{\pi}_e &= (I^{\pi} \times (I_e \setminus \{sel\})) \cup \{sel \mid sel \in I_e\} \\ F^{\pi}_e &= (F^{\pi} \times (F_e \setminus \{sel\})) \cup \{sel\} \end{aligned}$$

The transition rules in Δ^{π}_e are given by the in Fig. A21. Any run of A^{π}_e synchronizes parallel runs of A^{π} and A_e as follows: whenever A_e goes to *sel*, then A^{π}_e does so too, and whenever A^{π} goes into a subhedge projection state then it makes A_e jump over the subsequent subhedge.

For illustration, we reconsider the dSHA from Fig. 1. The earliest $d\text{SHA}^{\downarrow}$ with subhedge projection $A^{snc}_{e(\mathbf{S})}$ is given in Fig. 17. Here we combine the safe-no-change projection A^{snc} from Fig. 9 with the earliest membership tester $A_{e(\mathbf{S})}$ from Fig. 15. Their combination $A^{snc}_{e(\mathbf{S})}$ represents the pattern `[self::list][child::item]` with schema $[\![doc]\!]$ as nicely as one might ever have hoped for. After prefix ⟨list⟨item it detects certain membership, for the prefix ⟨list⟨list it detects that the subhedge is irrelevant, and after prefix ⟨item it detects certain nonmembership.

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta^\pi \quad r \xrightarrow{a} r' \text{ in } \Delta_e \quad q \notin P \quad r' \neq sel}{(q,r) \xrightarrow{a} (q',r') \text{ in } \Delta_e^\pi} \qquad \frac{q \xrightarrow{a} q' \text{ in } \Delta^\pi \quad r \xrightarrow{a} sel \text{ in } \Delta_e}{(q,r) \xrightarrow{a} sel \text{ in } \Delta_e^\pi}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta^\pi \quad r \xrightarrow{a} r' \text{ in } \Delta_e \quad q \in P \quad r' \neq sel}{(q,r) \xrightarrow{a} (q',r) \text{ in } \Delta_e^\pi}$$

$$\frac{q@p \to q' \text{ in } \Delta^\pi \quad r@s \to r' \text{ in } \Delta_e \quad r' \neq sel \quad q \notin P}{(q,r)@(p,s) \to (q',r') \text{ in } \Delta_e^\pi}$$

$$\frac{q@p \to q' \text{ in } \Delta^\pi \quad r@s \to r' \text{ in } \Delta_e \quad r' \neq sel \quad q \in P}{(q,r)@(p,s) \to (q',r) \text{ in } \Delta_e^\pi}$$

$$\frac{q@p \to q' \text{ in } \Delta^\pi \quad r@s \to r' \text{ in } \Delta_e \quad r' = sel}{(q,r)@(p,s) \to sel \text{ in } \Delta_e^\pi}$$

$$\frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta^\pi \quad r \xrightarrow{\langle\rangle} r' \text{ in } \Delta_e \quad q \notin P \quad r' \neq sel}{(q,r) \xrightarrow{\langle\rangle} (q',r') \text{ in } \Delta_e^\pi} \qquad \frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta^\pi \quad r \xrightarrow{\langle\rangle} sel \text{ in } \Delta_e}{(q,r) \xrightarrow{\langle\rangle} sel \text{ in } \Delta_e^\pi}$$

$$\frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta^\pi \quad r \xrightarrow{\langle\rangle} r' \text{ in } \Delta_e \quad q \in P \quad r' \neq sel}{(q,r) \xrightarrow{\langle\rangle} (q',r) \text{ in } \Delta_e^\pi}$$

$$\frac{a \in \Sigma}{sel \xrightarrow{a} sel \text{ in } \Delta_e^\pi} \qquad \frac{\mu \in \mathcal{Q}_e^\pi}{sel@\mu \to sel \text{ in } \Delta_e^\pi} \qquad \frac{\mu \in \mathcal{Q}_e^\pi}{\mu@sel \to sel \text{ in } \Delta_e^\pi} \qquad \frac{true}{sel \xrightarrow{\langle\rangle} sel \text{ in } \Delta_e^\pi}$$

Figure 16. The transition rules Δ_e^π inferred from those of the SHA$^\downarrow$s A^π with projection states P and A_e with selection state *sel*.

8.3. Soundness and Completeness

We next show that the soundness and completeness of a projection algorithm are preserved when combined with some earliest membership tester when assuming determinism.

Proposition 33. *Let A^π and A_e be d*SHA$^\downarrow$*s with the same schema $\mathbf{S}$ and the same language up to the schema, i.e. $\text{Ł}(A^\pi) \cap \text{Ł}(\mathbf{S}) = \text{Ł}(A_e) \cap \text{Ł}(\mathbf{S})$. The combination d*SHA$^\downarrow$ A_e^π *then has the same language up to schema $\mathbf{S}$ too. Furthermore, if A_e is earliest for $\mathbf{S}$ then A_e^π is earliest for $\mathbf{S}$ too and goes into some subhedge projection state whenever A^π does.*

Proof. If q is a subhedge projection state of A^π and r a state of A_e then (q,r) is a subhedge projection state of A_e^π. The evaluator for automaton A_e^π runs A^π and A_e in parallel on the input hedge, while skipping subhedges starting in subhedge projection states of A^π. These include the subhedges starting in a projection state when evaluating A_e^π on h. By Proposition 10 applied to A^π such subhedges are irrelevant for $\text{Ł}(A^\pi)$ with respect to $\mathbf{S}$ since A^π is assumed to be deterministic. Since $\text{Ł}(A^\pi) \cap \text{Ł}(\mathbf{S}) = \text{Ł}(A_e) \cap \text{Ł}(\mathbf{S})$, skipping such subhedge does not affect acceptance by A_e for hedges inside schema $\mathbf{S}$. Therefore the evaluator of A_e^π is earliest with respect to $\mathbf{S}$ too. □

The above proposition shows that if A^π is complete for subhedge projection then A_e^π is also complete for subhedge projection while being earliest in addition.

Theorem 6. *For any complete d*SHA*s A with schema $\mathbf{S}$, the d*SHA $A_{e(\mathbf{S})}^{cgr(\mathbf{S})}$ *is earliest and sound complete for subhedge projection for schema $\mathbf{S}$.*

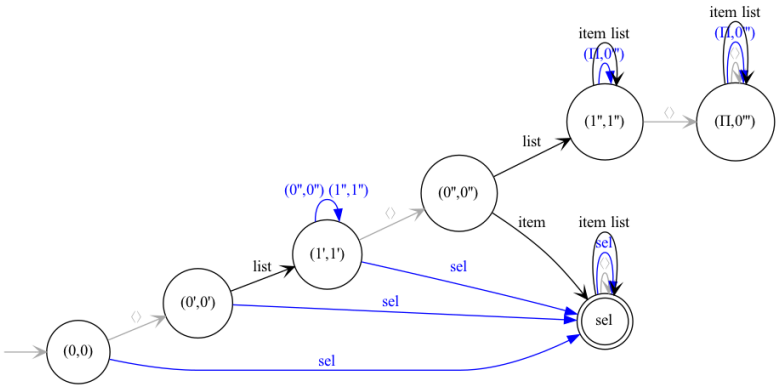


Figure 17. The earliest $d\text{SHA}^{\downarrow}$ $A^{snc}_{e(\mathbf{S})}$ with safe-no-change subhedge projection for the dSHA A in Fig. 1 with schema $\mathbf{S} = [\![doc]\!]$ for the XPath filter `[self::list][child::item]`.

Proof. The congruence projection $A^{cgr(\mathbf{S})}$ is sound by Proposition 2 and complete for subhedge projection wrt $\mathbf{S}$ by Theorem 4. The automaton $A_{e(\mathbf{S})}$ discussed in Section 8.1 is earliest for $\mathbf{S}$. So their combination $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ yields an earliest automaton with complete subhedge projection wrt $\mathbf{S}$ by Proposition 33. □

8.4. In-Memory Complexity

We next discuss the complexity of earliest membership testing with complete subhedge projection by an in-memory evaluator for $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$.

Lemma 34. *The number of states of* $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ *is in* $O(2^{(n+2)log(n)+n})$ *where* $n = |\mathcal{Q}|$.

Proof. By Lemma 29 the number of states of $A^{cgr(\mathbf{S})}$ is in $O(2^{(n+1)log(n)})$ where $n = |\mathcal{Q}|$. The number of states of $A_{e(\mathbf{S})}$ is in $O(n2^n)$ and thus in $O(2^{n+log(n)})$. The number of states of $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ is thus in $O(2^{(n+1)log(n)}2^{n+log(n)})$ which is in $O(2^{(n+2)log(n)+n})$ □

Corollary 7. *Let the signature* Σ *be fixed. For any complete d*SHA A *with schema* $\mathbf{S} \subseteq \mathcal{H}_\Sigma$*, earliest membership on a input hedge* $h \in \mathbf{S}$ *can be tested in-memory in time* $O(|1|)$ *per nonprojected node of* h *after a preprocessing time of* $O(2^{2(n+2)log(n)+2n})$ *where* n *is the number of states of* A.

Proof. Since A is complete, Thereom 6 shows that $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ is earliest and sound and complete for subhedge projection for schema $\mathbf{S}$. The in-memory evaluator of $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ can be run on any input hedge $h \in \mathbf{S}$ in time $O(|1|)$ per nonprojected node of h after a preprocessing time of $O(|A| + |A^{cgr(\mathbf{S})}_{e(\mathbf{S})}| + n^3|\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|)$. Since Σ is fixed the time is bounded by $O(|\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|^2)$. Lemma 34 shows that $|\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|$ is in $O(2^{(n+2)log(n)+n})$ □

As before, instead of precomputing $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ from the input dSHA A and $F_{\mathbf{S}}$ we can compute only the part needed for evaluating the input hedge h on-the-fly. If the needed part of $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ for evaluating h is small, the overall time and space go down considerably.

9. Streaming Algorithms

SHA$^{\downarrow}$s can be evaluated on nested words in streaming manner while yielding the same result as by in-memory evaluation. This permits us to obtain earliest streaming algorithms with complete subhedge projection for dSHAs.

9.1. Streaming Evaluators for Downward Hedge Automata

The property of having equivalent streaming and in-memory evaluators was already noticed for Neumann and Seidl's pushdown forest automata [26] and for nested word automata [11,15]. We here show how to transpose this result to downward hedge automata. What is more original is our approach to prove the correctness of streaming algorithms based on properties of in-memory evaluators.

We define the streaming evaluator of a SHA$^{\downarrow}$ $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ by a visibly pushdown machine. The set of configurations of this machine is $\mathcal{K} = \mathcal{Q} \times \mathcal{Q}^*$ containing pairs of states and stacks of states. The visibly pushdown machine provides for any word $v \in \hat{\Sigma}^*$ a streaming transition relation $\xrightarrow{v}^{\text{str}} \subseteq \mathcal{K} \times \mathcal{K}$ such that for all states $q, q' \in \mathcal{Q}$ and stacks $\sigma \in \mathcal{Q}^*$ and words $v, v' \in \hat{\Sigma}^*$:

$$\frac{\mathit{true}}{(q,\sigma) \xrightarrow{\varepsilon}^{\text{str}} (q,\sigma) \text{ wrt } \Delta} \qquad \frac{(q,\sigma) \xrightarrow{v}^{\text{str}} (q',\sigma) \qquad (q',\sigma) \xrightarrow{v'}^{\text{str}} (q'',\sigma) \text{ wrt } \Delta}{(q,\sigma) \xrightarrow{v\cdot v'}^{\text{str}} (q'',\sigma) \text{ wrt } \Delta}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a}^{\text{str}} q' \text{ wrt } \Delta} \qquad \frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta}{(q,\sigma) \xrightarrow{\langle}^{\text{str}} (q',\sigma\cdot q) \text{ wrt } \Delta} \qquad \frac{q@p \to q' \text{ in } \Delta}{(p,\sigma\cdot q) \xrightarrow{\rangle}^{\text{str}} (q',\sigma) \text{ wrt } \Delta}$$

We note that the same visibly pushdown machine can be obtained by compiling the SHA$^{\downarrow}$ to an NWA, whose streaming evaluator is defined by a visibly pushdown machine too (see Section 12.3 on related work).

The notion of partial runs of the in-memory evaluator is closely related to streaming runs. To see this, we first note that each partial run v wrt Δ naturally defines a stack as follows. Since v is a nested word prefix, there exist unique nested words $v_0, \ldots, v_n \in nw(\mathcal{H}_{\Sigma\cup\mathcal{Q}})$ such that:

$$v = v_0 \cdot \langle \cdot v_1 \cdot \langle \cdot \ldots \cdot \langle \cdot v_n$$

These nested words must be partial runs of Δ too, so there exist states $p_0, \ldots p_n, q_0, \ldots, q_n \in \mathcal{Q}$ such that v_i goes from p_i to q_i for all $0 \leq i \leq n$. We define the stack σ of v by $\sigma = q_0 \cdot \ldots \cdot q_{n-1}$ and say that v goes from p_0 to q_n and has stack σ. Note that the stack is empty if v is a nested word, since in this case $n = 0$ so that $v = v_0$ and $\sigma = \varepsilon$. For instance, the partial run $v_0 = 0 \cdot \langle \cdot 0 \cdot \mathit{list} \cdot 1 \cdot \langle \cdot 0 \cdot \mathit{item} \cdot 2$ has the stack $\sigma_0 = 0 \cdot 1$ while the partial run $v_1 = v_0 \cdot \rangle \cdot 3 \cdot \rangle \cdot 4$ has stack $\sigma_1 = \varepsilon$.

Lemma 35. *Consider a* SHA $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$*, states* $q, q' \in \mathcal{Q}$*, and a stack* $\sigma \in \mathcal{Q}^*$*. Let* v *be a prefix of some nested word in* $nw(\mathcal{H}_\Sigma)$*. Then there exists a partial run* r *wrt* Δ *from* q *to* q' *with stack* σ *such that* $proj_\Sigma(r) = v$ *iff* $(q,\varepsilon) \xrightarrow{v}^{str} (q',\sigma)$ *wrt* Δ*.*

Proof. By induction on the length of prefix v. □

This lemma implies that any in-memory transition of A on a hedge h from q to q' can be identified with some streaming transition of A on $nw(h)$ from (q,ε) to (q',ε).

Proposition 36. $q \xrightarrow{h} q'$ *wrt* $\Delta \Leftrightarrow (q,\varepsilon) \xrightarrow{nw(h)}^{str} (q',\varepsilon)$ *wrt* Δ*.*

Proof. If $q \xrightarrow{h} q'$ wrt Δ there exists a run R of h with Δ that starts with q and ends with q'. Consider the partial run $v = nw(R)$ on h. Since v is a nested word, its stack is empty. By

Lemma 35, we have $(q,\varepsilon) \xrightarrow{proj_\Sigma(v)}{}^{\text{str}} (q',\varepsilon)$ wrt Δ and $proj_\Sigma(v) = nw(h)$. For the inverse implication assume that $(q,\varepsilon) \xrightarrow{nw(h)}{}^{\text{str}} (q',\varepsilon)$ wrt Δ. Let $u = nw(h)$. By Lemma 35 there exists a partial run v of Δ from q to q' with stack ε. Then there exists a run R of Δ on h such that $v = nw(R)$. Hence, $q \xrightarrow{h} q'$ wrt Δ. □

For any deterministic $d\text{SHA}^{\downarrow}$ A, hedge h, and state q, the streaming transition on $nw(h)$ with respect to Δ starting with (q,ε) can be computed in time $O(1)$ per letter after a precomputation in time $O(|A|)$. So the overall computation time is in $O(|A| + |h|)$. The streaming memory needed to store a configuration is of size $O(depth(h) + |A|)$ since the size of the visible stack is bounded by the depth of the input hedge.

9.2. Adding Subhedge Projection

We next extend the streaming transition relation with subhedge projection, in analogy to what we did for the in-memory transition relation. We define a transition relation with subhedge projection $\xrightarrow{h}{}^{\text{str}}_{shp} \subseteq \mathcal{K} \times \mathcal{K}$ with respect to Δ such that for all words $v, v' \in \hat{\Sigma}^*$, letters $a \in \Sigma$, states $p, q, q', q'' \in \mathcal{Q}$ and stacks $\sigma, \sigma', \sigma'' \in \mathcal{Q}^*$:

$$\frac{q \in \mathcal{Q}^\Delta_{shp} \qquad v \in nw(\mathcal{H}_\Sigma)}{(q,\sigma) \xrightarrow{v}{}^{\text{str}}_{shp} (q,\sigma) \text{ wrt } \Delta}$$

$$\frac{q \notin \mathcal{Q}^\Delta_{shp} \qquad q \xrightarrow{a} q' \text{ in } \Delta}{(q,\sigma) \xrightarrow{a}{}^{\text{str}}_{shp} (q',\sigma) \text{ wrt } \Delta} \qquad \frac{q \notin \mathcal{Q}^\Delta_{shp}}{(q,\sigma) \xrightarrow{\varepsilon}{}^{\text{str}}_{shp} (q,\sigma) \text{ wrt } \Delta}$$

$$\frac{q \notin \mathcal{Q}^\Delta_{shp} \qquad (q,\sigma) \xrightarrow{v}{}^{\text{str}}_{shp} (q',\sigma') \text{ wrt } \Delta \qquad (q',\sigma') \xrightarrow{v'}{}^{\text{str}}_{shp} (q'',\sigma'') \text{ wrt } \Delta}{(q,\sigma) \xrightarrow{v \cdot v'}{}^{\text{str}}_{shp} (q'',\sigma'') \text{ wrt } \Delta}$$

$$\frac{q \notin \mathcal{Q}^\Delta_{shp} \qquad q \xrightarrow{\langle\rangle} q' \text{ in } \Delta}{(q,\sigma) \xrightarrow{\langle}{}^{\text{str}}_{shp} (q',\sigma \cdot q) \text{ wrt } \Delta} \qquad \frac{p \notin \mathcal{Q}^\Delta_{shp} \qquad q@p \to q' \text{ in } \Delta}{(p,\sigma \cdot q) \xrightarrow{\rangle}{}^{\text{str}}_{shp} (q',\sigma) \text{ wrt } \Delta}$$

The projecting transition relation stays in a configuration with a subhedge projection state until the end of the current subhedge is reached. This is correct since a subhedge projection state cannot be changed anyway.

Proposition 37. *For any* $\text{SHA}^{\downarrow}$ $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$*, states* $q, q' \in \mathcal{Q}$*, nested word* v *on which* Δ *has no bocking partial run starting from* q*:*

$$(q,\varepsilon) \xrightarrow{v}{}^{str} (q',\varepsilon) \text{ wrt } \Delta \Leftrightarrow (q,\varepsilon) \xrightarrow{v}{}^{str}_{shp} (q',\varepsilon) \text{ wrt } \Delta$$

Proof. Let $h \in \mathcal{H}_\Sigma$ be the hedge such that $v = nw(h)$. The proof is by induction on the size of v. For the implication from the left to the right we assume that $(q,\varepsilon) \xrightarrow{v}{}^{\text{str}} (q',\varepsilon)$ wrt Δ. Since $v = nw(h)$, Proposition 36 proves $q \xrightarrow{h}{}^{\text{str}} q'$ wrt Δ.

Case $q \in \mathcal{Q}^\Delta_{shp}$**.** By Lemma 9, it follows from $q \xrightarrow{h}{}^{\text{str}} q'$ wrt Δ that $q = q'$. We can then apply the following rule since v is a nested word:

$$\frac{q \in \mathcal{Q}^\Delta_{shp} \qquad v = nw(\mathcal{H}_\Sigma)}{(q,\varepsilon) \xrightarrow{v}{}^{\text{str}}_{shp} (q,\varepsilon) \text{ wrt } \Delta}$$

Hence $(q,\varepsilon) \xrightarrow{v}{}^{\mathrm{str}}_{shp} (q,\varepsilon)$ wrt Δ.

Case $q \notin \mathcal{Q}^{\Delta}_{shp}$**.** We distinguish all possible forms of hedge h.

Subcase $h = h' \cdot h''$**.** Let $v' = nw(h')$ and $v'' = nw(h'')$ so $v = v' \cdot v'' = nw(h)$. The following rule must have been applied:

$$\frac{(q,\varepsilon) \xrightarrow{v'}{}^{\mathrm{str}} (p,\sigma) \text{ wrt } \Delta \qquad (p,\sigma) \xrightarrow{v''}{}^{\mathrm{str}} (q',\varepsilon) \text{ wrt } \Delta}{(q,\varepsilon) \xrightarrow{v' \cdot v''}{}^{\mathrm{str}} (q',\varepsilon) \text{ wrt } \Delta}$$

Since v' is a nested word, it follows that $\sigma = \varepsilon$. By induction hypothesis applied to the nested words v' and v'' we get $(q,\varepsilon) \xrightarrow{v'}{}^{\mathrm{str}}_{shp} (p,\varepsilon)$ wrt Δ and $(p,\varepsilon) \xrightarrow{v''}{}^{\mathrm{str}}_{shp} (q',\varepsilon)$ wrt Δ. Hence, we can apply the following rule:

$$\frac{q \notin \mathcal{Q}^{\Delta}_{shp} \quad (q,\varepsilon) \xrightarrow{v'}{}^{\mathrm{str}}_{shp} (p,\varepsilon) \text{ wrt } \Delta \quad (p,\varepsilon) \xrightarrow{v''}{}^{\mathrm{str}}_{shp} (q',\varepsilon) \text{ wrt } \Delta}{(q,\varepsilon) \xrightarrow{v' \cdot v''}{}^{\mathrm{str}}_{shp} (q',\varepsilon) \text{ wrt } \Delta}$$

This shows $(q,\varepsilon) \xrightarrow{v}{}^{\mathrm{str}}_{shp} (q',\varepsilon)$ as required.

Subcases $h = \langle h' \rangle$ **or** $h = a$ **or** $h = \varepsilon$**.** Straightforward.

For the implication from the right to the left we assume that $(q,\varepsilon) \xrightarrow{v}{}^{\mathrm{str}}_{shp} (q',\varepsilon)$ wrt Δ.

Case $q \in \mathcal{Q}^{\Delta}_{shp}$**.** Then the following rule must have been applied with $q = q'$:

$$\frac{q \in \mathcal{Q}^{\Delta}_{shp} \qquad v = nw(\mathcal{H}_{\Sigma})}{(q,\varepsilon) \xrightarrow{v}{}^{\mathrm{str}}_{shp} (q,\varepsilon) \text{ wrt } \Delta}$$

Since we assume that v does not have any blocking partial runs with Δ, there exists a partial run on v with Δ that starts with q. Since v is a nested word, any partial run on v is the nested word of some run, so there exists a run on the hedge h that starts with q. This shows the existence of some state p such that $q \xrightarrow{h} p$ wrt Δ. Since $q \in \mathcal{Q}^{\Delta}_{shp}$, Lemma 9 shows $q = p$ and thus $q \xrightarrow{h} q$ wrt Δ. Proposition 36 then proves that $(q,\varepsilon) \xrightarrow{nw(h)}{}^{\mathrm{str}} (q,\varepsilon)$ wrt Δ, which is equal to $(q,\varepsilon) \xrightarrow{v}{}^{\mathrm{str}} (q',\varepsilon)$ wrt Δ as required.

Case $q \notin \mathcal{Q}^{\Delta}_{shp}$**.** We distinguish all possible forms of the hedge h.

Subcase $h = h' \cdot h''$**.** Let $v' = nw(h')$ and $v'' = nw(h'')$ so $v = v' \cdot v'' = nw(h)$. The following rule must have been applied:

$$\frac{q \notin \mathcal{Q}^{\Delta}_{shp} \qquad (q,\varepsilon) \xrightarrow{v'}{}^{\mathrm{str}}_{shp} (p,\sigma) \text{ wrt } \Delta \qquad (p,\sigma) \xrightarrow{v''}{}^{\mathrm{str}}_{shp} (q',\varepsilon) \text{ wrt } \Delta}{(q,\varepsilon) \xrightarrow{v' \cdot v''}{}^{\mathrm{str}}_{shp} (q',\varepsilon) \text{ wrt } \Delta}$$

Since v' is a nested word, it follows that $\sigma = \varepsilon$ too. By induction hypothesis applied to the nested words v' and v'' we get $(q,\varepsilon) \xrightarrow{v'}{}^{\mathrm{str}} (p,\varepsilon)$ wrt Δ and $(p,\varepsilon) \xrightarrow{v''}{}^{\mathrm{str}} (q',\varepsilon)$ wrt Δ. Hence, we can apply the following rule:

$$\frac{(q,\varepsilon) \xrightarrow{v'}{}^{\mathrm{str}} (p,\varepsilon) \text{ wrt } \Delta \quad (p,\varepsilon) \xrightarrow{v''}{}^{\mathrm{str}} (q',\varepsilon) \text{ wrt } \Delta}{(q,\varepsilon) \xrightarrow{v' \cdot v''}{}^{\mathrm{str}} (q',\varepsilon) \text{ wrt } \Delta}$$

This shows $(q,\varepsilon) \xrightarrow{v}^{\text{str}} (q',\varepsilon)$ as required.

Subcases $h = \langle h' \rangle$ or $h = a$ or $h = \varepsilon$. Straightforward.

This ends the proofs of both directions. □

A streaming evaluator with subhedge projection for deterministic SHA$^{\downarrow}$s A on hedges h can thus be obtained by computing the streaming transition relation with subhedge projection for A of $nw(h)$ starting with the initial configuration. This costs time at most $O(1)$ per letter of $nw(h)$, i.e. constant time per event of the stream.

9.3. Streaming Complexity of Earliest Membership with Subhedge Projection

By using streaming evaluators of SHA$^{\downarrow}$s we can test membership earliest and with subhedge projection in streaming mode. We obtain the following complexity result by using the SHA$^{\downarrow}$ $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ for a given dSHA A with schema $\mathbf{S} = Ł(A[F/F_{\mathbf{S}}])$.

Corollary 8. *For any complete d*SHA*s A with schema $\mathbf{S}$, earliest membership with complete subhedge projection wrt $\mathbf{S}$ can be tested in streaming mode for any hedge $h \in \mathbf{S}$ in time $O(|1|)$ per nonprojected letter of $nw(h)$ after a preprocessing time of $O(2^{2(n+2)log(n)+2n})$ where $n = |\mathcal{Q}|$ is the number of states of A. The space required is in $O(depth(h) + 2^{2(n+2)log(n)+2n})$.*

Proof. By Theorem 6 the dSHA$^{\downarrow}$ $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ is earliest and sound and complete for subhedge projection. By Propositions 36 and 37 we can thus obtain a streaming membership tester with subhedge projection for a hedge $h \in \mathbf{S}$ by running the streaming evaluator with subhedge projection of $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ on the nested word $nw(h)$. Since this SHA$^{\downarrow}$ is deterministic, the streaming evaluation can be done in time $O(|1|)$ per letter of $nw(h)$ after a preprocessing time of $O(|A| + |A^{cgr(\mathbf{S})}_{e(\mathbf{S})}| + n^3|\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|)$. Since Σ is fixed, the preprocessing time is also in $O(|\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|^2)$. The space required is in $O(depth(h) + |A^{cgr(\mathbf{S})}_{e(\mathbf{S})}|)$ and thus in $O(depth(h) + |\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|^2)$. Furthermore by Lemma 34, $|\mathcal{Q}^{cgr(\mathbf{S})}_{e(\mathbf{S})}|$ is in $O(2^{(n+2)log(n)+n})$. Since $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$ is earliest and complete for subhedge projection, so is its streaming evaluator. □

10. Monadic Queries

We next consider the problem of how to answer monadic regular queries on hedges in a earliest manner and with subhedge projection. This requires to select all position selected by regular XPath queries such as `child::list[child::item]`, which is more complicated than verifying whether the XPath filter `[child::list/child::item]` matches at the root of first subtree of the hedge: the answer set needs to be constructed too.

Monadic queries on hedges in $\mathcal{H}_\Sigma$ can be defined by nested regular expressions in $nRegExp_{\Sigma\cup\{x\}}$ where x is a fresh special symbol standing for the position to be selected. The regular XPath query `self::list[child::item]` corresponds to the following nested regular expression in $nRegExp_{\Sigma\cup\{x\}}$ where $\top = \mu b.(+_{a\in\Sigma} a + \langle b \rangle)^*$:

$$\langle \text{list} \cdot x \cdot \top \cdot \langle \cdot \text{item} \cdot \top \cdot \rangle \cdot \top \rangle \cdot \top$$

Note that the nested regular expression for the XPath filter `[self::list/child::item]` in $nRegExp_\Sigma$ can be obtained by removing the x.

The earliest query answering algorithm from [21] can deal with regular monadic queries, not only with earliest membership. For this, the nested regular expression has to be compiled to a dSHA A with signature $\Sigma \cup \{x\}$ in the first step. The schema has to be changed from $[\![doc]\!]$ to $\mathbf{S} = [\![doc_x]\!] \cap one_x$ where $doc_x \in nRegExp_{\Sigma\cup\{x\}}$ is like doc but over the signature $\Sigma \cup \{x\}$ instead of Σ and one_x is the set of all hedge in $nRegExp_{\Sigma\cup\{x\}}$ that contain exactly one occurrence of x. One can then compute the earliest SHA$^{\downarrow}$ with respect to the new schema $A_{e(\mathbf{S})}$. For earliest query answering on a input hedge $h \in [\![doc]\!]$, the idea

is that the variable x needs to be inserted into h at all possible positions, so that the SHA$^{\downarrow}$ $A_{e(\mathbf{S})}$ can be run on all extensions $h' \in \mathbf{S}$ of h. How to run an dSHA$^{\downarrow}$ with signature $\Sigma \cup \{x\}$ on all extensions $h' \in \mathbf{S}$ of h efficiently is the main topic of [21], just that dNWAs are used there instead. As discussed earlier, this can be done either in in-memory or in streaming mode.

In order to add safe-no-change projection to the earliest query answering algorithm, we have to run the automaton $A^{snc}_{e(\mathbf{S})}$ and for adding congruence projection, we have to run the automaton $A^{cgr(\mathbf{S})}_{e(\mathbf{S})}$. Fortunately, the soundness and completeness theorems for earliest membership testing with subhedge projection do only affect the constructions of these SHA$^{\downarrow}$s, but not how they are to be evaluated.

11. Experiments with Streaming XPath Evaluation

We integrated safe-no-change projection into the streaming earliest monadic query answering tool AStream [21]. It is implemented in Scala while computing safety with ABC Datalog. We did not yet implement congruence projection though, basically since it may require higher computational costs.

In order to benchmark AStream 2.01 with safe-no-change projection for efficiency, and to compare it to AStream 1.01 without projection, we considered the regular XPath queries from the XPathMark [27] $A1 - A8$, see Table 1. We used the deterministic SHAs for all these XPath queries constructed by the compiler from [8]. These were evaluated on XML documents of variable size created by the XPathMark generator. We did further experiments on a sub-corpus of 79 regular XPath queries extracted by Lick and Schmitz from real-world XSLT and XQUERY programs, for which dSHAs are available [7]. These experiments confirm the results presented here, so we don't describe them in detail.

The XPath queries of the XPathMark without descendant axis are A1,A4 and A6-A8. The evaluation time on these queries a 1.2 GB document are reduced between $88 - 97\%$. In average, it is 92.5%, so the overall time is divided by 12. While the parsing time remains unchanged the gain on the automaton evaluation time is proportional to the percentage of subhedge projection for the respective query. This remains true for the other queries with the descendant axis, just that the projection percentage is much lower.

Finally, we compared AStream with for QuiXPath [23], the best previous streaming tool that can answer A1-A8 in an earliest manner. QuiXPath compiles regular XPath queries to possibly nondeterministic early NWAs, and evaluates them with subtree and descendant projection [4]. QuiXPath is not generally earliest though. On the queries without descendant axis, AStream 1.01 without projection is by a factor of 60 slower than QuiXPath [21]. With subhedge projection in version 2.01, the overhead goes down to a factor of $5 = 60/12$. So our current implementation is close to becoming competitive with the best existing streaming tool while guaranteeing earliest query answering in addition.

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`

Table 2. Timings in seconds for XPathMark queries that have child axis exclusively with QuiXPath and the two versions of our tool.

	QuiXPath	Astream 1.01 nonproj.	Astream 2.01 proj.	Gain: Astream2.01 vs Astream1.01	Factor: QuiXPath/ Astream2.01
A1	11	644.67	72.83	88.7%	*6.62
A4	11.6	723.03	78.5	89.14%	*6.77
A6	10.7	780.78	65.26	91.64%	*6.1
A7	8.6	601.26	24.64	95.9 %	*2.87
A8	8.8	801.96	24.85	96.9%	*2.82
average	10.14	710.34	53.2	92.45%	*5.036

Table 3. Timings in seconds for XPathMark queries that have descendant axis with QuiXPath and the two versions of our tool.

	QuiXPath	Astream 1.01 nonproj.	Astream 2.01 proj.	Gain: Astream2.01 vs Astream1.01	Factor: QuiXPath/ Astream2.01
A2	11.4	569.0112	664.6	-16.8%	*58.3
A3	11.5	673.716	666.124	1.13%	*57.92
A5	12	593.4648	77.785	86.89%	*6.48

12. Related Work

A large number of alternative automata notions for trees, forest and hedges were proposed in the literature. We compare them to SHAs and SHA$^{\downarrow}$s in what follows, and discuss the implications.

12.1. SHAs *versus Tree, Forest, and Hedge Automata*

Standard Hedge Automata. Standard hedge automata [9,10,33,34] operate on labeled hedges in a bottom-up manner similarly to SHAs. Also they have the same expressiveness, but horizontal languages are specified differently, leading to a problematic notion of determinism [35]. For this reason unique minimization fails for deterministic standard hedge automata. Still, they have the same expressiveness as SHAs when restricted to labeled hedges.

Standard Tree Automata and SHA Minimization. Syntactically, any SHA A is a standard tree automaton over the following ranked signature:

- a unary symbol for any letter $a \in \Sigma$,
- a binary symbol @, and
- a constant $\langle\rangle$.

Semantically, however, there is no perfect general correspondence. Only for subclasses a perfect correspondence can be made up via binary encoding. This was shown in [8] for the subclass of dSHAs with $I = \langle\rangle^{\Delta}$. In [36], it could also be shown for the subclass of multi-module dSHAs. This leads to unique minimization results for both subclasses of deterministic SHAs.

Bojanczyk's Forest Automata. Standard hedge automata also have the same expressiveness as Bojanczyk's forest automata (see Section 3.3 of [37]). These are based on the idea of transition monoids, rather than on the idea of transition rules such as SHAs or finite state

automata or Neuman's and Seidl's forest automata. As a consequence, however, there is an exponential difference in succinctness between SHAs and Bojanczyk's forest automata [36].

12.2. SHA$^{\downarrow}$s versus SHAs

We start explaining why SHA$^{\downarrow}$'s have the same expressiveness as SHAs. Basically it is the same reason for why NWAs have the same expressiveness as SHAs (see e.g. [8]). For the easier direction, we argued already that any SHA $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ can converted to a SHA$^{\downarrow}$ $down(A)$.

Downward Elimination. Conversely, we can convert any dSHA$^{\downarrow}$ A into an equivalent SHA $elim^{\downarrow}(A)$ while possibly introducing nondeterminism as follows:

$$I^{elim^{\downarrow}} = I \times \mathcal{Q} \qquad F^{elim^{\downarrow}} = F \times \mathcal{Q} \qquad \frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta}{\xrightarrow{\langle\rangle} (q,q') \text{ in } \Delta^{elim^{\downarrow}}}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \qquad r \in \mathcal{Q}}{(r,q) \xrightarrow{a} (r,q') \text{ in } \Delta^{elim^{\downarrow}}} \qquad \frac{q@p \to q' \text{ in } \Delta \qquad r \in \mathcal{Q}}{(r,q)@(q,p) \to (r,q') \text{ in } \Delta^{elim^{\downarrow}}}$$

Proposition 38. $Ł(A) = Ł(elim^{\downarrow}(A))$.

The construction is analogous to the conversion of NWAs to SHAs [8] or to hedge automata [26]. The correctness proofs for these compilers are standard.

Unique Minimization. Unique minimization fails for dSHA$^{\downarrow}$s as usual for deterministic multiway automata. This even happens for deterministic two-way finite state automata on words. In contrast, the subclass of dSHAs with the restriction that the initial state and the tree initial state are the same enjoys unique minimization [8].

Neuman and Seidl's Pushdown Forest Automata. Standard hedge automata are also known to have the same expressiveness than Neumann and Seidl's pushdown forest automata [25]. The latter can be identified with SHA$^{\downarrow}$s for labeled hedges. What is needed to map pushdown forest automata to standard hedge automata is downward elimination, as for mapping SHA$^{\downarrow}$s to SHAs.

12.3. SHA$^{\downarrow}$s versus NWAs

The streaming evaluator for SHA$^{\downarrow}$s via a visibly pushdown machine can also be obtained by compiling a SHA$^{\downarrow}$ to a nested word automaton (NWA) [11], whose streaming evaluator is given by a visibly pushdown machine [12], previously known as *input driven automata* [13–15].

Definition 39. *A nested word automata (*NWA*) is a tuple* $(\Sigma, \mathcal{Q}, \Gamma, \Delta, I, F)$*, where* Σ, Γ *and* $\mathcal{Q}$ *are sets,* $I, F \subseteq \mathcal{Q}$*, and* $\Delta = ((a^{\Delta})_{a\in\Sigma}, \langle^{\Delta}, \rangle^{\Delta})$ *contains relations:* $a^{\Delta} \subseteq \mathcal{Q} \times \mathcal{Q}$*,* $\langle^{\Delta} \subseteq \mathcal{Q} \times (\Gamma \times \mathcal{Q})$ *and* $\rangle^{\Delta} \subseteq \mathcal{Q} \times \Gamma \times \mathcal{Q}$*. A* NWA *is deterministic or equivalently a d*NWA *if* I *contains at most one element and all above relations are partial functions.*

The elements of Γ are called stack symbols. The transition rules in Δ again have three forms: Internal rules $q \xrightarrow{a} q'$ in Δ as for SHAs, opening rules $q \xrightarrow{\langle\downarrow\gamma} q'$ in Δ if $\langle^{\Delta}(q) = (q', \gamma)$ and closing rules $q \xrightarrow{\rangle\uparrow\gamma} q'$ in Δ if $\rangle^{\Delta}(q, \gamma) = q'$.

Streaming evaluators for NWAs. The streaming evaluator for NWAs can be seen as pushdown machines for evaluating the nested words of hedges in streaming manner. A configuration of the pushdown machine is a pair in $\mathcal{K} = \mathcal{Q} \times \Gamma^*$ containing a state and a stack. For

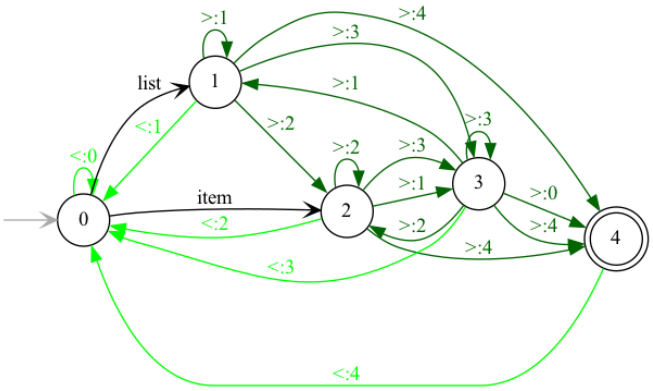


Figure 18. The dNWA A^{nwa} for the $d\text{SHA}^{\downarrow}$ A in Fig. 7 obtained by applying the *down* operator to the dSHA in Fig. 1 for the filter `[self::list][child::item]`.

any word $v \in \hat{\Sigma}^*$ we define a streaming transition relation $\xrightarrow{v}^{\text{str}} \subseteq \mathcal{K} \times \mathcal{K}$ such that for all $q, q' \in \mathcal{Q}$ and $\sigma \in \Gamma^*$:

$$\frac{\mathit{true}}{(q,\sigma) \xrightarrow{\varepsilon}^{\text{str}} (q,\sigma) \text{ wrt } \Delta} \qquad \frac{(q,\sigma) \xrightarrow{v}^{\text{str}} (q',\sigma) \qquad (q',\sigma) \xrightarrow{v'}^{\text{str}} (q'',\sigma) \text{ wrt } \Delta}{(q,\sigma) \xrightarrow{v \cdot v'}^{\text{str}} (q'',\sigma) \text{ wrt } \Delta}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a}^{\text{str}} q' \text{ wrt } \Delta} \qquad \frac{q \xrightarrow{\langle\downarrow\gamma} q' \text{ in } \Delta}{(q,\sigma) \xrightarrow{\langle}^{\text{str}} (q',\sigma) \text{ wrt } \Delta} \qquad \frac{q \xrightarrow{\rangle\uparrow\gamma} q' \text{ in } \Delta}{(q,\sigma \cdot \gamma) \xrightarrow{\rangle}^{\text{str}} (q',\sigma) \text{ wrt } \Delta}$$

From SHA$^{\downarrow}$s to NWAs. For any $\text{SHA}^{\downarrow}$ $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ we can define the NWA $A^{nwa} = (\Sigma, \mathcal{Q}, \Gamma, \Delta^{nwa}, I^{nwa}, F^{nwa})$ while preserving determinism, such that $\Gamma = \mathcal{Q}$, while Δ^{nwa} contains for all $a \in \Sigma$ and $q, p \in \mathcal{Q}$ the transition rules:

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta}{q \xrightarrow{a} q' \text{ in } \Delta^{nwa}} \qquad \frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta}{q \xrightarrow{\langle\downarrow q} q' \text{ in } \Delta^{nwa}} \qquad \frac{q@p \to q' \text{ in } \Delta}{p \xrightarrow{\rangle\uparrow q} q' \text{ in } \Delta^{nwa}}$$

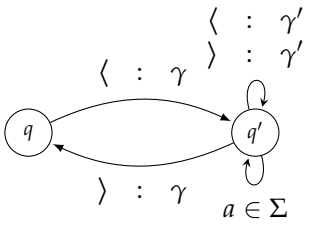
For instance, the dNWA A^{nwa} of the $d\text{SHA}^{\downarrow}$ A in Fig. 7 obtained from the dSHA for the `[self::list][child::item]` from the introduction is given in Fig. 18. And the dNWA A^{nwa} of the $d\text{SHA}^{\downarrow}$ A in Fig. 17 is given in Fig. 19.

Lemma 40. $Ł(A^{nwa}) = Ł(A)$.

The runs of $\text{SHA}^{\downarrow}$s A and NWAs A^{nwa} can be identified.

Projection for NWAs. Projecting evaluators for NWAs were proposed in the context of projecting NWAs [4]. They are based on the following notion of states of irrelevant subtrees for NWAs.

Definition 41 (Variant of Definition 3 of [4])**.** *We call a state q of an* NWA a state of irrelevant subtrees *if there exist two different stack symbols γ, γ' and a state q' such that the transitions shown on the right exist, but no further opening transitions with γ, no further transitions with γ', and no further closing transition in q popping γ. In this case, we write $q \in i\text{-tree}_{\Sigma \setminus \emptyset}$.*



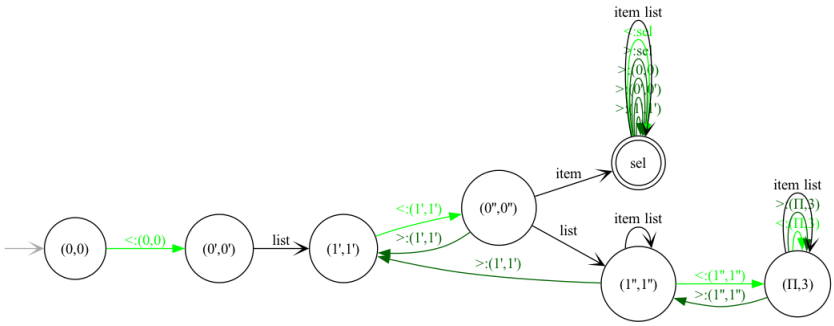


Figure 19. The dNWA $(A^{snc}_{e([\![doc]\!])})^{nwa}$ for the earliest dSHA$^{\downarrow}$ with safe-no-change projection in Fig. 17.

Lemma 42. *Any subhedge projection state of a complete* SHA$^{\downarrow}$ *A is a state of irrelevant subtrees of* A^{nwa}.

It was then shown in [4] how to map an NWA with a subset $\mathcal{Q}^{\Delta}_{shp}$ of states of irrelevant subtrees to a projecting NWA, whose streaming semantics yields a streaming evaluator with subhedge projection. The presentation of subhedge projection for SHA$^{\downarrow}$s without passing via NWAs yields the same result in a more direct manner.

It should also be noticed that projecting NWAs support descendant projection beside of subhedge projection. For SHAs, this is left to future research.

13. Conclusion and Future Work

We introduced the notion of irrelevant subhedges for membership to hedge language under schema restrictions. We developed algorithms with subhedge projection that jump over irrelevant subhedges for testing membership to the regular hedge languages defined by dSHAs. All our algorithms can be run in in-memory mode and in streaming mode. The difficulty was how to push the needed finite state information for subtree projection top-down given that SHAs operate bottom-up. We solved it based on compilers from SHAs to downward SHAs.

This first compiler propagates safety information about non-changing states. The second compiler, which we prove to be complete for subhedge projection, propagates equivalence classes. We then combined subhedge projection with earliest membership testing and combine it earliest monadic query answering. We confirmed the usefulness of our safe-no-change subhedge projection algorithm for SHAs experimentally: we showed that it can indeed speed up the the AStream tool, for answering algorithm regular XPath queries on XML streams. This way it becomes close to competitive in time with the best existing streaming tools for regular XPath queries without recursive axis.

A remaining open theoretical and practical question is how to obtain descendant projection for SHAs, as needed to speed up the evaluation of XPath queries with descendant axis. On the implementation and experimental side, it would be nice to investigate the in-memory version of our subhedge algorithms too. Furthermore, an experimental comparison between safe-no-state and congruence projection would be nice to have. And finally, it would be nice to optimize streaming earliest-query answering tools further until they become fully competitive with the best streaming tools in all case, while being better for queries where the other tools are not earliest.

References

1. Marian, A.; Siméon, J. Projecting XML Documents. In Proceedings of the VLDB, 2003, pp. 213–224.

2. Frisch, A. Regular Tree Language Recognition with Static Information. In Proceedings of the Exploring New Frontiers of Theoretical Informatics, IFIP 18th World Computer Congress, TCS 3rd International Conference on Theoretical Computer Science, 2004, pp. 661–674.
3. Maneth, S.; Nguyen, K. XPath Whole Query Optimization. *VLPB Journal* **2010**, *3*, 882–893.
4. Sebastian, T.; Niehren, J. Projection for Nested Word Automata Speeds up XPath Evaluation on XML Streams. In Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Harrachov, Czech Republic, 2016.
5. Kay, M. The saxon XSLT and XQuery processor, 2004. https://www.saxonica.com.
6. Gienieczko, M.; Murlak, F.; Paperman, C. Supporting Descendants in SIMD-Accelerated JSONPath. In Proceedings of the Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1, ASPLOS 2024. ACM, 2024, pp. 15–29.
7. Al Serhali, A.; Niehren, J. A Benchmark Collection of Deterministic Automata for XPath Queries. In Proceedings of the XML Prague 2022, Prague, Czech Republic, 2022.
8. Niehren, J.; Sakho, M. Determinization and Minimization of Automata for Nested Words Revisited. *Algorithms* **2021**, *14*, 68. https://doi.org/10.3390/a14030068.
9. Comon, H.; Dauchet, M.; Gilleron, R.; Löding, C.; Jacquemard, F.; Lugiez, D.; Tison, S.; Tommasi, M. Tree Automata Techniques and Applications. Available online since 1997: http://tata.gforge.inria.fr, 2007.
10. Thatcher, J.W. Characterizing derivation trees of context-free grammars through a generalization of automata theory. *Journal of Computer and System Science* **1967**, *1*, 317–322. https://doi.org/10.1016/S0022-0000(67)80022-9.
11. Alur, R. Marrying Words and Trees. In Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. ACM-Press, 2007, pp. 233–242.
12. Alur, R.; Madhusudan, P. Visibly pushdown languages. In Proceedings of the Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004; Babai, L., Ed. ACM, 2004, pp. 202–211. https://doi.org/10.1145/1007352.1007390.
13. Mehlhorn, K. Pebbling Moutain Ranges and its Application of DCFL-Recognition. In Proceedings of the Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings; de Bakker, J.W.; van Leeuwen, J., Eds. Springer, 1980, Vol. 85, *Lecture Notes in Computer Science*, pp. 422–435. https://doi.org/10.1007/3-540-10003-2_89.
14. von Braunmühl, B.; Verbeek, R. Input Driven Languages are Recognized in log n Space. In *Topics in the Theory of Computation*; Karplnski, M.; van Leeuwen, J., Eds.; North-Holland, 1985; Vol. 102, *North-Holland Mathematics Studies*, pp. 1 – 19. https://doi.org/https://doi.org/10.1016/S0304-0208(08)73072-X.
15. Okhotin, A.; Salomaa, K. Complexity of input-driven pushdown automata. *SIGACT News* **2014**, *45*, 47–67. https://doi.org/10.1145/2636805.2636821.
16. Niehren, J.; Sakho, M.; Al Serhali, A. Schema-Based Automata Determinization. In Proceedings of the Proceedings of the 13th International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2022, Madrid, Spain, September 21-23, 2022; Ganty, P.; Monica, D.D., Eds., 2022, Vol. 370, *EPTCS*, pp. 49–65. https://doi.org/10.4204/EPTCS.370.4.
17. Lick, A.; Schmitz, S. XPath Benchmark, Last visited April 13th 2022.
18. Mozafari, B.; Zeng, K.; Zaniolo, C. High-performance complex event processing over XML streams. In Proceedings of the SIGMOD Conference; Candan, K.S.; Chen, Y.; Snodgrass, R.T.; Gravano, L.; Fuxman, A.; Candan, K.S.; Chen, Y.; Snodgrass, R.T.; Gravano, L.; Fuxman, A., Eds. ACM, 2012, pp. 253–264. https://doi.org/10.1145/2213836.2213866.
19. Grez, A.; Riveros, C.; Ugarte, M. A Formal Framework for Complex Event Processing. 2019, pp. 5:1–5:18. https://doi.org/10.4230/LIPIcs.ICDT.2019.5.
20. Muñoz, M.; Riveros, C. Streaming Enumeration on Nested Documents. In Proceedings of the 25th International Conference on Database Theory, ICDT 2022, March 29 to April 1, 2022, Edinburgh, UK (Virtual Conference); Olteanu, D.; Vortmeier, N., Eds. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, Vol. 220, *LIPIcs*, pp. 19:1–19:18. https://doi.org/10.4230/LIPIcs.ICDT.2022.19.
21. Al Serhali, A.; Niehren, J. Earliest Query Answering for Deterministic Stepwise Hedge Automata. In Proceedings of the Implementation and Application of Automata - 27th International Conference, CIAA 2023, Famagusta, North Cyprus, September 19-22, 2023, Proceedings; Nagy, B., Ed. Springer, 2023, Vol. 14151, *Lecture Notes in Computer Science*, pp. 53–65. https://doi.org/10.1007/978-3-031-40247-0_3.

22. Gauwin, O.; Niehren, J.; Tison, S. Earliest Query Answering for Deterministic Nested Word Automata. In Proceedings of the 17th International Symposium on Fundamentals of Computer Theory. Springer Verlag, 2009, Vol. 5699, *Lecture Notes in Computer Science*, pp. 121–132. https://doi.org/10.1007/978-3-642-03409-1_12.
23. Debarbieux, D.; Gauwin, O.; Niehren, J.; Sebastian, T.; Zergaoui, M. Early nested word automata for XPath query answering on XML streams. *Theor. Comput. Sci.* **2015**, *578*, 100–125. https://doi.org/10.1016/j.tcs.2015.01.017.
24. Al Serhali, A.; Niehren, J. Subhedge Projection for Stepwise Hedge Automata. In Proceedings of the Fundamentals of Computation Theory - 24th International Symposium, FCT 2023, Trier, Germany, September 18-21, 2023, Proceedings; Fernau, H.; Jansen, K., Eds. Springer, 2023, Vol. 14292, *Lecture Notes in Computer Science*, pp. 16–31. https://doi.org/10.1007/978-3-031-43587-4_2.
25. Neumann, A.; Seidl, H. Locating Matches of Tree Patterns in Forests. In Proceedings of the Foundations of Software Technology and Theoretical Computer Science. Springer Verlag, 1998, Vol. 1530, *Lecture Notes in Computer Science*, pp. 134–145. https://doi.org/10.1007/978-3-642-03409-1_12.
26. Gauwin, O.; Niehren, J.; Roos, Y. Streaming Tree Automata. *Information Processing Letters* **2008**, *109*, 13–17. https://doi.org/10.1016/j.ipl.2008.08.002.
27. Franceschet, M. XPathMark Performance Test. https://users.dimi.uniud.it/~massimo.franceschet/xpathmark/PTbench.html. Accessed: 2020-10-25.
28. Hosoya, H.; Pierce, B.C. XDuce: A Statically Typed XML Processing Language. *ACM Trans. Internet Technol.* **2003**, *3*, 117–148. https://doi.org/10.1145/767193.767195.
29. Gécseg, F.; Steinby, M. *Tree Automata*; Akadémiai Kiadó, Budapest, 1984.
30. Carme, J.; Niehren, J.; Tommasi, M. Querying Unranked Trees with Stepwise Tree Automata. In Proceedings of the Rewriting Techniques and Applications, 15th International Conference, RTA 2004, Aachen, Germany, June 3-5, 2004, Proceedings; van Oostrom, V., Ed. Springer, 2004, Vol. 3091, *Lecture Notes in Computer Science*, pp. 105–118. https://doi.org/10.1007/978-3-540-25979-4_8.
31. Neumann, A.; Seidl, H. Locating Matches of Tree Patterns in Forests. In Proceedings of the 18-th Conference on Foundations of Software Technology and Theoretical Computer Science, 1998. https://doi.org/10.1007/b71635.
32. Debarbieux, D.; Gauwin, O.; Niehren, J.; Sebastian, T.; Zergaoui, M. Early nested word automata for XPath query answering on XML streams. *Theor. Comput. Sci.* **2015**, *578*, 100–125. https://doi.org/10.1016/J.TCS.2015.01.017.
33. Pair, C.; Quéré, A. Définition et etude des bilangages réguliers. *Information and Control* **1968**, *13*, 565–593.
34. Murata, M. Hedge Automata: a Formal Model for XML Schemata. Web page, 2000.
35. Martens, W.; Niehren, J. On the Minimization of XML Schemas and Tree Automata for Unranked Trees. *Journal of Computer and System Science* **2007**, *73*, 550–583.
36. Niehren, J. Stepwise Hedge Automata are exponentially more succinct than Forest Automata, **in preparation**.
37. Bojanczyk, M.; Walukiewicz, I. Forest algebras. In Proceedings of the Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]; Flum, J.; Grädel, E.; Wilke, T., Eds. Amsterdam University Press, 2008, Vol. 2, *Texts in Logic and Games*, pp. 107–132.

Appendix O Earliest Membership

We adapt the earliest membership tester for dSHAs from [21] so that it compiles to SHA$^{\downarrow}$s instead of NWAs and such that it accounts for schemas.

Appendix O.1 Schema Safety is an Inclusion Problem

The idea for testing certain membership and certain nonmembership is to consider states are safe to reach final states except if going out of the schema. Let $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ be an SHA with schema $\mathbf{S}$ such that there exists $F \subseteq F_{\mathbf{S}} \subseteq \mathcal{Q}$ with $S = Ł(A[F/F_{\mathbf{S}}])$. Note that $Ł(A) \subseteq S \subseteq \mathcal{H}_\Sigma$ in particular. The set of states that are safe to reach a subset $P \subseteq \mathcal{Q}$ for all hedges that do not go out of the schema $\mathbf{S}$ then is:

$$safe^{\Delta}_{\mathbf{S}}(P) = safe^{\Delta}(P \cup (\mathcal{Q} \setminus F_{\mathbf{S}}))$$

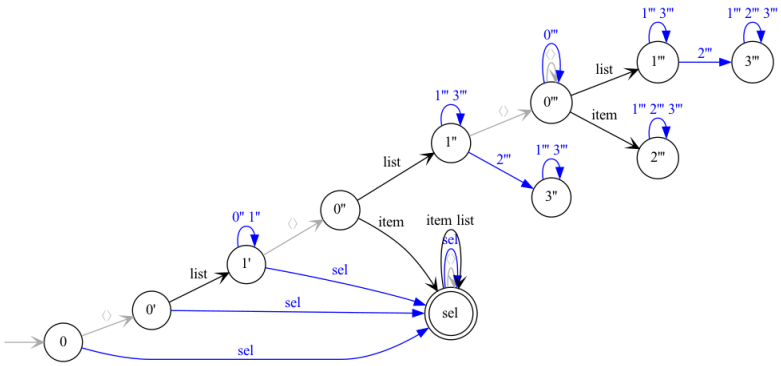


Figure A20. The earliest $d\text{SHA}^{\downarrow}$ A_e for the dSHA A in Fig. 15 with schema $[\![doc]\!]$ for the XPath filter `[self::list][child::item]`. The states of A_e correspond to the following tuples: $0 = (0,\{1,2,3,5\},\{4\})$, $0' = (0,\{2,4,5\},\{3,4\})$, $1' = (1,\{2,4,5\},\{3,4\})$, $0'' = (0,\emptyset,\{2,4,5\})$, $1'' = (1,\emptyset,\{2,4,5\})$, $0''' = (0,\emptyset,\emptyset)$, $1''' = (1,\emptyset,\emptyset)$, $2''' = (1,\emptyset,\emptyset)$, and $3''' = (1,\emptyset,\emptyset)$.

If A is deterministic and schema-complete for $\mathbf{S}$ then the safety of a state of A with respect to schema $\mathbf{S}$ is an inclusion problem.

Lemma A43. *Let $S = Ł(A[F/F_{\mathbf{S}}])$. If A is deterministic and schema-complete for $\mathbf{S}$ then for all states $q \in Q$:*

$$Ł(A[I/\{q\}, F/F_{\mathbf{S}}]) \subseteq Ł(A[I/\{q\}]) \Leftrightarrow q \in safe^{\Delta}_{\mathbf{S}}(F)$$

Proof. **"⇐".** Suppose that $q \notin safe^{\Delta}_{\mathbf{S}}(F)$. The definition of schema-based safety yields $acc^{\Delta}(\{q\}) \cap (F_{\mathbf{S}} \setminus F) \neq \emptyset$. So there exists some hedge $h \in \mathcal{H}_{\Sigma}$ such that such that $q \xrightarrow{h} F_{\mathbf{S}} \setminus F$ wrt Δ. In particular, $q \in Ł(A[I/\{q\}, F/F_{\mathbf{S}}])$. By determinism there exists no other transition on h from q and thus $q \notin Ł(A[I/\{q\}])$. Hence $Ł(A[I/\{q\}, F/F_{\mathbf{S}}]) \not\subseteq Ł(A[I/\{q\}]$.

"⇒". Suppose that $q \in safe^{\Delta}_{\mathbf{S}}(F)$. Let $h \in Ł(A[I/\{q\}, F/F_{\mathbf{S}}])$. Then there exists $q' \in F_{\mathbf{S}}$ such that $q \xrightarrow{h} q'$ wrt Δ. Hence $q' \in acc^{\Delta}(\{q\})$ so that $acc^{\Delta}(\{q\}) \subseteq F \cup (Q \setminus F_{\mathbf{S}})$ implies $q' \in F$. Thus, $h \in Ł(A[I/\{q\}])$.

□

Lemma A43 with $F_{\mathbf{S}} = Q$ shows Lemma 5 of [21], which states for any complete dSHAs A that safety is a universality problem:

$$Ł(A[I/\{q\}]) = \mathcal{H}_{\Sigma} \Leftrightarrow q \in safe^{\Delta}(F)$$

This is since the schema-completeness of A for $\mathbf{S} = \mathcal{H}_{\Sigma}$ implies the completeness of A and thus $Ł(A[I/\{q\}, F/Q]) = \mathcal{H}_{\Sigma}$.

Appendix O.2 Algorithm

We now present our earliest membership tester for dSHAs with schema restrictions. Given a SHA A we compute a $\text{SHA}^{\downarrow}$ $A_{e(\mathbf{S})} = (\Sigma, Q_e, \Delta_e, I_{e(\mathbf{S})}, F_{e(\mathbf{S})})$ as follows. Let *sel* be a fresh symbol. We start with set set of states that are safe for selection $S_0 = safe^{\Delta}_{\mathbf{S}}(F)$ and respectively safe for rejection $R_0 = safe^{\Delta}_{\mathbf{S}}(Q \setminus F)$. The state sets of $A_{e(\mathbf{S})}$ are then:

$$\begin{aligned} Q_e &= (Q \times 2^Q \times 2^Q) \cup \{sel\} \\ I_{e(\mathbf{S})} &= \{(q, R_0, S_0) \mid q \in I,\ q \notin R_0 \cup S_0\} \cup \{sel \mid I \cap S_0 \neq \emptyset\} \\ F_{e(\mathbf{S})} &= \{(q, R_0, S_0) \mid q \in F\} \cup \{sel\} \end{aligned}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q' \notin S \cup R}{(q,S,R) \xrightarrow{a} (q',S,R) \text{ in } \Delta_e}$$

$$\frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta \quad q' \notin S \cup R}{(q,S,R) \xrightarrow{\langle\rangle} (q', s\text{-}down^{\Delta}(q,S), s\text{-}down^{\Delta}(q,R)) \text{ in } \Delta_e}$$

$$\frac{q@p \to q' \text{ in } \Delta \quad q' \notin S \cup R}{(q,S,R)@(p, s\text{-}down^{\Delta}(q,S), s\text{-}down^{\Delta}(q,R)) \to (q',S,R) \text{ in } \Delta_e}$$

$$\frac{q \xrightarrow{a} q' \text{ in } \Delta \quad q' \in S}{(q,S,R) \xrightarrow{a} sel \text{ in } \Delta_e} \qquad \frac{q \xrightarrow{\langle\rangle} q' \text{ in } \Delta \quad q' \in S}{(q,S,R) \xrightarrow{\langle\rangle} sel \text{ in } \Delta_e}$$

$$\frac{q@p \to q' \text{ in } \Delta \quad q' \in S}{(q,S,R)@(p, s\text{-}down^{\Delta}(q,S), s\text{-}down^{\Delta}(q,R)) \to sel \text{ in } \Delta_e}$$

$$\frac{a \in \Sigma}{sel \xrightarrow{a} sel \text{ in } \Delta_e} \qquad \frac{\mu \in \mathcal{Q}_e}{sel@\mu \to sel \text{ in } \Delta_e} \qquad \frac{\mu \in \mathcal{Q}_e}{\mu@sel \to sel \text{ in } \Delta_e} \qquad \frac{true}{sel \xrightarrow{\langle\rangle} sel \text{ in } \Delta_e}$$

Figure A21. The earliest transition rules Δ_e inferred from transition rules Δ of some SHA.

The transition rules in Δ_e are given by the in Fig. A21. For illustration, reconsider the dSHA from Fig. 1. This dSHA is not complete but schema-complete for the schema $[\![doc]\!]$. We get $s\text{-}down^{\Delta}(0,\{4\}) = \{3,4\}$. It may seem counter intuitive that note only state 3 but also state 4 down remains safe for selection. This reflects the fact that no proper subhedge of any hedge satisfying the intended schema may ever get into state 4. Applying the earliest construction with safe-no-change projection to dSHA in Fig. 1 yields the dSHA$^{\downarrow}$ in Fig. A20.

Appendix O.3 Soundness and Completeness

We next provide soundness and completeness results for our earliest in-memory membership tester with projection.

Proposition A44 (Soundness)**.** *For any* SHA A *with schema* $\mathbf{S} = Ł(A[F/F_{\mathbf{S}}])$*, hedge* $h \in \mathbf{S}$*, and partial run* v *of* $A_{e(\mathbf{S})}$ *on* h *that ends in state sel then* $h \in Ł(A)$*.*

So if the partial run reaches state *sel* then $proj_{\Sigma}(v)$ is certain for membership to $Ł(A)$ with respect ot $\mathbf{S}$. Evaluating a SHA$^{\downarrow}$s $A_{e(\mathbf{S})}$ yields an earliest in-memory membership tester for dSHA A. A single adaptation is in order: whenever *sel* is reached, the evaluation can stop all over and accept the input hedge.

Theorem A9 (Completeness)**.** *Let* $\mathcal{A} = (\Sigma, \mathcal{Q}, \Delta, I, F)$ *be a* dSHA *that is schema-complete for schema* $\mathbf{S} = Ł(A[F/F_{\mathbf{S}}])$*. For any hedge* $h \in \mathbf{S}$ *and prefix* v *of* $nw(h)$*:*

- *if* v *is certain for membership in* $Ł(A)$ *with* $\mathbf{S}$ *then there exists a partial run* w *for* $A_{e(\mathbf{S})}$ *on* h *such that* $v = proj_{\Sigma}(w)$ *and* w *ends with state sel.*
- *if* v *is certain for nonmembership in* $Ł(A)$ *with* $\mathbf{S}$ *then there exists blocking partial run* w *for* $A_{e(\mathbf{S})}$ *on* h *such that* $proj_{\Sigma}(w)$ *is a prefix of* v*.*

This means that certainty for membership and nonmembership are detected at the earliest prefix when running the evaluator for $A_{e(\mathbf{S})}$.

Proof sketch. We notice that this algorithm is basically the same as the earliest membership tester from Proposition 6 of [21], except that it also check for safe rejection and that it ac-

counts for schemas. The fact that NWAs are used there instead of SHA$^{\downarrow}$s here is not essential. So we can lift the soundness and completeness proof from there without problem. □