Question 2 : Principe de remodelage expliqué

Style de paragraphe

Rubrique	Style_rubrique	<label type="rubrique">…</label>
Paragraphe	Style_paragraphe	…
Strophe	Style_strophe	<lg> </lg>

Styles de caractère

Abréviations		<choice> <abbr type="contraction">ã</abbr> <expan>an</expan> </choice>
• par contraction	style_abréviation_contraction	
• par signe spécial	style_abréviation_signe_spéciaux	<choice> <abbr type="signeSpecial">9</abbr> <expan>us</expan> </choice>
Résolution	Style_abréviation_résolue	
Remplacement du & en et	Style_initial_&	<choice> <abbr type="remplacementEt">&</abbr> <expan>et</expan> </choice>
	Style_remplacement_&	
Lettres ramistes		<choice change="lettreRamiste"> <orig>v</orig> <reg>u</reg> </choice>
• version fac-similaire	**Style_lettre_ramiste**	
• version normalisée	**Style_remplacement_lettre_ramiste**	
S long		
- version fac-similaire	**style_s_long**	<choice change="sLong"> <orig>ſ</orig> <reg>s</reg> </choice>
- version normalisée	**Style_remplacement_s_long**	
Signes diacritiques		<choice change="signesDiacritiques"> <orig>e</orig> <reg>é</reg> </choice>
• e original	Style_e_initial	
• é normalisé	Style_ajout_accent_aigu	
Corrections éditoriales		
• Ajout	Style_ajout_editeur	<choice > <corr resp="editor"> r </corr>

		</choice>
Ajout c cédille	style_c_inital **style_ajout_cedille**	<choice change="normalisationC"> <orig>c</orig> <reg>ç</reg> </choice>
Ajout de cédille ou d'apostrophe (désagglutination)	**ajout_apostrophe**	<choice change="desagglutination"> <reg>'</reg></choice>
Ajout espace	Style_séparation_mot	<choice change="desagglutination"> <reg> </reg> </choice>
• Ajout ponctuation	Style_ajout_ponctuation	<pc type="PONfrt"> <choice> <reg>.</reg> </choice> </pc>
• Marque de césure originale	**Style_cesure**	<pc type="marqueCesure"> <choice> <orig>-</orig> </choice> </pc>
Éléments structurels		
• Initiale	*Style_lettrine*	<seg type="initial">O</seg>
• Début de vers	Style_debut_vers	<l>
• Fin de ligne	style_fin_de_ligne	<lb/>
• Fin de vers	Style_fin_de_vers	</l>
Traitement de l'évolution linguistique		
Terminaison imparfait	Style_imparfait_moyen_fr	<m type="desinence">oit</m>
Terminaison en z ou g	**Style_trace_cas**	<m type="diacritique">g z</m>
Terminaison lx	**Style_lx**	<m type="pluriel">lx</m>

Le passage en TEI

```
<xsl:stylesheet version="2.0" xmlns="http://www.tei-c.org/ns/1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
    xmlns:rels="http://schemas.openxmlformats.org/package/2006/relationships"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
    xmlns:sml="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
    xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:dcterms="http://purl.org/dc/terms/" xmlns:dcmitype="http://purl.org/dc/dcmitype/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tei-spreadsheet="https://github.com/oucs/tei-spreadsheet"
    xmlns:fn="http://www.w3.org/2005/xpath-functions" exclude-result-prefixes="#all">
```

L'élément racine <document> est transformé en élément <TEI> afin de se conformer au système d'encodage de l'XML-TEI auquel le document est soumis par l'usage de <xsl:stylesheet> en haut du document.

Les styles de paragraphes

```
<!-- style_de_paragraphe -->
<xsl:template match="style_rubrique">
    <xsl:element name="label">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">rubrique</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
<xsl:template match="style_paragraphe">
    <xsl:element name="p">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
<xsl:template match="style_strophe">
    <xsl:element name="lg">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
<!-- Fin_des_styles_de_paragraphe -->
```

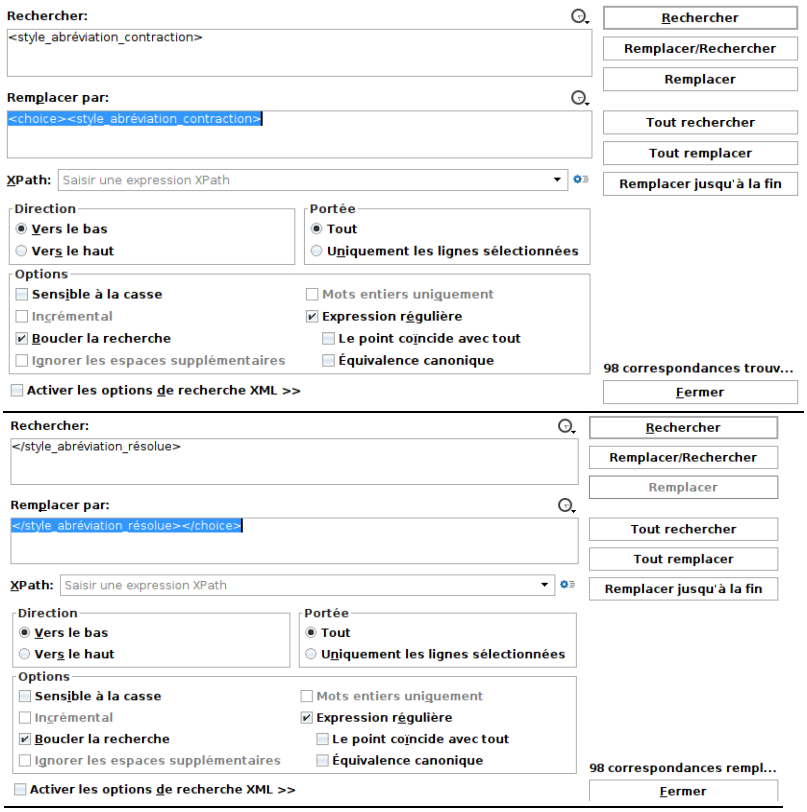
Les styles de paragraphe sont au nombre de trois et sont rendus conformes à la TEI par des <xsl:template match=" « >. Ainsi le style_rubrique devient <label> auquel nous ajoutons

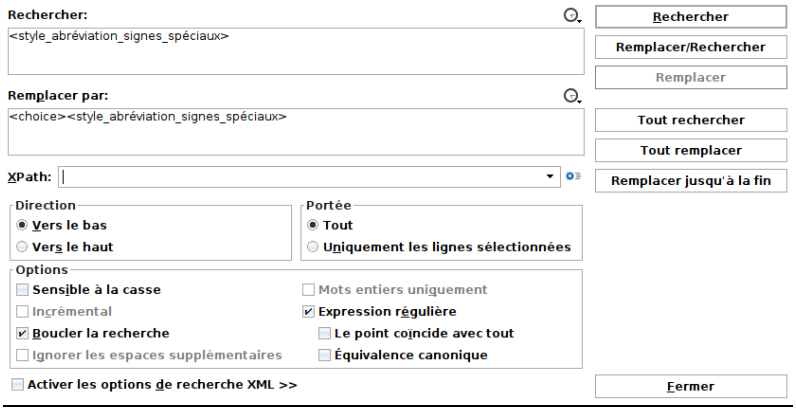
un @type afin de préciser qu'il s'agit bien d'une rubrique. Le style_paragraphe correspond à <p> tandis que le style_strophe se transforme en <lg>. L'utilisation de template est ici pertinente car son usage permet de systématiser le traitement qui ne possède pas de nuance particulière qu'il faudrait rendre à l'aide de rechercher-remplacer ou d'expression régulière.

Les styles de caractères

Les abréviations

Les abréviations par contractions et par signes spéciaux avec leurs abréviations résolues



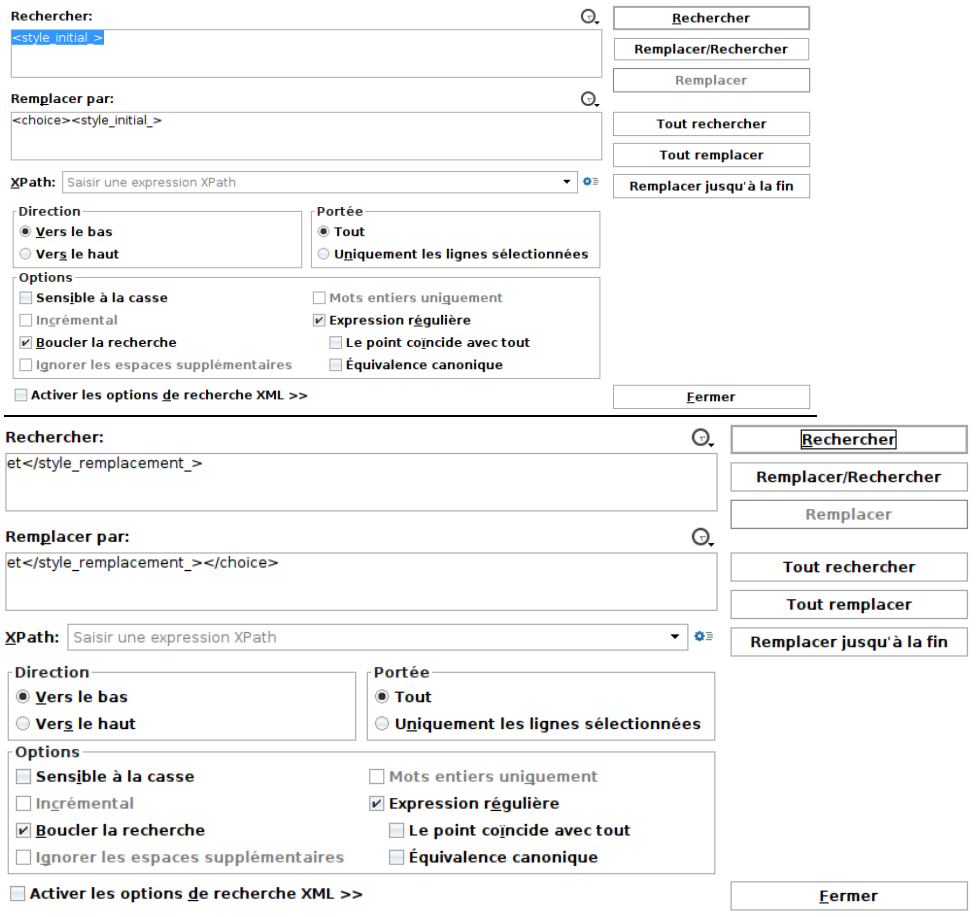



```
<xsl:template match="style_abréviation_contraction">
    <xsl:element name="abbr">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">contraction</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="style_abréviation_résolue">
    <xsl:element name="expan">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="style_abréviation_signes_spéciaux">
    <xsl:element name="abbr">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">signeSpecial</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

À l'aide de l'outil Rechercher/Remplacer nous avons placé une balise <choice> avant la balise ouvrante <style_abréviation_signes_spéciaux> et <style_abréviation_contraction> et une balise </choice> après le </style_abréviation_résolue>.

À l'aide de l'outil Rechercher/Remplacer nous avons placé une balise <choice> avant la balise ouvrante <style_abréviation_signes_spéciaux> et <style_abréviation_contraction> et une balise </choice> après le </style_abréviation_résolue>. Le style_abréviation_résolue est remplacé dans l'xsl par le nom d'élément <expan> car il s'agit de l'expansion de l'abréviation tandis que les deux autres styles sont transformés en <abbr> car ce sont des abréviations. De plus, nous les spécifions par un @type précisant leur nature, soit « contraction » soit « signeSpecial ».

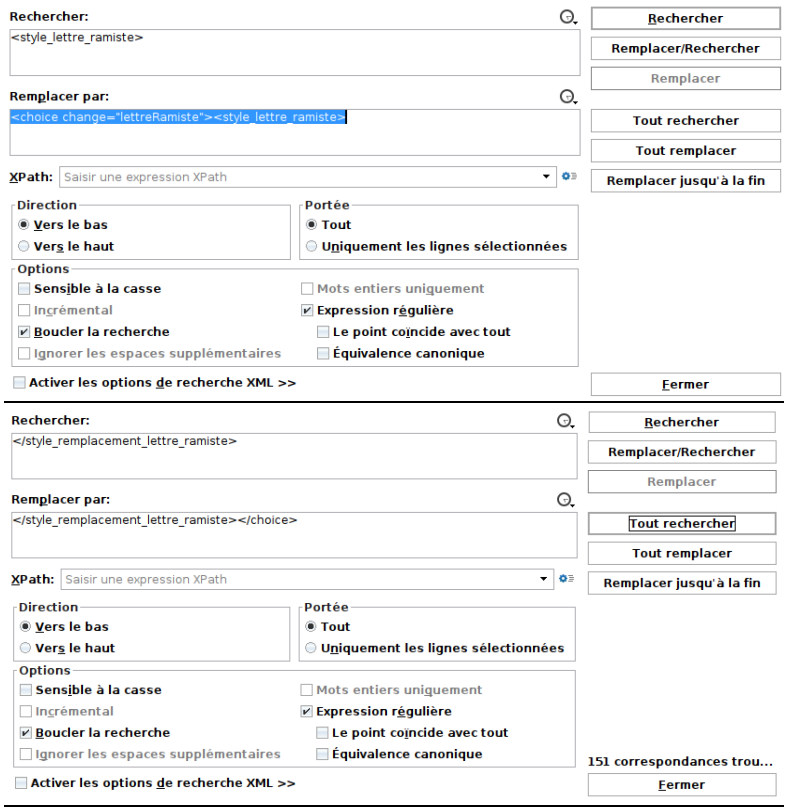
Le remplacement de & par et




```
<xsl:template match="style_initial_">
    <xsl:element name="abbr">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">remplacementET</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="style_remplacement_">
    <xsl:element name="expan">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Le signe « & » ne peut pas être intégré dans une balise, nos styles le concernant sont donc nommés style_initial_ et style_remplacement_ . Dans le fichier xsl le style_initial_ se transforme en <abbr> car c'est une abréviation tandis que le style_remplacement_ est balisé par <expan> car il change l'abréviation par le mot complet. Dans le fichier xml, le <style_initial_> et <style_remplacement_> sont placés dans un même <choice>.

Les lettres ramistes




```
<xsl:template match="style_lettre_ramiste">
    <xsl:element name="orig">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="style_remplacement_lettre_ramiste">
    <xsl:element name="reg">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Nous avons ajouté, par un Rechercher/Remplacer, un <choice change="lettreRamiste"> précédant la balise ouvrante <style_lettre_ramiste>, <choice> qui se ferme après la balise fermante </style_replacement_lettre_ramiste> afin de regrouper dans un même <choice> la lettre ramiste et son remplacement. Dans le fichier xsl le style_lettre_ramiste est remplacé par <orig> car il style la lettre ramiste présente à l'origine dans le texte tandis que le style_remplacement_lettre_ramiste se transforme en <reg> car ce remplacement intervient lors de la normalisation.

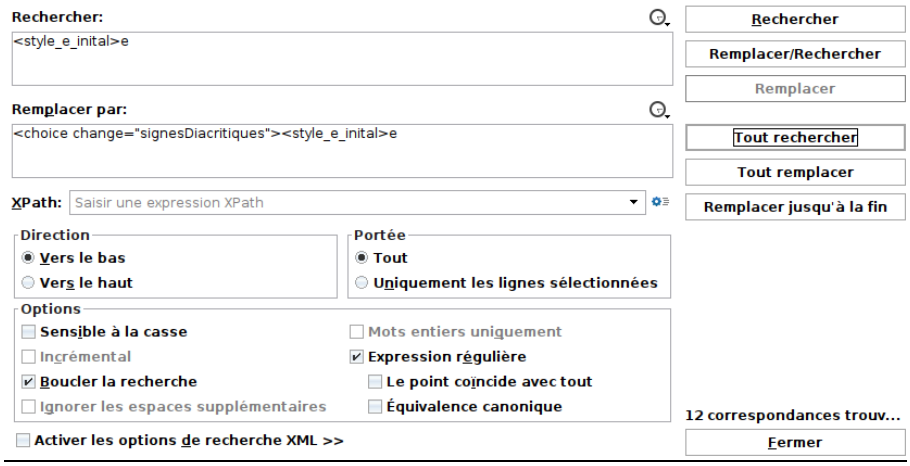
S long

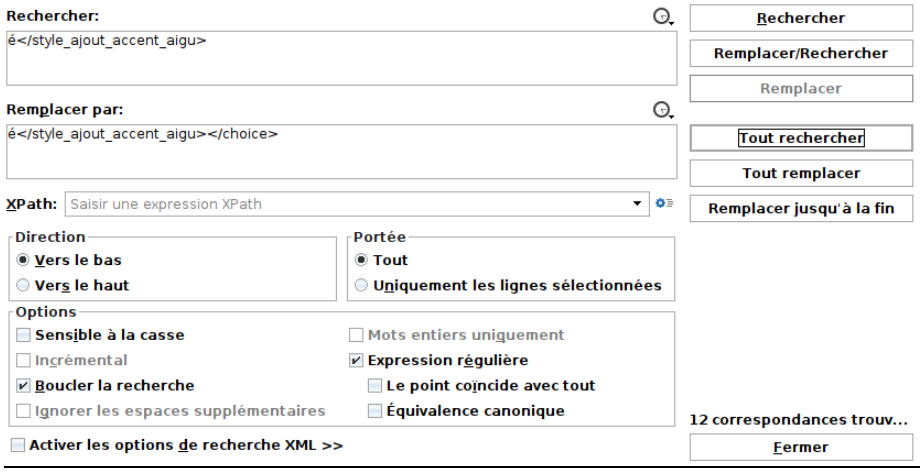
```
<xsl:template match="style_s_long">
    <xsl:element name="orig">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
<xsl:template match="style_remplacement_s_long">
    <xsl:element name="reg">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Pour les « s » longs, tout comme avec les lettres ramistes, nous avons effectué un Rechercher/Remplacer pour placer un <choice change="sLong"> en avant de la balise ouvrante <style_s_long>. Le <choice> se ferme après la balise fermante </style_replacement_s_long> regroupant ainsi dans le même <choice> le « s » long et son remplacement. Dans le fichier xsl le style_s_long est remplacé par <orig> car il est présent à l'origine dans le texte tandis que le style_remplacement_s_long se transforme en <reg> car ce changement découle de la normalisation.

Style diacritique

Remplacement des e par é

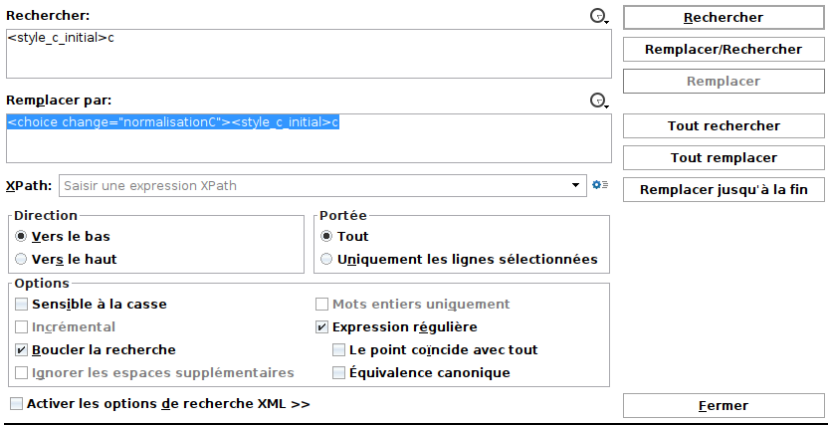


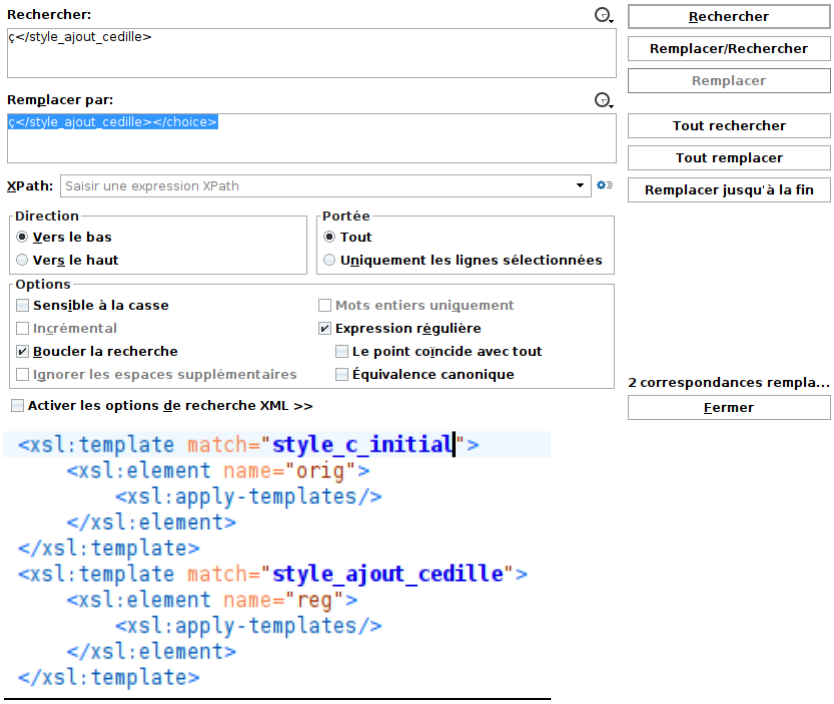



```
<xsl:template match="style_e_inital">
    <xsl:element name="orig">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
<xsl:template match="style_ajout_accent_aigu">
    <xsl:element name="reg">
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Par un Rechercher/Remplacer, le <style_e_initial> se voit précéder d'un <choice change=" signesDiacritiques "> qui permet de regrouper les balises <style_e_initial> et <style_ajout_accent_aigu> dans un <choice> spécifié par un @change qui précise qu'il s'agit d'un signe diacritique. Dans le fichier xsl, le style_e_initial devient un <orig> puisqu'il est présent dans le fac-similé et le style_ajout_accent_aigu devient un <reg> comme il intervient dans la version normalisée.

Ajout ç

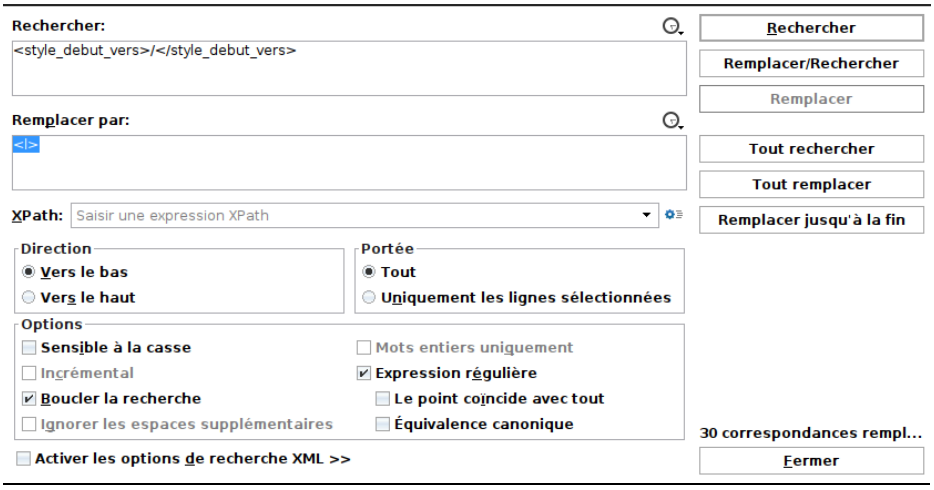


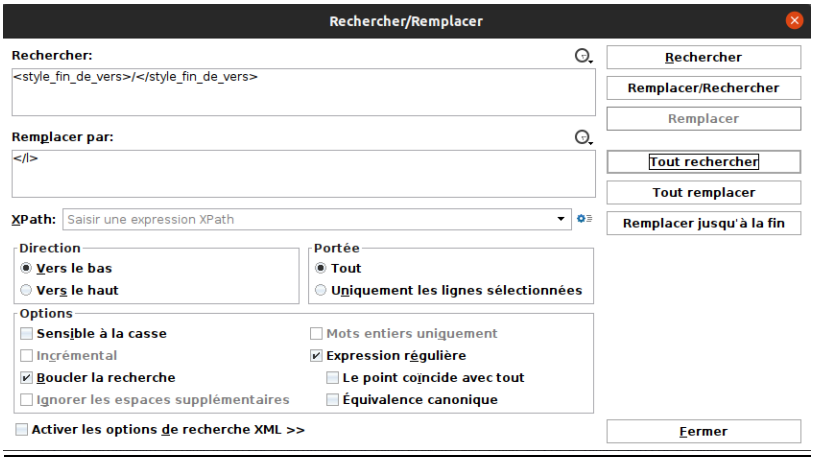


Dans le fichier xsl, le style_c_initial, balisant les « c » de la version fac-similaire, devient donc un <orig> tandis que le style_ajout_cedille se transforme en un <reg> puisqu'il est présent dans la version normalisée. Par un Rechercher_Remplacer, tout comme pour les « S » longs ou les lettres ramistes, nous avons intégrer un <choice> dont le @change indique qu'il s'agit d'un acte de normalisation. Ce <choice> s'ouvre avant le <style_c_initial> et se ferme après le <style_ajout_cedille>.

Éléments structurels

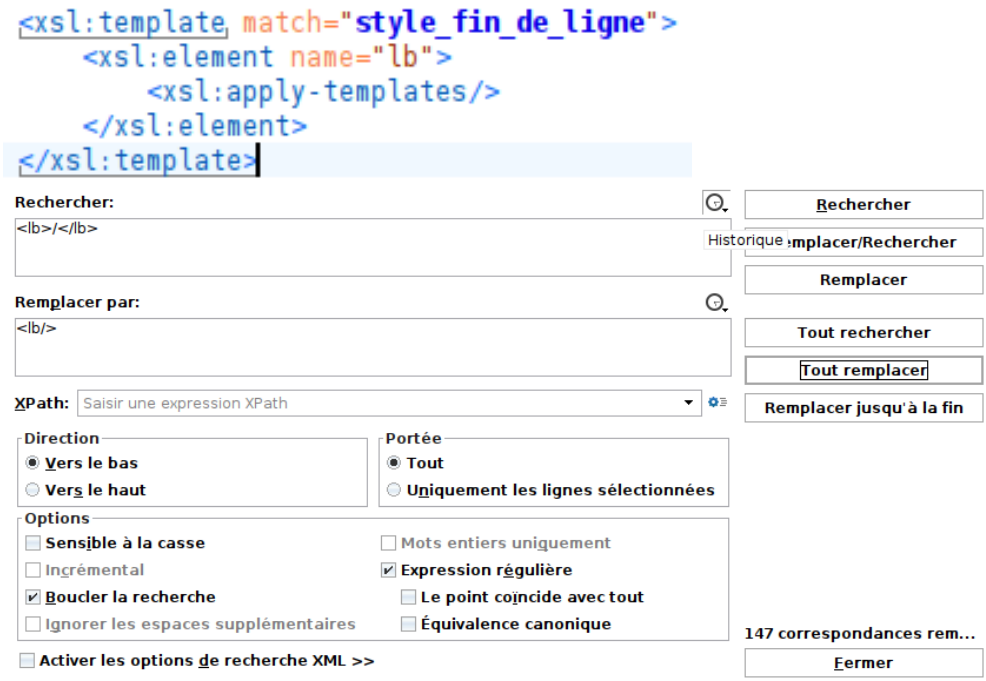
Style vers





Pour les éléments structurels, c'est-à-dire les styles de début et fin de vers, nous avons procédé à un rechercher-remplacer car le style_fin_de_vers est ici remplacé par une balise fermante </l> tandis que le style_debut_de_vers devient une balise ouvrante <l>. L'utilisation de template nous est apparue complexe puisqu'ici ces deux styles distincts sont réunis pour former un couple de balises, ouvrante et fermante.

Style ligne



Le style_fin_de_ligne, contrairement au style_fin_de_vers, ne nécessite pas de style_début_de_ligne. Par conséquent, à l'aide d'un rechercher-remplacer, le style_fin_de_ligne devient une balise auto-fermante <lb/>.

Corrections éditoriales

Ajout

```
<xsl:template match="style_ajout_editeur">
    <xsl:element name="choice">
        <xsl:element name="corr">
            <xsl:attribute name="resp">
            <xsl:attribute name="nom">editor</xsl:attribute>
        </xsl:attribute>
                <xsl:apply-templates/>
        </xsl:element>
    </xsl:element>
</xsl:template>
```

Le style_ajout_editeur devient par l'expression <xsl:template match=" " > un élément <choice> dans lequel vient s'imbriquer un élément <corr> possédant un attribut @resp (qui indique la personne à l'origine de la modification) que nous nommons Editor.

Ajout apostrophe

```
<xsl:template match="ajout_apostrophe">
    <xsl:element name="choice">
        <xsl:attribute name="change">
            <xsl:attribute name="nom">desagglutination</xsl:attribute>
        </xsl:attribute>
        <xsl:element name="reg">
            <xsl:apply-templates/>
        </xsl:element>
    </xsl:element>
</xsl:template>
```

L'ajout_apostrophe se transforme en <choice>, lequel contient un attribut @change puisque nous révisons le texte en y ajoutant une apostrophe, c'est pour cette raison que nous nommons l'attribut desagglutination car nous séparons deux lettres par une apostrophe. De plus, nous insèrons un <reg> puisque nous avons ajouté des apostrophes à la version normalisée, apostrophes absentes du fac-similé.

Ajout ponctuation

```
<xsl:template match="style_ajout_ponctuation">
    <xsl:element name="pc">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">PONfrt</xsl:attribute>
        </xsl:attribute>
        <xsl:element name="choice">
            <xsl:element name="reg">
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:element>
    </xsl:element>
</xsl:template>
```

Le style_ajout_ponctuation devient un élément <pc>, élément TEI symbolisant la ponctuation. De surcroît, il est spécifié par un attribut type que nous nommons PONfrt. À cela, nous ajoutons un élément <choice> dans lequel se trouve un <reg> puisqu'il s'agit d'un enrichissement du contenu du fac-similé.

La césure

```
<xsl:template match="style_cesure">
    <xsl:element name="pc">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">marqueCesure</xsl:attribute>
        </xsl:attribute>
        <xsl:element name="choice">
            <xsl:element name="orig">
                <xsl:apply-templates/>
            </xsl:element>
        </xsl:element>
    </xsl:element>
</xsl:template>
```

Le style_cesure devient également un <pc> avec un attribut @type du nom de marqueCesure. Conjointement, nous exprimons un élément <choice> contenant un élément <orig> puisqu'ici il ne s'agit pas d'ajout ou de modification mais d'une retranscription fidèle du fac-similé.

La lettrine

```
<xsl:template match="style_lettrine">
    <xsl:element name="seg">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">initial</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Le style_lettrine devient un élément <seg> utilisé pour mettre en valeur une section particulière du texte, ici la lettrine qui est détachée et agrandie du bloc textuel. Cet élément est enrichie d'un attribut @type du nom d'initial.

Traitement de l'évolution linguistique

L'imparfait, le pluriel et les traces de cas

```
<xsl:template match="style_imparfait_moyen_fr">
    <xsl:element name="m">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">desinence</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="style_trace_cas">
    <xsl:element name="m">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">diacritique</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>

<xsl:template match="style_lx">
    <xsl:element name="m">
        <xsl:attribute name="type">
            <xsl:attribute name="nom">pluriel</xsl:attribute>
        </xsl:attribute>
        <xsl:apply-templates/>
    </xsl:element>
</xsl:template>
```

Nous avons choisi d'orienter notre travail sur l'évolution linguistique du français en mettant en lumière la conjugaison de l'imparfait du moyen français, les traces de cas et le pluriel en -lx. Ainsi, nos styles (style_imparfait_moyen_fr / style_trace_cas / style_lx) deviennent des éléments <n> se différenciant par leur attribut @type. Le <n>, anciennement

style_imparfait_moyen_fr, possède l'attribut @type nommé desinence. Le <n> remplaçant le style_trace_cas est associé à l'attribut @type nommé diacritique tandis que le <n> remplaçant le style_lx est lié à l'attribut @type nommé pluriel.