Hedge automata revisited: Transforming texts to and from XML

Akihisa Yamada[1], Jérémy Dubut[1], and Takeshi Tsukada[2]

[1] National Institute of Advanced Industrial Science and Technology (AIST)
[2] Chiba University

Abstract. We develop automata-theoretic concepts for hedges, a mathematical model for XML documents. First, we define context-free grammars on hedges, and connect our formulation with existing definitions of regular hedge languages and hedge automata. We also extend regular automata, push-down automata, and push-down transducers to hedges, and verify that the well-known correspondences with language classes in the string case carry over to the hedge setting. Based on the theory, we introduce a tool TXtruct, which serves as a grammar validator and transformer for XML and text files.

1 Introduction

XML is a widely used format for structured documents in many applications. Once documents are written in XML, there are well-established tools to manipulate them. XML Schema [MTSM$^+$12] is the modern standard for defining a grammar over XMLs and validating documents against the grammar. XSL Translation (XSLT) [Kay21] is another standard for transforming XML documents into other formats, such as browsable HTMLs. As an example application, `Tact` [HOT17] is a document analysis tool based on XML. `Tact` reads structured documents such as industry standards or product specifications written in an XML format (see Fig. 1 left), and applies natural language processing techniques for instance to measure distances between sections. Those XML documents can be transformed into HTML and viewed by browsers.

While XML and HTML are perfect for machines to manipulate, for humans they are not easy formats to write or edit. As a consequence, Markdown (cf. [Leo16]) was invented as a human-friendly format for writing structured or semi-structured texts. Markdown is basically a plain text file with special grammar for structuring, but it also allows HTML snippets. Such semi-structured documents are called HTML fragments.

While XML documents are widespread, for their theoretical model *hedges*, the authors are aware of only basic foundation in the literature: regular hedge languages [PQ68] and (bottom-up) hedge automata [Mur99]. Therefore, in this paper we enrich the theory of hedges by importing well-known notions from automata theory of strings (words). First, we naturally extend regular and context-free grammars to hedges. We justify our definition by showing that

```
<tree>
 <title>The First Section</title>
 <body>This is a section text.</body>
 <tree>
  <title>A Subsection</title>
  <body>Texts can be
   <em>emphasized</em>.</body></tree>
 <tree>
  <title>Another Subsection</title>
...</tree></tree>
```

```
# The First Section

This is a section text.

## A Subsection

Texts can be
<em>emphasized</em>.

## Another Subsection
...
```

Fig. 1. A simplified snippet of an input XML file for `Tact` on the left and the same document in Markdown on the right.

– our regular hedge grammars characterize the existing class of regular hedge languages defined in the literature;
– the correspondence carries over to the context-free grammars when we naturally extend the existing formulation of regular hedge languages;
– the pumping lemma extends to the context-free hedge languages.

Then we introduce a (top-down) definition of hedge automata, which we prove to be equivalent to the existing bottom-up definition [Mur99]. We then generalize to push-down hedge automata, where the top-down nature of our definition becomes crucial.

Further, we introduce the hedge version of translation grammars and push-down transducers. We show that hedge relations describable by translation grammars are precisely those describable by push-down hedge transducers.

Based on the theory, we developed a tool `TXtruct` for defining and validating grammars over semi-structured documents including XML, Markdown and plain texts, and at the same time transforming them to other semi-structured formats. `TXtruct` is incorporated into `Tact`, which now supports Markdown files (see Fig. 1 right) as input and output format. A part of a `TXtruct` translation file from Markdown to XML is shown in Fig. 2.

The source code of `TXtruct` is made available at:

`https://github.com/AkihisaYamada/TXtruct`

When one is concerned with defining an XML grammar or transforming XML into other formats, XSD and XSLT are the standard ways. However, since these standards are not dedicated for defining string grammars or processing plain texts, they provide only limited support for defining or processing text contents inside XML elements. In contrast, `TXtruct` provides dedicated and uniform support for defining and processing text contents.

When one is concerned with translating Markdown documents into other formats, `Pandoc` [Dom14] is such a tool. Although `Pandoc` supports many different input and output formats, adding a new format requires to implement a new

```
<txtruct method="xml" indent="no">
<input method="html-fragment"/>
<class name="section">
 <option level="1"> <match pattern="#␣"/>
  <element name="tree">
   <call ref="section-body"/>
   <call ref="section" level="2" minOccurs="0" maxOccurs="unbounded"/>
  </element></option>
 <option level="2"> <match pattern="##␣"/>...
```

Fig. 2. A `TXtruct` translation rule file for translating Markdown into the XML format.

interface. In contrast, `TXtruct` is a generic tool, where adding a new format is defining a hedge translation grammar.

2 Hedge Grammars

We assume certain familiarity with string grammars. We refer to a textbook [CBM19] for terminologies unexplained in the paper.

Hedges are extension of strings with unary function symbols.

Definition 1 (hedges). *A* hedge signature *is $\Sigma = \Sigma_0 \uplus \Sigma_1$, where Σ_0 and Σ_1 are finite sets of characters and function symbols, respectively. We write $\Sigma(X)$ for $\Sigma_0 \uplus \Sigma_1 \times X$, and $f(x)$ for $\langle f, x\rangle \in \Sigma_1 \times X$. The set $\mathcal{H}(\Sigma)$ of* hedges *is defined by the following grammar:*

$$s ::= \varepsilon \mid s\, s \mid c \mid f(s)$$

where c ranges over Σ_0 and f over Σ_1.

Example 1. In the following examples we consider characters in the `typewriter` font, ␣ for space and \n for the new line code are in Σ_0; and words in roman font are in Σ_1. We view parts of XML and Markdown of Fig. 1 as the following hedges:

$$t_{\mathrm{xml}} = \mathrm{tree}\big(\mathrm{title}(w_1)\,\mathrm{body}(w_2)\,\mathrm{tree}(\mathrm{title}(w_3)\,\mathrm{body}(w_4\,\mathrm{em}(w_5)\,\texttt{.}))\big)$$
$$t_{\mathrm{md}} = \texttt{\#}\,␣\,w_1\,\backslash\mathrm{n}\,w_2\,\backslash\mathrm{n}\,\texttt{\#\#}\,␣\,w_3\,\backslash\mathrm{n}\,w_4\,\mathrm{em}(w_5)\,\texttt{.}\,\backslash\mathrm{n}$$

where w_1 = `The␣First␣Section`, w_2 = `This␣is␣a␣section␣text.`, w_3 = `A␣Subsection`, w_4 = `Texts␣can␣be␣`, and w_5 = `emphasized`.

Not all hedges are valid HTML or Markdown; e.g., an HTML file must consist of a single html element, and br elements must not contain inner content. Hence we introduce grammars on hedges.

Definition 2 (hedge grammars). *Let D be a set, whose elements are called* non-terminal *symbols, disjoint with Σ. The set $\mathcal{H}(\Sigma, D)$ is defined as $\mathcal{H}(\Sigma')$ where $\Sigma'_0 = \Sigma_0 \uplus D$ and $\Sigma'_1 = \Sigma_1$. A* production rule *$d \to e$ is a pair of $d \in D$ and $e \in \mathcal{H}(\Sigma, D)$. A* context-free hedge grammar (CFHG) *is $G = \langle \Sigma, D, d_0, R\rangle$ where $d_0 \in D$ and $R \subseteq D \times \mathcal{H}(\Sigma, D)$ is a finite set of production rules.*

We extend some classes of string grammars to hedges. A CFHG is realtime *if every rule is in one of the forms (1) $d \to \varepsilon$, or (2) $d \to c\, e_1 \dots e_n$, or (3) $d \to f(e')\, e_1 \dots e_n$. It is in* Greibach normal form *if moreover $e', e_1, \dots, e_n \in D$, and is* regular *if moreover $n = 1$.*

Greibach normal form of hedge grammars extends that of string grammars with case (3). Moreover, it allows case (1) for any $d \in D$, whereas that of string grammars allows only for d_0 to admit ε in the language. In the hedge setting, case (1) is crucial for generating hedges such as $f(\varepsilon)$.

We may use BNF $d ::= e_1 \mid \cdots \mid e_n$ to represent multiple rules $d \to e_1, \dots, d \to e_n$. When the set of rules of concern is clear, we may also use the extended BNF: expressions e^* and $(e_1 \mid \cdots \mid e_n)$ represent fresh nonterminals d with rules $d ::= \epsilon \mid e\, d$ and $d ::= e_1 \mid \cdots \mid e_n$, respectively.

Example 2. In examples, symbols in *italic* are nonterminals. The hedge grammar G_{md} consisting of the following production rules defines the structure of (a simplified version of) Markdown.

$$\begin{aligned}
\textit{section}_1 &::= \texttt{\#}\,␣\ \textit{line}\backslash\mathrm{n}\,\textit{lines}\ \textit{section}_2{}^*\\
\textit{section}_2 &::= \texttt{\#\#}\,␣\,\textit{line}\backslash\mathrm{n}\,\textit{lines}\ \textit{section}_3{}^* \dots\\
\textit{line} &::= \varepsilon \mid \textit{char}\ \textit{line} \mid \textit{element}\ \textit{line} \qquad \textit{lines} ::= \varepsilon \mid \textit{line}\backslash\mathrm{n}\,\textit{lines}\\
\textit{char} &::= ␣ \mid \texttt{0} \mid \texttt{a} \mid \texttt{A} \mid \cdots \qquad \textit{element} ::= \mathrm{br}() \mid \mathrm{em}(\textit{lines}) \mid \cdots
\end{aligned}$$

Example 3. The following regular hedge grammar describes the (simplified) XML format that `Tact` can read:

$$\begin{aligned}
&\textit{trees} ::= \varepsilon \mid \mathrm{tree}(\textit{tree})\ \textit{trees}\\
&\textit{tree} ::= \mathrm{title}(\textit{text})\ \textit{tree}' \qquad \textit{tree}' ::= \mathrm{body}(\textit{text})\ \textit{end} \qquad \textit{end} ::= \varepsilon\\
&\textit{text} ::= \varepsilon \mid ␣\ \textit{text} \mid \texttt{0}\ \textit{text} \mid \texttt{a}\ \textit{text} \mid \texttt{A}\ \textit{text} \mid \mathrm{br}(\textit{end})\ \textit{text} \mid \mathrm{em}(\textit{text})\ \textit{text} \mid \cdots
\end{aligned}$$

Definition 3 (hedge languages). *A* hedge language *is a set of hedges. For a set $R \subseteq D \times \mathcal{H}(\Sigma, D)$ of production rules, $e \in \mathcal{H}(\Sigma, D)$, and $s \in \mathcal{H}(\Sigma)$, we define when $R \vdash e : s$ holds by the following inference rules:*

$$\frac{}{R \vdash \varepsilon : \varepsilon} \qquad \frac{R \vdash e : s \qquad R \vdash e' : s'}{R \vdash e\,e' : s\,s'} \qquad \frac{R \vdash e : s}{R \vdash d : s}\ \textit{if}\ \ d \to e$$

$$\frac{}{R \vdash c : c} \qquad \frac{R \vdash e : s}{R \vdash f(e) : f(s)}$$

We define the hedge language *generated by a CFHG $G = \langle \Sigma, D, d_0, R\rangle$ as $\mathcal{L}(G) := \{s \in \mathcal{H}(\Sigma) \mid R \vdash d_0 : s\}$. A hedge language is* context-free *if it is generated by a CFHG.*

Theorem 1 (Greibach normal form). *Every context-free hedge language is generated by a CFHG in Greibach normal form.*

This theorem will be proved more generally in Section 5.

3 Correspondence with Existing Formulations

In this section we justify our definition of regular hedge grammars by showing that a hedge language is generated by a regular hedge grammar if and only if it is a regular hedge language [PQ68]. We also show that this equivalence holds when replacing regularity by context-freeness. This suggests pumping lemmas for languages generated by regular or context-free hedge grammars. Finally, we show that the balanced-string representation [MT06] of context-free hedge languages is strictly less expressive than context-free string languages.

3.1 Levelwise Definition of Hedge Languages

Pair and Quere [PQ68] defined regular hedge languages via regular string languages. We extend their definition also to context-free languages. First, their formulation considers truncation of hedges into the string of the root symbols.

Definition 4 (root word). *The* root word $\overline{s} \in \Sigma^*$ *of a hedge* $s \in \mathcal{H}(\Sigma)$ *is defined by* $\overline{\varepsilon} = \varepsilon$; $\overline{c\,s} = c\,\overline{s}$ *if* $c \in \Sigma_0$; *and* $\overline{f(s)\,s'} = f\,\overline{s'}$ *if* $f \in \Sigma_1$.

Their definition then considers the set of root words that can occur under each function symbol. We formulate the notion concisely using contexts.

Definition 5 (f-productions). *A* hedge context C *is a hedge, where a special character* $\Box$ *occurs exactly once. We write* $C[s]$ *for the hedge obtained by replacing* $\Box$ *by* s. *Given* $f \in \Sigma_1$, *the set* $\mathsf{P}_f(s) \subseteq \Sigma^*$ *of* f-productions *of a hedge* $s \in \mathcal{H}(\Sigma)$ *is defined by* $\mathsf{P}_f(s) := \left\{\overline{s'} \;\middle|\; \exists C.\ s = C[f(s')]\right\}$.

Definition 6 (grammatical language). *Given a family* $L_{\_} = \{L_x\}_{x \in \Sigma_1 \cup \{\varepsilon\}}$ *of string languages over* Σ, *the* grammatical (hedge) language *induced by* $L_{\_}$ *is:*

$$\mathcal{G}(L_{\_}) := \{s \in \mathcal{H}(\Sigma) \mid \overline{s} \in L_\varepsilon \wedge \forall f \in \Sigma_1.\ \mathsf{P}_f(s) \subseteq L_f\}$$

We are not yet there; Pair and Quere's formulation further allows to *transcribe* signatures used in grammatical languages into the target signatures.

Definition 7 (transcription). *A* transcription $\mu : \Sigma' \to \Sigma$ *over hedge signatures* Σ' *to* Σ *is a pair of mappings* $\mu_0\colon\ \Sigma'_0 \to \Sigma_0$ *and* $\mu_1\colon\ \Sigma'_1 \to \Sigma_1$. *We write* c^μ *for* $\mu_0(c)$ *and* f^μ *for* $\mu_1(f)$. *Given* $s \in \mathcal{H}(\Sigma', D)$, *we define* $\mu(s) \in \mathcal{H}(\Sigma, D)$ *by* $\mu(\varepsilon) = \varepsilon$, $\mu(c\,s) = c^\mu\,\mu(s)$, $\mu(f(s)\,s') = f^\mu(\mu(s))\,\mu(s')$, *and* $\mu(d\,s) = d\,\mu(s)$.

Definition 8 (hedge languages [PQ68]). *We say that a hedge language* $H \subseteq \mathcal{H}(\Sigma)$ *is* regular *(resp.* context-free*) if there are hedge signature* Σ', *transcription* $\mu : \Sigma' \to \Sigma$, *and family* $L_{\_} = \{L_x\}_{x \in \Sigma'_1 \cup \{\varepsilon\}}$ *of regular (resp. context-free) languages such that* $H = \mu(\mathcal{G}(L_{\_}))$, *i.e.,* $\{\mu(s) \mid s \in \mathcal{G}(L_{\_})\}$.

In [PQ68], regular hedge languages are actually defined as those languages that are recognized by *binoids*, and it is then proved that this is equivalent to the above definition. Another equivalent characterization proved there is that the class of regular hedge languages is the smallest class containing finite languages and closed under several operations such as boolean operations, concatenation, iteration, and transcriptions (similar to Kleene's theorem).

The following theorem justifies our terminology based on hedge grammars.

Theorem 2. *A hedge language is context-free (regular) if and only if it is generated by a context-free (regular) hedge grammar.*

Proof.

($\Rightarrow$) Consider a hedge language given by transcription $\mu : \Sigma' \to \Sigma$ and family $\{L_x\}_{x \in \Sigma'_1 \cup \{\varepsilon\}}$ of context-free (regular) string languages. Suppose that each L_x is generated by a string grammar $G_x = \langle \Sigma', D_x, d_{0,x}, R_x \rangle$ which is in Greibach normal form (regular). Without loss of generality, we assume D_x are pairwise disjoint. We define $\hat{\mu} : \Sigma' \to \mathcal{H}(\Sigma, D)$ by $\hat{\mu}(c) = c^\mu$ and $\hat{\mu}(f) = f^\mu(d_{0,f})$, and extend naturally on $(\Sigma' \cup D)^*$. Now consider CFHG $G^\mu = \langle \Sigma, D, d_{0,\varepsilon}, R^\mu \rangle$ defined by $D = \bigcup_{x \in \Sigma'_1 \cup \{\varepsilon\}} D_x$ and

$$R^\mu = \{d \to \hat{\mu}(e) \mid d \to e \in R_x,\ x \in \Sigma'_1 \cup \{\varepsilon\}\}$$

Note that G^μ is regular if all G_x are regular. We first prove

$$R_x \vdash d : w \quad \text{and} \quad R^\mu \vdash \hat{\mu}(w) : s \quad \text{implies} \quad R^\mu \vdash d : s \tag{1}$$

by induction on the former derivation. By the assumption on G_x, we must have $y \in \Sigma'$, $d \to y\, d_1 \ldots d_n \in R_x$, and $w = y\, w_1 \ldots w_n$ such that $R_x \vdash d_i : w_i$ for each i. With the second premise, we know $s = s'\, s_1 \ldots s_n$ such that $R^\mu \vdash \hat{\mu}(y) : s'$ and $R^\mu \vdash \hat{\mu}(w_i) : s_i$. So by induction hypothesis (IH) we have $R^\mu \vdash d_i : s_i$. We conclude $R^\mu \vdash d : s'\, s_1 \ldots s_n$ with $d \to \hat{\mu}(y)\, d_1 \ldots d_n \in R^\mu$.

Let us prove now that $\mu(\mathcal{G}(L_\text{-})) = \mathcal{L}(G^\mu)$.

($\subseteq$) Let $s \in \mu(\mathcal{G}(L_\text{-}))$, so there is $s' \in \mathcal{G}(L_\text{-})$ with $\mu(s') = s$. We show $s \in \mathcal{L}(G^\mu)$, that is, $R^\mu \vdash d_{0,\varepsilon} : s$. By $s' \in \mathcal{G}(L_\text{-})$ we know $R_\varepsilon \vdash d_{0,\varepsilon} : \overline{s'}$, so thanks to (1), it suffices to show $R^\mu \vdash \hat{\mu}(\overline{s'}) : s$. We prove that $R^\mu \vdash \hat{\mu}(\overline{s''}) : \mu(s'')$ for any subhedge s'' of s' by induction. Let $s'' = t_1 \ldots t_n$ with $t_1, \ldots, t_n \in \Sigma'(\mathcal{H}(\Sigma'))$. We conclude by showing $R^\mu \vdash \hat{\mu}(\overline{t_i}) : \mu(t_i)$ for all i. If $t_i \in \Sigma'_0$, then $\hat{\mu}(\overline{t_i}) = \mu(t_i)$ and we are done. Let $t_i = f(t')$. Then $\hat{\mu}(\overline{t_i}) = \hat{\mu}(f) = f^\mu(d_{0,f})$ and $\mu(t_i) = f^\mu(\mu(t'))$. By IH $R^\mu \vdash \hat{\mu}(\overline{t'}) : \mu(t')$, and since $f(t')$ is a subhedge of $s' \in \mathcal{G}(L_\text{-})$, we know $R_f \vdash d_{0,f} : \overline{t'}$. By (1) we get $R^\mu \vdash d_{0,f} : \mu(t')$ and we are done.

($\supseteq$) We prove that, if $R^\mu \vdash d : s$ for $x \in \Sigma'_1 \cup \{\varepsilon\}$ and $d \in D_x$, then there exists $s' \in \mathcal{H}(\Sigma')$ such that $s = \mu(s')$, $R_x \vdash d : \overline{s'}$, and $\mathsf{P}_f(s') \subseteq L_f$. This is sufficient, since if $s \in \mathcal{L}(G^\mu)$, i.e., $R^\mu \vdash d_{0,\varepsilon} : s$, then we obtain s' such that $s = \mu(s')$, $R_\varepsilon \vdash d_{0,\varepsilon} : \overline{s'}$ and $\mathsf{P}_f(s') \subseteq L_f$, i.e., $s' \in \mathcal{G}(L_\text{-})$, and thus $s \in \mu(\mathcal{G}(L_\text{-}))$. We prove the claim by induction on the derivation of $R^\mu \vdash d : s$. The following three cases are possible:

– $d \to \varepsilon \in R_x$ and $s = \varepsilon$. Then $s' = \varepsilon$ works.
– $d \to c\, d_1 \dots d_m \in R_x$, $s = c^\mu\, s_1 \dots s_m$, and $R^\mu \vdash d_i : s_i$. By IH we obtain $s'_1, \dots, s'_m$ such that $R_x \vdash d_i : \overline{s'_i}$ and $s_i = \mu(s'_i)$. Thus $s' = c\, s'_1 \dots s'_m$ works.
– $d \to f\, d_1 \dots d_m \in R_x$, $s = f^\mu(t)\, s_1 \dots s_m$, $R^\mu \vdash d_{0,f} : t$ and $R^\mu \vdash d_i : s_i$. By IH we obtain t' such that $t = \mu(t')$ and $R_f \vdash d_{0,f} : \overline{t'}$, i.e., $\overline{t'} \in L_f$, which is needed as $\overline{t'} \in \mathsf{P}_f(s')$; and $s'_1, \dots, s'_m$ such that $R_x \vdash d_i : \overline{s'_i}$ and $s_i = \mu(s'_i)$. By $s' = f(t')\, s'_1 \dots s'_m$ we conclude.

(⇐) Let $G = \langle \Sigma, D, d_0, R\rangle$ be a hedge grammar in Greibach normal form. We view $\Sigma(D)$ as a hedge signature, whose characters are original characters and function symbols are of form $f(d)$. Transcription $\mu : \Sigma(D) \to \Sigma$ is given by $c^\mu = c$ and $f(d)^\mu = f$. Then we consider the family $\{G_x\}_{x \in \Sigma(D)_1 \cup \{\varepsilon\}}$ of string grammars $\langle \Sigma(D), D, d_{0,x}, R\rangle$, where $d_{0,\varepsilon} = d_0$ and $d_{0,f(d)} = d$. Note that they are regular if G is regular. Consider the hedge grammar G^μ obtained from $\{G_x\}_x$ as in the (⇒) direction above. Observe that this grammar is exactly the same as G except that the non-terminals have been duplicated. It is easy to see that $\mu(\mathcal{G}(\mathcal{L}(G_{\text{-}}))) = \mathcal{L}(G^\mu) = \mathcal{L}(G)$. □

This correspondence with string languages at each level allows us to prove a pumping lemma for regular and context-free hedge languages. In the following, we formulate one in the case of context-free languages.

Lemma 1. *Let $H \subseteq \mathcal{H}(\Sigma)$ be context free. Then there exists $p \in \mathbb{N}$ such that if $C[s] \in H$ and $|\overline{s}| \geq p$, then s has a decomposition $s = s_1\, s_2\, s_3\, s_4\, s_5$ satisfying $|\overline{s_2\, s_3\, s_4}| \leq p$, $|\overline{s_2}| + |\overline{s_4}| > 0$, and $C[s_1\, s_2^m\, s_3\, s_4^m\, s_5] \in H$ for any $m \in \mathbb{N}$.*

Proof. By Theorem 2, there are a signature Σ', family $L_{\text{-}} = \{L_x\}_{x \in \Sigma'_1 \cup \{\varepsilon\}}$ of context-free string languages over Σ', and transcription $\mu : \Sigma' \to \Sigma$ such that $\mu(\mathcal{G}(L_{\text{-}})) = H$. For each $x \in \Sigma' \cup \{\varepsilon\}$, the pumping lemma (see [BPS61]) on L_x gives $p_x \in \mathbb{N}$ such that every word $w \in L_x$ with $p_x \leq |w|$ has a decomposition $w = w_1 w_2 w_3 w_4 w_5$ with $|w_2 w_3 w_4| \leq p_x$, $|w_2| + |w_4| > 0$, and $w_1 w_2^m w_3 w_4^m w_5 \in L_x$ for any $m \in \mathbb{N}$. Define $p = \max\{p_x \mid x \in \Sigma'_1 \cup \{\varepsilon\}\}$.

Consider now s with $|\overline{s}| \geq p$ and $C[s] \in H = \mu(\mathcal{G}(L_{\text{-}}))$. Hence there exists $t' \in \mathcal{G}(L_{\text{-}})$ such that $C[s] = \mu(t')$, and hence $t' = C'[s']$ with $\mu(C') = C$ and $\mu(s') = s$. Let $x = \varepsilon$ if $C' = \square$ and $x = f'$ if $C' = C''[f'(\square)]$. Since $C'[s'] \in \mathcal{G}(L_{\text{-}})$, we know $\overline{s'} \in L_x$. Moreover $\left|\overline{s'}\right| = |\overline{s}| \geq p \geq p_x$. So by the pumping lemma in L_x, we decompose $\overline{s'} = w_1 w_2 w_3 w_4 w_5$ as above. So let $s' = s'_1\, s'_2\, s'_3\, s'_4\, s'_5$ with $\overline{s'_i} = w_i$, and let $s_i = \mu(s'_i)$.

Now we prove $C[s_1\, s_2^m\, s_3\, s_4^m\, s_5] \in H$. By construction, it means to prove that $t'_m := C'[s'_1\, (s'_2)^m\, s'_3\, (s'_4)^m\, s'_5]$ is in $\mathcal{G}(L_{\text{-}})$. This holds because $\mathsf{P}_y(t'_m) \subseteq \mathsf{P}_y(t') \subseteq L_y$ if $y \neq x$ and $\mathsf{P}_x(t'_m) \subseteq \mathsf{P}_x(t') \cup \{w_1 w_2^m w_3 w_4^m w_5\} \subseteq L_x$ since $t' \in \mathcal{G}(L_{\text{-}})$ and by the pumping lemma in L_x. □

3.2 Balanced-String Representation

Here we consider representation of hedges as balanced strings over paired alphabets [MT06]. Given a hedge signature Σ, we define the *paired alphabet* by

$\widehat{\Sigma} = \Sigma_0 \cup \{ \acute{f}, \grave{f} \mid f \in \Sigma_1 \}$. The string representation $\widehat{s} \in \widehat{\Sigma}^*$ of a hedge $s \in \mathcal{H}(\Sigma)$ is defined as follows:

$$\widehat{\varepsilon} = \varepsilon \qquad \widehat{s_1\, s_2} = \widehat{s_1}\, \widehat{s_2} \qquad \widehat{c} = c \qquad \widehat{f(s)} = \acute{f}\, \widehat{s}\, \grave{f}$$

Given a hedge language H, we let $\widehat{H} = \{\widehat{s} \mid s \in H\}$.

Theorem 3. *Let $H \subseteq \mathcal{H}(\Sigma)$ be a context-free hedge language. Then $\widehat{H} \subseteq \widehat{\Sigma}^*$ is a context-free string language.*

Proof. Let H be generated by a hedge grammar $G = \langle \Sigma, D, d_0, R \rangle$ in Greibach normal form. Define the context-free string grammar $\widehat{G} = \langle \widehat{\Sigma}, D, d_0, \widehat{R} \rangle$ by

$$\widehat{R} = \{ d \to \widehat{e} \mid d \to e \in R \}$$

where $\widehat{\cdot}$ is extended with $\widehat{d} = d$. Let us prove that $\mathcal{L}(\widehat{G}) = \widehat{H}$.

($\subseteq$) We show by induction on the derivation that $\widehat{R} \vdash d : w$ implies $R \vdash d : s$ for some hedge s with $\widehat{s} = w$. Three cases are possible:
- $d \to \varepsilon \in R$ and $w = \varepsilon$.
- $d \to c\, d_1 \dots d_n \in R$, $w = c\, w_1 \dots w_n$, and $\widehat{R} \vdash d_i : c_i$ for each i.
- $d \to f(d')\, d_1 \dots d_n \in R$, $w = \acute{f} w' \grave{f}\, w_1 \dots w_n$, $\widehat{R} \vdash d' : w'$, and $\widehat{R} \vdash d_i : w_i$ for each i.

We only consider the last case, as other cases are trivial or similar. Induction hypothesis gives $s', s_1, \dots, s_n$ such that $\widehat{s'} = w'$, $R \vdash d' : s'$, $\widehat{s_i} = w_i$, $R \vdash d_i : s_i$ for each i. By letting $s = f(s')\, s_1 \dots s_n$ we have $\widehat{s} = w$, and we conclude $R \vdash d : s$ by an obvious derivation.

($\supseteq$) It is enough to prove by induction on the derivation that $R \vdash d : s$ implies $\widehat{R} \vdash d : \widehat{s}$. Three cases are possible:
- $d \to \varepsilon \in R$ and $s = \varepsilon$.
- $d \to c\, d_1 \dots d_n$, $s = c\, s_1 \dots s_n$, and $R \vdash d_i : s_i$ for each i.
- $d \to f(d')\, d_1 \dots d_n$, $s = f(s')\, s_1 \dots s_n$, $R \vdash d' : s'$, and $R \vdash d_i : s_i$.

We only consider the last case as the other cases are trivial or similar. By IH we have $\widehat{R} \vdash d' : \widehat{s'}$ and $\widehat{R} \vdash d_i : \widehat{s_i}$. Since $d \to \acute{f}\, d' \grave{f}\, d_1 \dots d_n \in R$, we conclude $\widehat{R} \vdash d : \widehat{s}$. ◻

The converse of the above theorem does not hold.

Proposition 1. *There is a hedge language $H \subseteq \mathcal{H}(\Sigma)$ such that $\widehat{H} \subseteq \widehat{\Sigma}^*$ is context free but H is not context free.*

Proof. A counterexample is given by the following string grammar:

$$W ::= \acute{\mathrm{f}}\, X \qquad\qquad X ::= \grave{\mathrm{f}} \mid \mathtt{a}\, X\, \mathtt{a} \mid \mathtt{b}\, X\, \mathtt{b}$$

where $\Sigma_0 = \{\mathtt{a}, \mathtt{b}\}$ and $\Sigma_1 = \{\mathrm{f}\}$. This grammar generates the string representations of hedges from $\{\mathrm{f}(w)\, w^{\mathrm{op}} \mid w \in \Sigma_0^*\}$, where w^{op} is the reverse of w. This hedge language is not context-free: Suppose on the contrary it is context-free. By Lemma 1 on $\mathrm{f}(w)\, w^{\mathrm{op}}$ with w long enough, we are able to pump inside f, that is, we can decompose $w = w_1 w_2 w_3 w_4 w_5$ with $|w_2 w_4| > 0$ such that $\mathrm{f}(w_1 w_3 w_5)\, w^{\mathrm{op}}$ would also be in L. This is impossible because $|w_1 w_3 w_5| < |w|$. ◻

Although the class of hedge languages with context-free string representations differs from the class of context-free hedge languages, they share an interesting property: languages in these classes are "regular in depth" (though they can be non-regular in width). This contrasts with another context-free notion, *context-free tree languages* [Rou69], which can be "non-regular in depth".

We formalize the claim. The *path language* $\mathsf{path}(s)$ of a hedge s is defined by

$$\mathsf{path}(\varepsilon) = \mathsf{path}(c) = \{\varepsilon\} \qquad \mathsf{path}(s_1\, s_2) = \mathsf{path}(s_1) \cup \mathsf{path}(s_2)$$
$$\mathsf{path}(f(s)) = \{fp \mid p \in \mathsf{path}(s)\}.$$

The path language $\mathsf{path}(H)$ of a hedge language H is the union $\bigcup_{s\in H} \mathsf{path}(s)$ of paths of hedges $s \in H$.

Lemma 2. *For a hedge language H, if $\widehat{H}$ is context free, $\mathsf{path}(H)$ is regular.*

Proof. We explicitly construct a grammar for $\mathsf{path}(H)$, based on the analysis in [MT06] of context-free string grammars that always generates matched tags. □

As a consequence, there exists a hedge language generated by a context-free tree grammar whose string representation is not context free.

4 (Push-Down) Hedge Automata

Here we present our definitions of regular and push-down hedge automata. Given a set X, we write $X^?$ for $X \uplus \{\varepsilon\}$. For simplicity, we do not assume a set of final states, but extend Q to $Q^?$ and use ε as the unique final state.

Definition 9 (hedge automaton). *A* hedge automaton *is $A = \langle \Sigma, Q, q_0, \delta\rangle$ where $q_0 \in Q$ is the initial state, and $\delta \subseteq Q \times \Sigma(Q)^? \times Q^?$ is the set of transition rules. We denote a transition rule $\langle q, \alpha, q'\rangle \in \delta$ by $q \xrightarrow{\alpha} q'$.*

We formulate runs of hedge automata as a transition relation over configurations. Unlike the string case, a configuration is not just the pair of the current state and remaining input ($Q \times \mathcal{H}(\Sigma)$), but additionally maintain a worklist in $(Q \times \mathcal{H}(\Sigma))^*$, assigning parts of input to states.

Definition 10 (runs). *Given a hedge automaton $A = \langle \Sigma, Q, q_0, \delta\rangle$, we define binary relation $\to_\delta$ over $Q \times \mathcal{H}(\Sigma) \times (Q \times \mathcal{H}(\Sigma))^*$ as the least relation such that:*

1. *$q \xrightarrow{\varepsilon} q' \in \delta$ implies $\langle q, s, W\rangle \to_\delta \langle q', s, W\rangle$;*
2. *$q \xrightarrow{c} q' \in \delta$ implies $\langle q, c\, s, W\rangle \to_\delta \langle q', s, W\rangle$;*
3. *$q \xrightarrow{f(q')} q'' \in \delta$ implies $\langle q, f(s)\, s', W\rangle \to_\delta \langle q', s, \langle q'', s'\rangle\, W\rangle$;*
4. *$\langle \varepsilon, \varepsilon, \langle q, s'\rangle\, W\rangle \to_\delta \langle q, s', W\rangle$.*

We say that A accepts *a hedge $s \in \mathcal{H}(\Sigma)$ if $\langle q_0, s, \varepsilon\rangle \to_\delta^* \langle \varepsilon, \varepsilon, \varepsilon\rangle$. We write $\mathcal{L}(A)$ for the set of hedges that A accepts, and say that A* recognizes *the language $\mathcal{L}(A)$.*

Item *1* of Definition 10 is the ε-transition: it moves states without touching the input (and the worklist). Item *2* is the character transition, which reads a character from the input and moves states. Item *3* is specific for hedge automata: it reads the function symbol f from the current input $f(s)\,s'$, moves on to read the argument s in state q', and assigns the remaining s' to state q'' in the worklist. Item *4* tells that if the current input was successfully read, and the worklist assigns s' to q', then one should proceed to this task.

Example 4. Consider a hedge automaton $A = \langle \Sigma, \{q_0, q_1, q_2\}, q_0, \delta\rangle$, where $\Sigma_0 = \{\mathtt{a}, \mathtt{b}, \dots\}$, $\Sigma_1 = \{\mathrm{tree}, \mathrm{title}, \dots\}$, and δ consists of the following transition rules:

$$\begin{array}{lllll} q_0 \xrightarrow{\varepsilon} \varepsilon & q_0 \xrightarrow{\mathrm{tree}(q_1)} q_0 & q_1 \xrightarrow{\mathrm{title}(q_2)} q_3 & q_3 \xrightarrow{\mathrm{body}(q_2)} q_0 & \\ q_2 \xrightarrow{\varepsilon} \varepsilon & q_2 \xrightarrow{\mathrm{em}(q_2)} q_2 & q_2 \xrightarrow{\mathtt{a}} q_2 & q_2 \xrightarrow{\mathtt{b}} q_2 & \dots \end{array}$$

A accepts the hedge t_{xml} of Example 1, as illustrated by the following run:

$$\begin{aligned} \langle q_0, t_{\mathrm{xml}}, \varepsilon\rangle &= \langle q_0, \mathrm{tree}(\mathrm{title}(w_1)\,\mathrm{body}(w_2)\,\mathrm{tree}(\dots)), \varepsilon\rangle \\ &\to_\delta \langle q_1, \mathrm{title}(w_1)\,\mathrm{body}(w_2)\,\mathrm{tree}(\dots), \langle q_0, \varepsilon\rangle\rangle \\ &\to_\delta \langle q_2, w_1, \langle q_3, \mathrm{body}(w_2)\,\mathrm{tree}(\dots)\rangle\,\langle q_0, \varepsilon\rangle\rangle \\ &\to_\delta^* \langle \varepsilon, \varepsilon, \langle q_3, \mathrm{body}(w_2)\,\mathrm{tree}(\dots)\rangle\,\langle q_0, \varepsilon\rangle\rangle \\ &\to_\delta \langle q_3, \mathrm{body}(w_2)\,\mathrm{tree}(\dots), \langle q_0, \varepsilon\rangle\rangle \\ &\to_\delta^* \langle q_0, \mathrm{tree}(\dots), \langle q_0, \varepsilon\rangle\rangle \to_\delta^* \langle \varepsilon, \varepsilon, \langle q_0, \varepsilon\rangle\rangle \to_\delta \langle q_0, \varepsilon, \varepsilon\rangle \to_\delta \langle \varepsilon, \varepsilon, \varepsilon\rangle \end{aligned}$$

As in the string case, hedge automata characterize regular hedge languages.

Theorem 4. *A hedge language is regular if and only if it is recognized by a hedge automaton.*

Proof. We can assume that a hedge automaton contains no transition rule of form $q \xrightarrow{\alpha} \varepsilon$ for $\alpha \in \Sigma(Q)$: One can replace such rules by $q \xrightarrow{\alpha} \epsilon$, where ϵ is a fresh state with transition rule $\epsilon \xrightarrow{\varepsilon} \varepsilon$. Then, the following ϕ is bijective from regular hedge grammars to such hedge automata: For regular hedge grammar $G = \langle \Sigma, D, d_0, R\rangle$, hedge automaton $\phi(G) = \langle \Sigma, D, d_0, \delta\rangle$ is defined by

$$\delta := \left\{ d \xrightarrow{\varepsilon} \varepsilon \;\middle|\; d \to \varepsilon \in R \right\} \cup \left\{ d \xrightarrow{\alpha} d' \;\middle|\; d \to \alpha\, d' \in R \right\}$$

It remains to show $\mathcal{L}(G) = \mathcal{L}(\phi(G))$.

($\subseteq$) We show $R \vdash d : s$ implies $\langle d, s, W\rangle \to_\delta^* \langle \varepsilon, \varepsilon, W\rangle$ for any W by induction on the derivation of $R \vdash d : s$. We have the following cases:

1. $d \to \varepsilon \in R$ and $R \vdash \varepsilon : s$. Then $s = \varepsilon$, $d \xrightarrow{\varepsilon} \varepsilon \in \delta$, and thus $\langle d, s, W\rangle \to_\delta \langle \varepsilon, \varepsilon, W\rangle$.
2. $d \to c\,d' \in R$ and $R \vdash c\,d' : s$. Then $d \xrightarrow{c} d' \in \delta$, $s = c\,s'$ and $R \vdash d' : s'$. Thus $\langle d, s, W\rangle \to_\delta \langle d', s', W\rangle$ and by IH $\langle d', s', W\rangle \to_\delta^* \langle \varepsilon, \varepsilon, W\rangle$.

3. $d \to f(d')\, d'' \in R$ and $R \vdash f(d')\, d'' : s$. Then $s = f(s')\, s''$, $R \vdash d' : s'$, and $R \vdash d'' : s''$. Thus

$$\begin{aligned} \langle d, s, W\rangle &\to_\delta \langle d', s', \langle d'', s''\rangle\, W\rangle \\ &\to_\delta^* \langle \varepsilon, \varepsilon, \langle d'', s''\rangle\, W\rangle && \text{by IH} \\ &\to_\delta \langle d'', s'', W\rangle \\ &\to_\delta \langle \varepsilon, \varepsilon, W\rangle && \text{by IH} \end{aligned}$$

($\supseteq$) We show $\langle d, s, W\rangle \to_\delta^n \langle \varepsilon, \varepsilon, W\rangle$ implies $R \vdash d : s$ by induction on n.
- $n = 0$: Then $s = \varepsilon$, $d = \varepsilon$ and hence $R \vdash d : s$.
- $d \xrightarrow{c} d' \in \delta$, $s = c\, s'$, and $\langle d, s, W\rangle \to_\delta \langle d', s', W\rangle \to_\delta^{n-1} \langle \varepsilon, \varepsilon, W\rangle$: By IH we have $R \vdash d' : s'$ and thus $R \vdash c\, d' : c\, s'$. Since $d \to c\, d' \in R$, we conclude $R \vdash d : s$.
- $d \xrightarrow{f(d')} d'' \in \delta$, $s = f(s')\, s''$, and

$$\begin{aligned} \langle d, s, W\rangle &\to_\delta \langle d', s', \langle d'', s''\rangle\, W\rangle \\ &\qquad \to_\delta^k \langle \varepsilon, \varepsilon, \langle d'', s''\rangle\, W\rangle \to_\delta \langle d'', s'', W\rangle \to_\delta^m \langle \varepsilon, \varepsilon, W\rangle \end{aligned}$$

with $k + m + 2 = n$. Then $d \to f(d')\, d'' \in R$. By IH we have $R \vdash d' : s'$ and $R \vdash d'' : s''$. Therefore $R \vdash d : f(s')\, s'' = s$. □

While our hedge automaton works top-down, that is, it reads the hedge from root to leaf, there is a bottom-up equivalent introduced by Murata [Mur99]:

Definition 11. *A* Murata hedge automaton *is $A = \langle \Sigma, Q, \iota, F, \Delta\rangle$ where $\iota \subseteq \Sigma_0 \times Q$ (initial relation), $F \subseteq Q^*$ is a regular set (of final states), and $\Delta \subseteq Q \times \Sigma_1 \times Q^*$ (transition relation) such that $\{w \in Q^* \mid \langle q, f, w\rangle \in \Delta\}$ is a regular language for each $q \in Q$ and $f \in \Sigma_1$. The relation $\to_A$ over $\mathcal{H}(\Sigma, Q)$ is defined as the congruence generated by $c \to_A q$ if $\langle c, q\rangle \in \iota$ and $f(w) \to_A q$ if $\langle q, f, w\rangle \in \Delta$. We say that a hedge $s \in \mathcal{H}(\Sigma)$ is* accepted *by A if $s \to_A^* w$ for some $w \in F$. The language $\mathcal{L}(A)$ of A is defined as the set of hedges accepted by A.*

Theorem 5. *A hedge language is recognized by a hedge automaton iff it is recognized by a Murata automaton.*

Proof. Thanks to Theorem 4 and Theorem 2, a hedge language is recognized by a hedge automaton iff it is regular. We prove that the latter is equivalent to being recognized by a Murata automaton.

($\Rightarrow$) Consider a transcription $\mu : \Sigma' \to \Sigma$ and family $\{L_x\}_{x \in \Sigma'_1 \cup \{\varepsilon\}}$ of regular languages. Define Murata automaton $A = \langle \Sigma, \Sigma', \iota, L_\varepsilon, \Delta\rangle$ by $\iota = \{\langle c^\mu, c\rangle \mid c \in \Sigma'_0\}$ and $\Delta = \{\langle f, f^\mu, w\rangle \mid f \in \Sigma'_1,\ w \in L_f\}$. It is easy to see that $\mu(\mathcal{G}(L_\_)) = \mathcal{L}(A)$.

($\Leftarrow$) Consider a Murata automaton $A = \langle \Sigma, Q, \iota, F, \Delta\rangle$. Define signature Σ' by $\Sigma'_i = Q \times \Sigma_i$, transcription $\mu : \Sigma' \to \Sigma$ by $\langle q, x\rangle^\mu = x$, and family $\{L_x\}_{x \in \Sigma'_1 \cup \{\varepsilon\}}$ by $L_\varepsilon = F$ and $L_{\langle q, f\rangle} = \{\langle q_1, f_1\rangle \ldots \langle q_n, f_n\rangle \mid \langle q, f, q_1 \ldots q_n\rangle \in \Delta\}$, which are regular. It is easy to prove that $\mathcal{L}(A) = \alpha(\mathcal{G}(L_\_))$. □

Compared to Murata's formulation, our top-down definition of hedge automata is more friendly for extension to push-down hedge automata and hedge transducers. Here we introduce a hedge version of stateless push-down automata. As in the string case, the difference to the hedge automata is that the target of each transition rule becomes a sequence of states (stack symbols). Similarly, each configuration has a sequence of states instead of a single state, replacing $Q^?$ by Q^*.

Definition 12 (push-down hedge automaton). *A* push-down hedge automaton (PDHA) *is $A = \langle \Sigma, Q, q_0, \delta \rangle$ where $\delta \subseteq Q \times \Sigma(Q)^? \times Q^*$. We define binary relation $\to_\delta$ over $Q^* \times \mathcal{H}(\Sigma) \times (Q^* \times \mathcal{H}(\Sigma))^*$ by*

1. *$q \xrightarrow{\varepsilon} v \in \delta$ implies $\langle q\,w, s, W \rangle \to_\delta \langle v\,w, s, W \rangle$;*
2. *$q \xrightarrow{c} v \in \delta$ implies $\langle q\,w, c\,s, W \rangle \to_\delta \langle v\,w, s, W \rangle$;*
3. *$q \xrightarrow{f(q')} v \in \delta$ implies $\langle q\,w, f(s)\,s', W \rangle \to_\delta \langle q', s, \langle v\,w, s' \rangle\, W \rangle$;*
4. *$\langle \varepsilon, \varepsilon, \langle w, s \rangle\, W \rangle \to_\delta \langle w, s, W \rangle$.*

As in the regular case, A accepts $s \in \mathcal{H}(\Sigma)$ if $\langle q_0, s, \varepsilon \rangle \to_\delta^* \langle \varepsilon, \varepsilon, \varepsilon \rangle$. The language $\mathcal{L}(A)$ that A recognizes is defined in the same way.

Theorem 6. *A hedge language is context-free iff it is recognized by a PDHA.*

Similar to Theorem 1, this theorem will be proved in the more general context of transducers in the next section.

5 Hedge Transducers

Here we define a hedge version of *translation grammars* and more operational push-down hedge transducers, and show that they are equally expressive. In this section we assume two signatures Σ^{i} and Σ^{o} for input and output, and define the signature Σ by $\Sigma_i = \left\{ f^{\mathsf{i}} \;\middle|\; f \in \Sigma^{\mathsf{i}}_i \right\} \cup \left\{ f^{\mathsf{o}} \mid f \in \Sigma^{\mathsf{o}}_i \right\}$ for $i = 0, 1$, where '$^{\mathsf{i}}$' and '$^{\mathsf{o}}$' are used to distinguish where the symbols are from.

Definition 13 (hedge translation grammars). *A* hedge translation grammar (HTG) *is $G = \left\langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, R \right\rangle$, where $d_0 \in D$, and $R \subseteq D \times \mathcal{H}(\Sigma, D)$ is a finite set of* translation rules. *Given $e \in \mathcal{H}(\Sigma, D)$, $s \in \mathcal{H}(\Sigma^{\mathsf{i}})$ and $t \in \mathcal{H}(\Sigma^{\mathsf{o}})$, we define $R \vdash e : s \hookrightarrow t$ by the following inference rules:*

$$\frac{}{R \vdash \varepsilon : \varepsilon \hookrightarrow \varepsilon} \qquad \frac{R \vdash e : s \hookrightarrow t \quad R \vdash e' : s' \hookrightarrow t'}{R \vdash e\,e' : s\,s' \hookrightarrow t\,t'} \qquad \frac{R \vdash e : s \hookrightarrow t}{R \vdash d : s \hookrightarrow t}\ \textit{if } d \to e \in R$$

$$\frac{}{R \vdash c^{\mathsf{i}} : c \hookrightarrow \varepsilon} \qquad \frac{R \vdash e : s \hookrightarrow t}{R \vdash f^{\mathsf{i}}(e) : f(s) \hookrightarrow t} \qquad \frac{}{R \vdash c^{\mathsf{o}} : \varepsilon \hookrightarrow c} \qquad \frac{R \vdash e : s \hookrightarrow t}{R \vdash f^{\mathsf{o}}(e) : s \hookrightarrow f(t)}$$

We say G generates *the hedge relation $\hookrightarrow_G \subseteq \mathcal{H}(\Sigma^{\mathsf{i}}) \times \mathcal{H}(\Sigma^{\mathsf{o}})$, where $s \hookrightarrow_G t :\Longleftrightarrow R \vdash d_0 : s \hookrightarrow t$. A* context-free hedge relation *is such a relation that is generated by some hedge translation grammar.*

Example 5. Consider hedge translation grammar $G_{\mathrm{md}\to\mathrm{xml}}$ consisting of the following translation rules:

$$\begin{array}{l}
\mathit{section_1} ::= \mathtt{\#}^{\mathsf{i}}\ \text{␣}^{\mathsf{i}}\ \mathrm{tree}^{\mathsf{o}}(\mathrm{title}^{\mathsf{o}}(\mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}})\ \mathrm{body}^{\mathsf{o}}(\mathit{lines})\ \mathit{section_2}^{*}) \\
\mathit{section_2} ::= \mathtt{\#}^{\mathsf{i}}\ \mathtt{\#}^{\mathsf{i}}\ \text{␣}^{\mathsf{i}}\ \mathrm{tree}^{\mathsf{o}}(\mathrm{title}^{\mathsf{o}}(\mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}})\ \mathrm{body}^{\mathsf{o}}(\mathit{lines})\ \mathit{section_3}^{*}) \quad \cdots
\end{array}$$

$$\begin{array}{rlrl}
\mathit{line} & ::= \varepsilon \mid \mathit{char}\ \mathit{line} \mid \mathit{element}\ \mathit{line} & \mathit{lines} & ::= \varepsilon \mid \mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}}\ \mathit{lines} \\
\mathit{element} & ::= \mathrm{br}^{\mathsf{i}}()\ \mathrm{br}^{\mathsf{o}}() \mid \mathrm{em}^{\mathsf{i}}(\mathrm{em}^{\mathsf{o}}(\mathit{lines})) \mid \cdots & \mathit{char} & ::= \text{␣}^{\mathsf{i}}\ \text{␣}^{\mathsf{o}} \mid \mathtt{a}^{\mathsf{i}}\ \mathtt{a}^{\mathsf{o}} \mid \mathtt{A}^{\mathsf{i}}\ \mathtt{A}^{\mathsf{o}} \mid \cdots
\end{array}$$

Following derivation illustrates how $G_{\mathrm{md}\to\mathrm{xml}}$ translates the hedge t_{md} of Example 1 to hedge t_{xml}.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{R \vdash \mathtt{\#}^{\mathsf{i}} : \mathtt{\#} \hookrightarrow \varepsilon} \quad \dfrac{}{R \vdash \text{␣}^{\mathsf{i}} : \text{␣} \hookrightarrow \varepsilon}}{R \vdash \mathtt{\#}^{\mathsf{i}}\ \text{␣}^{\mathsf{i}} : \mathtt{\#}\text{␣} \hookrightarrow \varepsilon} \quad \dfrac{\dfrac{\dfrac{\dfrac{\vdots}{R \vdash \mathit{line} : w_1 \hookrightarrow w_1} \quad \dfrac{}{R \vdash \backslash\mathrm{n}^{\mathsf{i}} : \backslash\mathrm{n} \hookrightarrow \varepsilon}}{R \vdash \mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}} : w_1\, \backslash\mathrm{n} \hookrightarrow w_1} \quad \dfrac{\vdots}{R \vdash \ldots : \ldots \hookrightarrow \ldots}}{R \vdash \mathrm{title}^{\mathsf{o}}(\mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}})\ldots : w_1\, \backslash\mathrm{n}\ldots \hookrightarrow \mathrm{title}(w_1)\ldots}}{R \vdash \mathrm{tree}^{\mathsf{o}}(\mathrm{title}^{\mathsf{o}}(\mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}})\ldots) : w_1\, \backslash\mathrm{n}\ldots \hookrightarrow \mathrm{tree}(\mathrm{title}(w_1)\ldots)}}{R \vdash \mathtt{\#}^{\mathsf{i}}\ \text{␣}^{\mathsf{i}}\ \mathrm{tree}^{\mathsf{o}}(\mathrm{title}^{\mathsf{o}}(\mathit{line}\ \backslash\mathrm{n}^{\mathsf{i}})\ldots) : \mathtt{\#}\text{␣}w_1\, \backslash\mathrm{n}\ldots \hookrightarrow \mathrm{tree}(\mathrm{title}(w_1)\ldots)}}{R \vdash \mathit{section_1} : \mathtt{\#}\text{␣}w_1\, \backslash\mathrm{n}\ldots \hookrightarrow \mathrm{tree}(\mathrm{title}(w_1)\ldots)}$$

In order to *translate* a hedge into another, it is not enough to check if $s \hookrightarrow_G t$ holds for given s and t, but one must compute such t (if exists) from given s. Therefore we provide such semantics using (push-down) hedge transducers. Compared to the automata case, configurations additionally maintain the current output, and worklist may contain tasks for output.

Definition 14 (push-down hedge transducers). *A* push-down hedge transducer (PDHT) *is $T = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, Q, q_0, \delta \rangle$ where $\delta \subseteq Q \times \Sigma(Q)^{?} \times Q^{*}$. The relation $\to_\delta$ over configurations in $Q^{*} \times \mathcal{H}(\Sigma^{\mathsf{i}}) \times \mathcal{H}(\Sigma^{\mathsf{o}}) \times \mathcal{W}^{*}$, where $\mathcal{W} = (Q \times \mathcal{H}(\Sigma^{\mathsf{i}})) \uplus (\Sigma^{\mathsf{o}} \times \mathcal{H}(\Sigma^{\mathsf{o}}) \times Q^{*})$, is defined as follows:*

1. *$q \xrightarrow{\varepsilon} v \in \delta$ implies $\langle q\, w, s, t, W \rangle \to_\delta \langle v\, w, s, t, W \rangle$;*
2. *$q \xrightarrow{c^{\mathsf{i}}} v \in \delta$ implies $\langle q\, w, c\, s, t, W \rangle \to_\delta \langle v\, w, s, t, W \rangle$;*
3. *$q \xrightarrow{f^{\mathsf{i}}(q')} v \in \delta$ implies $\langle q\, w, f(s)\, s', t, W \rangle \to_\delta \langle q', s, t, \langle v\, w, s' \rangle\, W \rangle$;*
4. *$\langle \varepsilon, \varepsilon, t, \langle w, s \rangle\, W \rangle \to_\delta \langle w, s, t, W \rangle$;*
5. *$q \xrightarrow{c^{\mathsf{o}}} v \in \delta$ implies $\langle q\, w, s, t, W \rangle \to_\delta \langle v\, w, s, t\, c, W \rangle$;*
6. *$q \xrightarrow{f^{\mathsf{o}}(q')} v \in \delta$ implies $\langle q\, w, s, t, W \rangle \to_\delta \langle q', s, \varepsilon, \langle f, t, v\, w \rangle\, W \rangle$;*
7. *$\langle \varepsilon, s, t', \langle f, t, w \rangle\, W \rangle \to_\delta \langle w, s, t\, f(t'), W \rangle$.*

The first four items correspond to those of push-down hedge automata, with additional argument t for the current output. Item *5* specifies to output a character at the end of the current output. Item *6* specifies to output a function symbol f; to specify how to output the arguments, it tells to move to state q' with fresh output, and push the task to write the resulting output as the argument of f, append to the previous output t, and proceed to the state stack $v\, w$. Item *7* specifies to do so when reaching the final state ε.

In the rest of the section, we prove that push-down hedge transducers characterize the context-free hedge relations. As in the string case, we first transform HTGs into Greibach normal forms. The classes of hedge grammars defined in Definition 2 are extended to hedge translation grammars in the obvious manner.

Theorem 7 (Greibach normal form). *Every HTG G has an equivalent Greibach normal form G', i.e., $\hookrightarrow_G = \hookrightarrow_{G'}$.*

Before the main transformation, we need to resolve the structure of grammars due to the hedge structures.

Definition 15 (shallow). *We say a context-free hedge (translation) grammar is* shallow *if all occurrences of $f(e)$ in the rules satisfy $e \in D$.*

Lemma 3. *Every HTG has an equivalent shallow HTG.*

Proof. Let $G = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, R \rangle$. We prove by induction on n_G, the number of occurrences of $f(e)$ with $e \notin D$ in the rules R. If $n_G = 0$, then G is already shallow. Otherwise, pick one such $f(e)$ and a fresh nonterminal d_e. We define $G' = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D \uplus \{d_e\}, d_0, R' \rangle$, where R' consists of the rules in R where all occurrences of $f(e)$ are replaced by $f(d_e)$, and additionally contains $d_e \to e$. It is easy to see that G' generates the same relation as G, and $n_{G'} < n_G$. Hence, we are done by IH. $\square$

The main step towards Greibach normal forms is the transformation to a realtime form. This transformation can be done as in the string case, but for completeness we present one.

Lemma 4. *Every HTG has an equivalent realtime shallow HTG.*

Proof. Thanks to Lemma 3, we consider shallow HTG $G = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, R \rangle$. Index nonterminals[3] $D = \{d_0, \dots, d_{|D|-1}\}$ and let $L(R) = \{d' \mid d \to d'e \in R\}$. We inductively construct R_i that satisfies invariant $L(R_i) \subseteq \{d_i, \dots, d_{|D|-1}\}$. Hence, $R_{|D|}$ is realtime, and the following transformations trivially preserve shallowness.

For the base case, just take $R_0 = R$. For the inductive case, consider R_i satisfying the invariant. We first remove rules of form $d \to d_i\, e$ with $d \neq d_i$. Denote the set of such rules by $X = \{d \to d_i\, e \in R_i \mid d \neq d_i\}$, and define

$$R' = R \setminus X \cup \{d \to e'e \mid d \to d_i\, e \in X,\ d_i \to e' \in R\}$$

Note that R' maintains the invariant. Moreover, $d \to d_i\, e \in R'$ only if $d = d_i$. Denote the set of those rules by $X' = \{d_i \to d_i\, e \in R'\}$. Let d_i' be a fresh nonterminal, and define

[3] For notational simplicity we index the initial nonterminal d_0 by 0, but obviously there is no need to do so.

$$R'' = R' \setminus X' \cup \{d_i \to e\, d'_i \mid d_i \to e \in R' \setminus X'\} \cup \\ \{d'_i \to e,\ d'_i \to e\, d'_i \mid d_i \to d_i\, e \in R',\ e \neq \varepsilon\}$$

The equivalence of R'' and R' is proved in the same way as in the string case. Note that the invariant may fail: $d'_i \to d_j\, e \in R''$ if $d_i \to d_i\, d_j\, e \in R'$, but now we can easily remove such rules. Let $X'' = \{d'_i \to d_j\, e \in R'' \mid j \leq i\}$ and define

$$R_{i+1} = R'' \setminus X'' \cup \{d'_i \to e' e \mid d'_i \to d_j\, e \in X'',\ d_j \to e' \in R''\}$$

It is easy to see that $L(R_{i+1}) = L(R_i) \setminus \{d_i\} \subseteq \{d_{i+1}, \ldots, d_{|D|-1}\}$. □

Proof (of Theorem 7). Thanks to Lemma 4, we only have to consider a realtime shallow HTG $G = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, R \rangle$. Let $X(R)$ denote the rules in R that do not satisfy the condition of Greibach. If $X(R) = \emptyset$, then G is already in Greibach normal form. Otherwise, pick $d \to e \in X(R)$. Due to the assumptions, e is of form $\alpha\, \beta_1 \ldots \beta_n$, where $\alpha \in \Sigma(D)$ and $\beta_i \in \Sigma(D) \cup D$. Let d_i be β_i if $\beta_i \in D$ and a fresh nonterminal otherwise. Define

$$R' = R \setminus \{d \to e\} \cup \{d \to \alpha\, d_1\, \ldots\, d_n\} \cup \{d_i \to \beta_i \mid i \leq n,\ \beta_i \notin D\}$$

It is clear to see $R \vdash e' : s \hookrightarrow t \iff R' \vdash e' : s \hookrightarrow t$. We see $X(R') \subsetneq X(R)$: Since G is shallow, $\beta_i \notin D$ implies $\beta_i \in \Sigma(D)$. Hence, repeating the process we reach to a Greibach normal form that generates the same hedge relation. □

We remark that shallowness is necessary for the above proof to work. If G is realtime but not shallow, then R' contains $d_i \to \beta_i$ for complex β_i, which is not necessarily realtime.[4]

We arrive at the final result of the section:

Theorem 8. *A hedge relation is context free iff it is recognized by a PDHT.*

Proof. Observe that the following mapping ϕ is bijective between Greibach normal forms and PDHTs: For translation grammar $G = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, R \rangle$ in Greibach normal form, PDHT $\phi(G) = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, \delta \rangle$ is defined by

$$\delta := \left\{ d \xrightarrow{\varepsilon} \varepsilon \;\middle|\; d \to \varepsilon \in R \right\} \cup \left\{ d \xrightarrow{\alpha} d_1 \ldots d_n \;\middle|\; d \to \alpha\, d_1 \ldots d_n \in R \right\}$$

where α ranges over $\Sigma(D)$. We conclude by showing $R \vdash d : s \hookrightarrow t \iff \langle d, s, \varepsilon, \varepsilon \rangle \to^*_\delta \langle \varepsilon, \varepsilon, t, \varepsilon \rangle$.

($\Rightarrow$) We show more generally that $R \vdash d : s \hookrightarrow t$ implies $\ell = \langle dw, s\, u, v, W \rangle \to^*_\delta \langle w, u, v\, t, W \rangle = r$ for any w, u, v, and W, by induction on the derivation of $R \vdash d : s \hookrightarrow t$. We consider the following cases:
 - $d \to \varepsilon \in R$ and $R \vdash \varepsilon : s \hookrightarrow t$. Then $s = t = \varepsilon$ and $d \xrightarrow{\varepsilon} \varepsilon \in \delta$. Hence $\ell = \langle d\, w, u, v, W \rangle \to_\delta \langle w, u, v, W \rangle = r$.

[4] One can consider transforming to realtime form in every iteration, but then the termination argument becomes more involved.

- $d \to c^{\mathsf{i}}\, d_1 \dots d_n \in R$ and $R \vdash c^{\mathsf{i}}\, d_1 \dots d_n : s \hookrightarrow t$. Then $d \xrightarrow{c^{\mathsf{i}}} d_1 \dots d_n \in \delta$, $s = c\, s_1 \dots s_n$, and $t = t_1 \dots t_n$ with $R \vdash d_i : s_i \hookrightarrow t_i$ for all i. We conclude the claim as follows:

$$\begin{aligned} \ell &\to_\delta \langle d_1 \dots d_n\, w, s_1 \dots s_n\, u, v, W\rangle \\ &\to_\delta^* \langle d_2 \dots d_n\, w, s_2 \dots s_n\, u, v\, t_1, W\rangle && \text{by IH for } R \vdash d_1 : s_1 \hookrightarrow t_1 \\ &\dots \\ &\to_\delta^* \langle w, u, v\, t_1 \dots t_n, W\rangle = r && \text{by IH for } R \vdash d_n : s_n \hookrightarrow t_n. \end{aligned}$$

- $d \to f^{\mathsf{i}}(d')\, d_1 \dots d_n \in R$ and $R \vdash f^{\mathsf{i}}(d')\, d_1 \dots d_n : s \hookrightarrow t$. Then $d \xrightarrow{f^{\mathsf{i}}(d')} d_1 \dots d_n \in \delta$, $s = f(s')\, s_1 \dots s_n$, and $t = t'\, t_1 \dots t_n$ with $R \vdash d' : s' \hookrightarrow t'$ and $R \vdash d_i : s_i \hookrightarrow t_i$ for all i. We conclude as follows:

$$\begin{aligned} \ell &\to_\delta \langle d', s', v, \langle d_1 \dots d_n\, w, s_1 \dots s_n\, u\rangle\, W\rangle \\ &\to_\delta^* \langle \varepsilon, \varepsilon, v\, t', \langle d_1 \dots d_n\, w, s_1 \dots s_n\, u\rangle\, W\rangle && \text{by IH for } R \vdash d' : s' \hookrightarrow t' \\ &\to_\delta \langle d_1 \dots d_n\, w, s_1 \dots s_n\, u, v\, t', W\rangle \to_\delta^* r && \text{same as above.} \end{aligned}$$

- $d \to c^{\mathsf{o}}\, d_1 \dots d_n \in R$ and $R \vdash c^{\mathsf{o}}\, d_1 \dots d_n : s \hookrightarrow t$. Then $d \xrightarrow{c^{\mathsf{o}}} d_1 \dots d_n \in \delta$, $s = s_1 \dots s_n$, and $t = c\, t_1 \dots t_n$ with $R \vdash d_i : s_i \hookrightarrow t_i$ for all i. We conclude $\ell \to_\delta \langle d_1 \dots d_n\, w, s_1 \dots s_n\, u, v\, a, W\rangle \to_\delta^* r$ where the last step is the same as above.
- $d \to f^{\mathsf{o}}(d')\, d_1 \dots d_n \in R$ and $R \vdash f^{\mathsf{o}}(d')\, d_1 \dots d_n : s \hookrightarrow t$. Then $d \xrightarrow{f^{\mathsf{o}}(d')} d_1 \dots d_n \in \delta$, $s = s'\, s_1 \dots s_n$, and $t = f(t')\, t_1 \dots t_n$ with $R \vdash d' : s' \hookrightarrow t'$ and $R \vdash d_i : s_i \hookrightarrow t_i$ for all i. We conclude:

$$\begin{aligned} \ell &\to_\delta \langle d', s\, u, \varepsilon, \langle f, v, d_1 \dots d_n\, w\rangle\, W\rangle \\ &\to_\delta^* \langle \varepsilon, s_1 \dots s_n\, u, t', \langle f, v, d_1 \dots d_n\, w\rangle\, W\rangle && \text{by IH for } R \vdash d' : s' \hookrightarrow t' \\ &\to_\delta \langle d_1 \dots d_n\, w, s_1 \dots s_n\, u, v\, f(t'), W\rangle \\ &\to_\delta^* \langle w, u, v\, f(t')\, t_1 \dots t_n, W\rangle = r && \text{same as above.} \end{aligned}$$

($\Leftarrow$) We show by induction on k that $\ell = \langle w, s\, u, v, W\rangle \to_\delta^k \langle \varepsilon, u, v\, t, W\rangle = r$ implies $R \vdash w : s \hookrightarrow t$, where $w \in D^*$. If $k = 0$ then $w = s = t = \varepsilon$ and the claim follows immediately. Otherwise, the following cases are possible.

1. $d \to \varepsilon \in R$, $w = d\, w'$, and $\ell \to_\delta \langle w', s\, u, v, W\rangle \to_\delta^{k-1} r$. Then by IH $R \vdash w' : s \hookrightarrow t$, and hence the claim follows.
2. $d \to c^{\mathsf{i}}\, w_1 \in R$, $w = d\, w_2$, $s = c\, s'$, and $\ell \to_\delta \langle w_1\, w_2, s'\, u, v, W\rangle \to_\delta^{k-1} r$. Then by IH, $R \vdash w_1\, w_2 : s' \hookrightarrow t$. So $R \vdash w_1 : s_1 \hookrightarrow t_1$ and $R \vdash w_2 : s_2 \hookrightarrow t_2$ with $s' = s_1\, s_2$ and $t = t_1\, t_2$. We conclude as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{R \vdash c^{\mathsf{i}} : c \hookrightarrow \varepsilon} \quad \dfrac{}{R \vdash w_1 : s_1 \hookrightarrow t_1}\,(\text{IH})}{R \vdash c^{\mathsf{i}}\, w_1 : c\, s_1 \hookrightarrow t_1}}{R \vdash d : c\, s_1 \hookrightarrow t_1} \qquad \dfrac{}{R \vdash w_2 : s_2 \hookrightarrow t_2}\,(\text{IH})}{R \vdash d\, w_2 : c\, s_1\, s_2 \hookrightarrow t_1\, t_2}$$

3. $d \to f^{\mathsf{i}}(d')\,w_1 \in R$, $w = d\,w_2$, $s = f(s')\,s''$, $t = t'\,t''$, and

$$\begin{aligned}\ell = \langle d\,w_2, f(s')\,s''\,u, v, W\rangle &\to_\delta \langle d', s', v, \langle w_1\,w_2, s''\,u\rangle\,W\rangle \\ \to_\delta^m &\ \langle \varepsilon, \varepsilon, v\,t', \langle w_1\,w_2, s''\,u\rangle\,W\rangle \\ \to_\delta &\ \langle w_1\,w_2, s''\,u, v\,t', W\rangle \to_\delta^{k-m-2} \langle \varepsilon, u, v\,t'\,t'', W\rangle = r\end{aligned}$$

   As $m, k-m-2 < k$, by IH we have $R \vdash d' : s' \hookrightarrow t'$ and $R \vdash w_1\,w_2 : s'' \hookrightarrow t''$. Hence we obtain $s'' = s_1\,s_2$ and $t'' = t_1\,t_2$ such that $R \vdash w_1 : s_1 \hookrightarrow t_1$ and $R \vdash w_2 : s_2 \hookrightarrow t_2$. We conclude similarly to the previous case.
4. $d \to c^{\circ}\,w_1 \in R$, $w = d\,w_2$, $t = c\,t'$, and $\ell \to_\delta \langle w_1\,w_2, s\,u, v\,c, W\rangle \to_\delta^{k-1} \langle \varepsilon, u, v\,c\,t', W\rangle = r$. Then the arguments are the same as case 2.
5. $d \to f^{\circ}(d')\,w_1 \in R$, $w = d\,w_2$, $s = s'\,s''$, $t = f(t')\,t''$, and

$$\begin{aligned}\ell = \langle w, s'\,s''\,u, v, W\rangle &\to_\delta \langle d', s'\,s''\,u, \varepsilon, \langle f, v, w_1\,w_2\rangle\,W\rangle \\ \to_\delta^m &\ \langle \varepsilon, s''\,u, t', \langle f, v, w_1\,w_2\rangle\,W\rangle \\ \to_\delta &\ \langle w_1\,w_2, s''\,u, v\,f(t'), W\rangle \to_\delta^{k-m-2} \langle \varepsilon, u, v\,f(t')\,t'', W\rangle = r\end{aligned}$$

   Then the arguments are the same as case 3. □

6 Implementation

Based on our theoretical development, we implemented an XML-processing tool `TXtruct`[5] in Java. The command line of `TXtruct` is as follows:

`java -jar txtr-`*version*`.jar` *txtr-file input output*

where *txtr-file* is an XML file representing a hedge translation grammar G, *input* is a file representing the input hedge s, and *output* is the file to which a hedge t such that $s \hookrightarrow_G t$ will be written. The grammar of *txtr-file* is as follows:

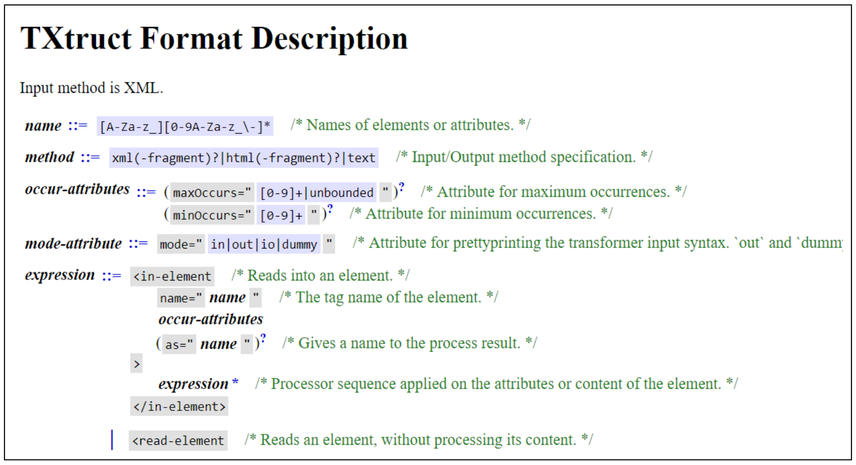
txtr-file ::= `<txtruct` (`method="`*method*`"`)$^?$`>`
`<input method="`*method*`"/>`$^?$ *class-declaration** *expression**
`</txtruct>`

The two *method*s indicate the output and input formats, in regular expression `xml(-fragment)`$^?$ | `html(-fragment)`$^?$ | `text`. A *class-declaration* of form

`<class name="`d`">`
`<option level="`ℓ_1`">`e_1`</option>`...`<option level="`ℓ_n`">`e_n`</option>`
`</class>`

defines a family $\{d_\ell\}_{\ell\in\mathbb{N}}$ of nonterminals with $d_\ell \to e_i \in R$ whenever $\ell \leq \ell_i$. The last sequence of *expressions* specifies $d_0 ::= e_1 \dots e_n$. Constructions of expressions are as follows:

[5] Available at `https://github.com/AkihisaYamada/TXtruct`.



TXtruct Format Description

Input method is XML.

```
name ::= [A-Za-z_][0-9A-Za-z_\-]*   /* Names of elements or attributes. */

method ::= xml(-fragment)?|html(-fragment)?|text   /* Input/Output method specification. */

occur-attributes ::= ( maxOccurs=" [0-9]+|unbounded " )?   /* Attribute for maximum occurrences. */
                     ( minOccurs=" [0-9]+ " )?   /* Attribute for minimum occurrences. */

mode-attribute ::= mode=" in|out|io|dummy "   /* Attribute for prettyprinting the transformer input syntax. `out` and `dumm

expression ::= <in-element   /* Reads into an element. */
                 name=" name "   /* The tag name of the element. */
                 occur-attributes
                 ( as=" name " )?   /* Gives a name to the process result. */
               >
                 expression *   /* Processor sequence applied on the attributes or content of the element. */
               </in-element>

             | <read-element   /* Reads an element, without processing its content. */
```

Fig. 3. The HTML obtained by transforming `txtr2html.txtr` by itself.

- `<group>`$e_1 \ldots e_n$`</group>` representing $e_1 \ldots e_n$.
- `<call ref="`d`" (level="`ℓ`")`$^?$`/>` representing nonterminal d (or d_ℓ).
- `<match pattern="`p`" (as="`x`")`$^?$`/>` representing p^{i}, but p is allowed to be a regular expression as defined by the Java regular expression library. One can save the match result by specifying attribute `as="`x`"`, and output the matched text by `<value-of select="`x`/match"/>`.
- `<in-element name="`f`">`e`</in-element>` representing $f^{\mathsf{i}}(e)$.
- `<text>`$c_1 \ldots c_n$`</text>` representing $c_1^{\mathsf{o}} \ldots c_n^{\mathsf{o}}$.
- `<element name="`f`">`e`</element>` representing $f^{\mathsf{o}}(e)$.

These elements can have *occurrence indicators* like XSD; e.g., `<call ref="`d`" minOccurs="0" maxOccurs="unbounded"/>` represents d^*.

The release of `TXtruct` contains a translation rule file `txtr2html.txtr`, whose input is a `TXtruct` rule file and output is an HTML prettyprint of the grammar defined by the input rule file. Hence, by defining a grammar in `TXtruct`, one obtains an HTML description of the grammar for free. Moreover, since `txtr2html.txtr` itself is a valid `TXtruct` rule file, it can transform itself and prettyprint the input format of `TXtruct` (see Fig. 3).

7 Conclusion

We have introduced context-free hedge grammars and translation grammars. We have proved that the class of hedge languages generated by regular hedge grammars corresponds to the existing class of regular hedge languages. We then introduced a top-down definition of hedge automata, which we also proved to be equivalent to the existing bottom-up definition. Our top-down definition led to the extension to push-down hedge automata and transducers. We proved that relationships among them hold as in the string case. Finally, based on the theoretical development we implemented the tool `TXtruct`. `TXtruct` has been incorporated into a document analysis tool `Tact` and used in industry.

For practical reasons, `TXtruct` implementation goes beyond the theory presented in this paper. For instance, `TXtruct` resolves nondeterminism in the manner of *parsing expression grammar* [For04]; has methods to read/write attributes and comments; allows calling nonterminals with parameters; allows cascading output into another transformation. We leave formalizing these details for future work. Finally, while our main focus was on translating documents, `TXtruct` can be used to validate if a document is in a hedge language. We expect our work to be a basis for developing techniques to verify that a given hedge translation grammar always results in a desired hedge language.

Acknowledgments The first author would like to thank Hitoshi Ohsaki and Ken'ichi Handa.

References

BPS61. Yehoshua Bar-Hillel, Micha Perles, and Eli Shamir. On formal properties of simple phrase-structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft, und Kommunikationsforschung*, 14(2):143–172, 1961.

CBM19. Stefano Crespi-Reghizzi, Luca Breveglieri, and Angelo Morzenti. *Formal Languages and Compilation, Third Edition.* Texts in Computer Science. Springer, 2019.

Dom14. Massimiliano Dominici. An overview of pandoc. *TUGboat*, 35(1):44–50, 2014.

For04. Bryan Ford. Parsing expression grammars: a recognition-based syntactic foundation. In *POPL 2004*, pages 111–122, 2004.

HOT17. Kenichi Handa, Hitoshi Ohsaki, and Izumi Takeuti. Security Requirements Analysis Supporting Tool: TACT. In *Proceeding of the Information Processing Society of Japan (IPSJ) SIGSE Winter Workshop*, 2017.

Kay21. Michael Kay. XSL transformations (XSLT) version 2.0 (second edition). W3C recommendation, W3C, March 2021. https://www.w3.org/TR/2021/REC-xslt20-20210330/.

Leo16. Sean Leonard. The text/markdown Media Type. RFC 7763, Internet Engineering Task Force (IETF), 2016.

MT06. Yasuhiko Minamide and Akihiko Tozawa. XML Validation for Context-Free Grammars. In Naoki Kobayashi, editor, *APLAS 2006*, volume 4279 of *LNCS*, pages 357–373. Springer, 2006.

MTSM$^+$12. Noah Mendelsohn, Henry Thompson, Michael Sperberg-McQueen, Murray Maloney, Sandy Gao, and David Beech. W3C xml schema definition language (XSD) 1.1 part 1: Structures. W3C recommendation, W3C, April 2012. https://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/.

Mur99. Makoto Murata. Hedge automata: a formal model for XML schemata. `http://www.xml.gr.jp/relax/hedge_nice.html`, 1999.

PQ68. C. Pair and A. Quere. Définition et Étude des Bilangages Réguliers. *Information and Control*, 13(6):565–593, December 1968.

Rou69. William C. Rounds. Context-free grammars on trees. In Patrick C. Fischer, Seymour Ginsburg, and Michael A. Harrison, editors, *STOC 1969*, pages 143–148. ACM, 1969.

A Omitted Proofs

Lemma 5. *Every HTG G has an equivalent shallow HTG G^{nlr} without direct left recursion, i.e., rules of the form $d \to d\,e$.*

Proof. Thanks to Lemma 3, we start with a shallow HTG $G^{\mathsf{s}} = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D, d_0, R \rangle$. Let $L(d)$ be the set of l-recursive rules in R. If $L(d) = \emptyset$ for every $d \in D$ then we are done. Otherwise, pick d with $L(d) \neq \emptyset$. Let d' be a fresh nonterminal, and define $G' = \langle \Sigma^{\mathsf{i}}, \Sigma^{\mathsf{o}}, D \uplus \{d'\}, d_0, R' \rangle$, where

$$\begin{aligned} R' = R \setminus L(d) \cup \{d \to e\,d' \mid d \to e \in R \setminus L(d)\} \cup \\ \{d' \to e,\ d' \to e\,d' \mid d \to d\,e \in L(d),\ e \neq \varepsilon\} \end{aligned}$$

Repeatedly applying the process we can remove all l-recursive rules. It is clear that G' is shallow as G^{s} is. It remains to show $R \vdash e : s \hookrightarrow t \iff R' \vdash e : s \hookrightarrow t$.

($\Rightarrow$) By induction on the derivation. The only nontrivial case is $R \vdash d\,e : s \hookrightarrow t$ with $d \to d\,e \in L(d)$. We show that this implies $R' \vdash e'\,d' : s \hookrightarrow t$ for some $d \to e' \in R \setminus L(d)$; then we easily conclude $R' \vdash d : s \hookrightarrow t$ as $d \to e'\,d' \in R'$. We prove the new claim by induction on the derivation of $R \vdash d\,e : s \hookrightarrow t$. We know that s and t are split into $s = s'\,s''$ and $t = t'\,t''$ such that $R \vdash d : s' \hookrightarrow t'$ and $R \vdash e : s'' \hookrightarrow t''$. We have $R' \vdash e : s'' \hookrightarrow t''$ by outer IH. We do case analysis.

- Suppose that there exists $d \to d\,e' \in L(d)$ with $R \vdash d\,e' : s' \hookrightarrow t'$. Then inner IH gives $d \to e'' \in R \setminus L(d)$ such that $R' \vdash e''\,d' : s' \hookrightarrow t'$. Then we know s' and t' are split into $s' = u\,u'$ and $t' = v\,v'$, such that $R' \vdash e'' : u \hookrightarrow v$ and $R' \vdash d' : u' \hookrightarrow v'$. Since $d' \to e''\,d' \in R'$, we derive $R' \vdash d' : u'\,s'' \hookrightarrow v'\,t''$. We conclude $R' \vdash e''\,d' : u\,u'\,s'' \hookrightarrow v\,v'\,t''$, i.e., $R' \vdash e''\,d' : s \hookrightarrow t$.
- Otherwise, we must have $d \to e' \in R \setminus L(d)$ and $R \vdash e' : s' \hookrightarrow t'$. We conclude $R' \vdash e'\,d' : s \hookrightarrow t$, since $R' \vdash d' : s'' \hookrightarrow t''$ as $d' \to e \in R'$.

($\Leftarrow$) By induction on the derivation. The only nontrivial case is $R' \vdash d : s \hookrightarrow t$ due to $R' \vdash e\,d' : s \hookrightarrow t$ for $d \to e \in R \setminus L(d)$. Then we have $R' \vdash e : s' \hookrightarrow t'$ and $R' \vdash d' : s'' \hookrightarrow t''$ for $s = s'\,s''$ and $t = t'\,t''$. By IH we have $R \vdash e : s' \hookrightarrow t'$, and hence $R \vdash d : s' \hookrightarrow t'$ as $d \to e \in R$. We conclude by showing that $R \vdash d : s' \hookrightarrow t'$ and $R' \vdash d' : s'' \hookrightarrow t''$ implies $R \vdash d : s'\,s'' \hookrightarrow t'\,t''$ by induction on the derivation of $R' \vdash d' : s'' \hookrightarrow t''$.

- Suppose that $R' \vdash e' : s'' \hookrightarrow t''$ for $d \to d\,e' \in L(d)$. By outer IH we have $R \vdash e' : s'' \hookrightarrow t''$. With $R \vdash d : s' \hookrightarrow t'$ we conclude $R \vdash d : s'\,s'' \hookrightarrow t'\,t''$.
- Suppose that $R' \vdash e'\,d' : s'' \hookrightarrow t''$ for $d \to d\,e' \in L(d)$. Then we have $R' \vdash e' : u \hookrightarrow v$ and $R' \vdash d' : u' \hookrightarrow v'$ for $s'' = u\,u'$ and $t'' = v\,v'$. By outer IH we have $R \vdash e' : u \hookrightarrow v$, and with inner IH, $R \vdash d : s'\,u \hookrightarrow t'\,v$. Since $d \to d\,e' \in R$, we conclude $R \vdash d : s'\,u\,u' \hookrightarrow t'\,v\,v'$. □

B On Context-Free Word Grammars Generating Matched Tags

This section studies context-free word grammars that generate matched tags and proves Lemma 2.

Let $G = (\mathcal{N}, W, \mathcal{R})$ be a context-free word grammar over a paired alphabet $\hat{\Sigma} = \Sigma_0 \cup \{\acute{f}, \grave{f} \mid f \in \Sigma_1\}$. For simplicity, we assume that $\Sigma_0 = \emptyset$ in this section. This does not essentially affect the following argument: one can consider another signature Σ' with $\Sigma'_0 = \emptyset$ and $\Sigma'_1 = \Sigma_0 \cup \Sigma_1$ and replace c with $\acute{c}\grave{c}$.

We assume that (1) every non-terminal $N \in \mathcal{N}$ is reachable (i.e. $W \to^* \alpha N \beta$ for some sequences α and β of terminals and non-terminals) and (2) every non-terminal produces a word. This assumption does not lose generality since we can simply remove a non-terminal that violates the above requirement without changing the generated language.

A word w over $\hat{\Sigma}$ is *matched* if $w = \widehat{s}$ for some hedge s. We assume that $L(G) = \widehat{H}$ for some hedge language H, so $w \in L(G)$ implies w is matched. The following argument is a slightly more detailed version of Minamide and Tozawa's analysis [MT06] of matched words and context-free grammars.

A substring w of a matched word $v = v_0 w v_1$ can be canonically decomposed as follows:

$$w \quad = \quad m_0 \grave{f}_1 m_1 \grave{f}_2 m_2 \dots m_{n-1} \grave{f}_n m_n \acute{g}_1 m_{n+1} \dots m_{n+k-1} \acute{g}_k m_{n+k}$$

for some $n, k \geq 0$, where $m_0, \dots, m_{n+k}$ are matched words. We call $\grave{f}_1 \dots \grave{f}_n \acute{g}_1 \dots \acute{g}_k$ the *shape of* w and write as $\sharp w$.

Lemma 6. *Let $G = (\mathcal{N}, W, \mathcal{R})$ be a context-free word grammar generating only matched words. For a $N \in \mathcal{N}$, the shapes of words generated by N are finite, i.e.,*

$$\{\sharp w \mid w \in L(N)\} \textit{ is a finite set.}$$

Proof. Since N is reachable, $W \to^* \alpha N \beta$ for some sequences α, β of terminals and non-terminals. By rewriting the non-terminals in α and β, one obtains $W \to^* vNu$. The shape of v and u must be

$$\sharp v = \acute{f}_1 \acute{f}_2 \dots \acute{f}_n \qquad \sharp u = \grave{g}_k \grave{g}_{k-1} \dots \grave{g}_1$$

since $w \in L(N)$ for some w and vwu is matched. For every $w \in L(N)$, the word vwu is matched. This means that

$$\sharp w \in \{\grave{f}_n \grave{f}_{n-1} \dots \grave{f}_i \acute{g}_i \acute{g}_{i+1} \dots \acute{g}_k \mid i \leq n, i \leq k, f_j = g_j \text{ for every } j = 1, \dots, i-1\}.$$

So $\{\sharp w \mid w \in L(N)\}$ has at most n elements. □

Let $\sharp N = \{\sharp w \in w \in L(N)\}$.

Lemma 7. *Let $G = (\mathcal{N}, W, \mathcal{R})$ be a context-free word grammar generating only matched words. There exists a grammar $G' = (\mathcal{N}', W', \mathcal{R}')$ that satisfies the following conditions:*

– $L(G') = L(G)$.
– *G' is in Chomsky normal form.*
– *$\sharp N'$ is a singleton for every non-terminal $N \in \mathcal{N}$.*

Proof. For simplicity, we assume that G is in Chomsky normal form. Of course, this does not lose generality. The non-terminals of G' is given by $\{N^{(\rho)} \mid N \in \mathcal{N}, \rho \in \sharp N\}$. This is a finite set by Lemma 6. We aim to provide rules such that

$$L(N^{(\rho)}) = L(N) \cap \{w \mid \sharp w = \rho\}.$$

The rules are given as follows:

– If $(N \to N_1 N_2) \in \mathcal{R}$, then $(N^{(\rho)} \to N_1^{(\rho_1)} N_2^{(\rho_2)}) \in \mathcal{R}'$ for $\rho_1 \in \sharp N_1$ and $\rho_2 \in \sharp N_2$ with $\rho = \sharp(\rho_1 \rho_2)$.
– If $(N \to a) \in \mathcal{R}$ and $a = \rho$, then $(N^{(\rho)} \to a) \in \mathcal{R}'$.
– If $(W \to \varepsilon) \in \mathcal{R}$, then $(W^{(\varepsilon)} \to \varepsilon) \in \mathcal{R}'$.

The start symbol is $W^{(\varepsilon)}$. It is easy to see that G' has the required properties. □

For a grammar satisfying the conditions in the previous lemma, we write $\sharp N$ for the unique shape of $w \in L(N)$. If $\sharp N = \grave{f}_1 \dots \grave{f}_n \acute{g}_1 \dots \acute{g}_k$, then $\mathfrak{c}(N) = \grave{f}_1 \dots \grave{f}_n$ and $\mathfrak{o}(N) = \acute{g}_1 \dots \acute{g}_k$. We write $|\mathfrak{c}(N)|$ and $|\mathfrak{o}(N)|$ for their lengths.

Proof (of Lemma 2). Let H be a hedge language and assume that $\widehat{H}$ is context free. Then there exists a context-free grammar $G = (\mathcal{N}, W, \mathcal{R})$ that generates $\widehat{H}$. By Lemma 7, we can assume without loss of generality that G satisfies the conditions in Lemma 7.

For a non-terminal $N \in \mathcal{N}$ and a prefix α of $\sharp N$, we define $L(N)^{(\alpha)}$ as the word language defined as

$$\bigcup \left\{ \mathsf{path}(s) \;\middle|\; v\widehat{s}v' \in L(N),\;\; \alpha = \sharp v,\;\; \sharp N = (\sharp v)(\sharp v') \right\}.$$

Note that the condition $\sharp N = (\sharp v)(\sharp v')$ means that no tag in $\sharp v$ matches a tag in $\sharp v'$. In other words, either $\sharp v$ has no open tag or $\sharp v'$ has no close tag (or both). Then the path language $\mathsf{path}(H)$ is equivalent to $L(W)^{(\varepsilon)}$.

We give a grammar consisting of non-terminals generating $L(N)^{(\alpha)}$. Let

$$\mathcal{N}' = \{N^{(\alpha)} \mid N \in \mathcal{N},\ \alpha \text{ is a prefix of } \sharp N\ \}.$$

The start symbol is $W^{(\varepsilon)}$. The rules are given as follows. Let $N \in \mathcal{N}$ and α be a prefix α of $\sharp N$. Assume $\alpha = \alpha_C \alpha_O$, where α_C and α_O consist of close and open tags, respectively. The rules for the non-terminal $N^{(\alpha)}$ are given as follows:

– $(N^{(\alpha)} \to \varepsilon) \in \mathcal{R}'$.
– Assume $(N \to N_1 N_2) \in \mathcal{R}$.
  - If $\alpha \acute{f}_1 \dots \acute{f}_k$ is a prefix of $\sharp N_1$, then

$$(N^{(\alpha)} \to f_1 \dots f_k N_1^{(\alpha \acute{f}_1 \dots \acute{f}_k)}) \in \mathcal{R}'.$$

- If $\sharp N_1 = \alpha \acute{f}_1 \dots \acute{f}_k$ and $\grave{f}_k \dots \grave{f}_{i+1}$ is a prefix of $\sharp N_2$, then

$$(N^{(k)} \to f_1 \dots f_i N_2^{(\grave{f}_k \dots \grave{f}_{i+1})}) \in \mathcal{R}'.$$

- If β is a prefix of $\sharp N_2$ and $\alpha = \sharp((\sharp N_1)\beta)$, then

$$(N^{(k)} \to N_2^{(\beta)}) \in \mathcal{R}'.$$

It is not difficult to see that $L(N^{(k)}) = L(N)^{(k)}$. Since the above rules are of the form $M \to \boldsymbol{g} M'$ where M, M' are non-terminals and $\boldsymbol{g}$ is a (possibly empty) sequence of terminals, these rules actually generate a regular language.

We prove $L(N)^{(\alpha)} \subseteq L(N^{(\alpha)})$. Let $p \in L(N)^{(\alpha)}$. Then $w = v\widehat{s}v' \in L(N)$ and $\alpha = \sharp v$ and $\sharp N = (\sharp v)(\sharp v')$. We prove $p \in L(N^{(\alpha)})$ by induction on the number of production rules to derive $w \in L(N)$. If the length is 1, the production must be $N \to \grave{f}$ or $N \to \acute{f}$. So $p = \varepsilon$ and we have $N^{(\alpha)} \to \varepsilon$. Assume that $N \to N_1 N_2 \to^* v\widehat{s}v'$.

– Case $v = v_1 v_2$ and $v_1 \in L(N_1)$ and $v_2 \widehat{s} v' \in L(N_2)$: Then $\beta_1 \beta_2 = \sharp v$, $\sharp v_1 = \sharp N_1 = \beta_1 \acute{f}_1 \dots \acute{f}_n$ and $\sharp v_2 = \grave{f}_n \dots \grave{f}_1 \beta_2$ for some $\beta_1, \beta_2, f_1, \dots, f_n$. Since there is no match in $(\sharp v)(\sharp v')$, there is no match in $\beta_2(\sharp v')$, hence $(\sharp v_2)(\sharp v')$ has no match. Therefore $(\sharp v_2)(\sharp v') = \sharp N_2$. Then $p \in L(N_2^{(\sharp v_2)})$ by the IH, so $N^{(\alpha)} \to N_2^{(\sharp v_2)} \to^* p$.

– Case $v' = v'_1 v'_2$ and $v\widehat{s}v'_1 \in L(N_1)$ and $v'_2 \in L(N_2)$: Then $\beta'_1 \beta'_2 = \sharp v'$, $\sharp v'_1 = \beta'_1 \acute{f}_1 \dots \acute{f}_n$ and $\sharp v'_2 = \sharp N_2 = \grave{f}_n \dots \grave{f}_1 \beta_2$ for some $\beta'_1, \beta'_2, f_1, \dots, f_n$. Since there is no match in $(\sharp v)(\sharp v')$, there is no match in $(\sharp v)\beta'_1$, hence $(\sharp v)(\sharp v'_1)$ has no match. So $(\sharp v)(\sharp v'_1) = \sharp N_1$. By the IH, $p \in L(N_1^{(\alpha)})$. Then $N^{(\alpha)} \to N_1^{(\alpha)} \to^* p$.

– Case $\widehat{s} = s_1 s_2$ and $v s_1 \in L(N_1)$ and $s_2 v' \in L(N_2)$: We can assume without loss of generality that $s = f(s')$ (by choosing the component that the path p belongs to). Since $\sharp(s_1 s_2) = \varepsilon$, $\sharp s_1$ consists only of open tags and $\sharp s_2$ consists only of close tags. So

$$\begin{aligned} s_1 &= \acute{f}\widehat{t_0}\acute{f}_1\widehat{t_1} \dots \widehat{t_{k-1}}\acute{f}_k\widehat{t_k} \\ s_2 &= \widehat{t'_k}\grave{f}_k\widehat{t'_{k-1}} \dots \widehat{t'_1}\grave{f}_1\widehat{t'_0}\grave{f}. \end{aligned}$$

A tail of the path p belongs to one of $t_0, t_1, \dots, t_k, t'_k, \dots, t'_1, t'_0$.

- Assume that the tail of the path belongs to t_i. Then $p = f f_1 \dots f_i p'$ for some $p' \in \mathsf{path}(t_i)$. Since $(\sharp v)\acute{f}\acute{f}_1 \dots \acute{f}_i$ is a prefix of $\sharp N_1$ and $\sharp(\acute{f}_{i+1}\widehat{t_{i+1}} \dots \widehat{t_{k-1}}\acute{f}_k\widehat{t_k}) = \acute{f}_{i+1} \dots \acute{f}_k$, we have $p' \in L(N_1^{(\alpha \acute{f}\acute{f}_1 \dots \acute{f}_i)})$. We have $N^{(\alpha)} \to f f_1 \dots f_i N_1^{(\alpha \acute{f}\acute{f}_1 \dots \acute{f}_i)} \to^* f f_1 \dots f_i p'$.
- Assume that the tail of the path belongs to t'_i. Then $p = f f_1 \dots f_i p'$ for some $p' \in \mathsf{path}(t'_i)$. Since $\grave{t}_i \dots \grave{f}_1 \grave{f}(\sharp v')$ is a suffix of $\sharp N_2$ and $\sharp(\widehat{t'_k}\grave{f}_k\widehat{t'_{k-1}} \dots \widehat{t'}_{i+1}\grave{f}_{i+1}) = \grave{f}_k \dots \grave{f}_{i+1}$, we have $p' \in L(N_2^{(\grave{f}_k \dots \grave{f}_{i+1})})$. We have $N^{(\alpha)} \to f f_1 \dots f_i N_2^{(\grave{f}_k \dots \grave{f}_{i+1})} \to^* f f_1 \dots f_i p'$.

The inclusion of the other direction can be easily proved. $\square$