Vers un analyseur de contexte multilingue universel

Malek Boualem
Centre National d'Etudes des Télécommunications - France Telecom
Anciennement : New Mexico State University, USA
Email: malek.boualem@cnet.francetelecom.fr

Résumé: Le but de cet article n'est pas de définir le mécanisme d'analyse de contexte pour la saisie et l'édition des caractères. La nécessité de règles d'analyse contextuelle est maintenant bien connue pour la plupart des langues naturelles. Notre objectif est de présenter un outil fonctionnel avec l'ambition de le proposer comme un modèle standard et universel pour l'analyse de contexte pour l'ensemble des langues. Cet outil se veut aussi d'être modulaire et portable sur les différents environnements logiciels et matériels. En effet, depuis l'émergence du domaine du traitement automatique des langues et du multilinguisme en informatique, un certain nombre d'outils et de logiciels d'édition de textes multilingues ont été proposés par des développeurs ou des entreprises. Ces outils intègrent dans la plupart des cas un analyseur de contexte pour le contrôle des règles d'écriture de chacune des langues (caractères à glyphes, accentuation, ligatures, ...). Cependant chaque analyseur de contexte est élaboré de façon différente soit pour l'adapter à un jeu de caractères précis, soit à un environnement local particulier. Le modèle que nous présentons ici est écrit en Java, il est compatible avec la norme Unicode (et ISO 10646) ainsi qu'avec les normes de translittération des caractères non latins. En outre la modularité et la simplicité du formalisme des règles d'écriture rend celles-ci modifiables par les utilisateurs et également adaptables à d'autres normes de codage des caractères. Enfin, un exemple est présenté à la fin de l'article pour illustrer ce modèle.

1. Introduction

L'analyse de contexte pour le contrôle des règles d'écriture des caractères est nécessaire pour la plupart des scripts. Rappelons que le script est une notion plus générale que la langue. En fait, un script est un jeu de caractères définissant un ensemble de langues (script latin pour l'anglais, le français, l'allemand, script arabe pour l'arabe, le persan, etc.). Dans l'attente de la stabilité du standard Unicode, la plupart des logiciels d'édition de textes utilisent différentes normes de codage des caractères *(Ascii, série ISO-8859-*, etc.)* ou même des codages spécifiques à certains environnements matériels *(MS-Windows character set for Western Europe MS CP1252, Dec Multinational Character Set, International IBM PC character set IBM CP850, Macintosh Extended Roman character set, Hewlett-Packard ROMAN8, etc.)*. Par conséquent, les programmes d'analyse de contexte sont écrits de façon dépendante de ces différents codages des caractères. Il arrive même que, pour simplifier les programmes de traitement, les polices de caractères soient modifiées et les caractères soient déplacés de leur emplacement standard. Il devient ainsi difficile de réutiliser ces programmes avec d'autres éditeurs de textes utilisant des polices de caractères structurées différemment. Un autre problème tout aussi important concerne le formalisme à utiliser par l'analyseur de contexte (classes de caractères, règles d'écriture, etc.). Ce formalisme devrait être lisible et facilement modifiable par l'utilisateur. En effet, la classification des caractères d'une langue ou la définition des règles d'écriture ne peut pas toujours être définitive. L'utilisateur, qui n'est pas nécessairement programmeur, mais qui connaît les caractéristiques scripturelles de sa langue, devrait pouvoir modifier ou rajouter des règles d'écriture. Malheureusement, le compromis de l'efficacité des programmes de traitement pousse souvent les développeurs à élaborer des formalismes que seuls eux sont capables de lire et de modifier.

Le travail présenté ici est issu de travaux antérieurs que nous avons menés et améliorés dans un soucis constant de standardisation et de réutilisabilité. Ainsi, les analyseurs de contexte associés soit à l'éditeur de textes multilingues X1 réalisé à Y1 [Boualem.M (1987)], soit à l'éditeur X2 réalisé au Y2 [Boualem.M (1997)], ne peuvent être réutilisés, en l'état, avec d'autres normes de codage des caractères ou sur d'autres plates-formes logicielles ou matérielles.

2. Saisie et codage des caractères

Les mécanismes de saisie et les normes de codage des caractères sont des éléments pertinents pour la mise en place d'un système d'analyse de contexte pour l'édition des textes monolingues ou multilingues. La standardisation d'une méthode pour l'analyse de contexte dépend en partie des méthodes de saisie ou des normes de codage des caractères.

2.1. Saisie des caractères

La plupart des claviers ne représentent que les caractères de l'ASCII (ou ISO 646), mais certains claviers localisés comportent des touches pour des caractères accentués ou des caractères spéciaux. Les solutions proposées par les constructeurs d'ordinateurs sont souvent hétérogènes. Pour saisir des caractères accentués sur un PC ou un Macintosh par exemple, l'accent est saisi avant le caractère. Mais sur une station SUN, une touche de composition est nécessaire avant d'entrer le caractère et l'accent. Les claviers français par exemple, comportent habituellement les touches correspondant aux caractères "à ç é è ù", mais les caractères comportant un accent circonflexe ou un tréma (ê ï ...) sont saisis au moyen de deux frappes successives. Pour ce qui est des langues non basés sur l'alphabet latin, la saisie des caractères sur un clavier standard repose sur leur translittération en caractères latins. Des normes de translittération existent, mais elles ne sont pas toujours respectées par les constructeurs. Pour la saisie de l'arabe, par exemple, où l'alphabet contient plus de 28 lettres dont chacune est accompagnée de plusieurs glyphes contextuels, il est difficile de définir un clavier contenant tous les glyphes possibles. Il est donc nécessaire, pour la plupart des langues, de définir des règles d'écriture contextuelle. Le mécanisme d'analyse de contexte permet une saisie plus intuitive en minimisant le nombre de frappes au clavier.

2.2. Codage des caractères

Les constructeurs d'ordinateurs et les concepteurs de logiciels utilisent de nombreux codes de caractères spécifiques et non compatibles. Par ailleurs, des versions successives de normes de codage des caractères ont été élaborées au niveau international et utilisées sur certaines plates-formes. En particulier, la série de normes ISO 8859-* propose un codage des caractères sur 8 bits pour les scripts latin, cyrillique, arabe, grec et hébreu. Plus récemment (depuis 1993), la norme ISO/IEC 10646 [ISO/IEC (1993)], couplée avec Unicode [Unicode (1996)], a proposé un codage sur 16 bits ayant l'ambition de couvrir l'ensemble des caractères de tous les scripts. Toutefois, les environnements matériels et logiciels ne sont pas tous encore prêts à l'implémentation de jeux de caractères sur plusieurs octets.

3. Aspects de l'analyse de contexte

Dans un éditeur de texte, l'analyseur de contexte est un programme qui analyse les codes des caractères saisis au clavier ou provenant de la structure interne du texte et génère des caractères individuels, des caractères combinés ou des glyphes destinés à être affichés à l'écran. Cette analyse est normalement effectuée sur la base de règles prédéfinies pour chaque langue. Bien que les algorithmes utilisés dans la plupart des analyseurs de contexte existants soient globalement les mêmes, les programmes d'analyse sont souvent élaborés de manière dépendante des polices de caractères utilisées ou des plate-formes de développement. Ainsi, certaines approches utilisent des polices de caractères adaptées afin de faciliter les traitements par les programmes, les caractères peuvent être déplacés de leurs positions standards dans les polices de caractères ou dupliqués (voir figure 1). Par conséquent, ces programmes ainsi personnalisés, ne peuvent fonctionner avec d'autres polices de caractères.

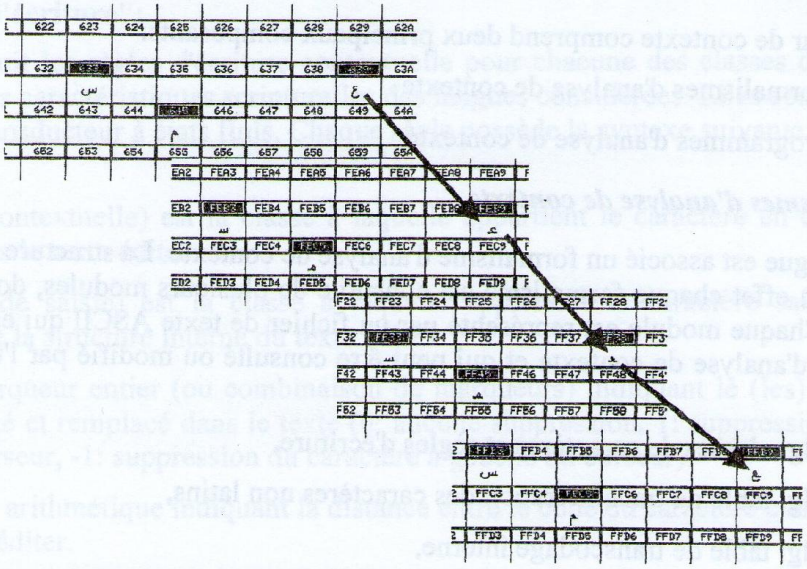


Figure 1. *Les glyphes de cette police de caractères personnalisée sont atteints par des translations régulières (exemple élaboré à l'aide de l'éditeur de polices de caractères XMBDFED développé par M.Leisher au CRL, New Mexico State University)*

Généralement le programme d'analyse de contexte est associé à un ensemble de classes de caractères et de règles d'écriture. Celles-ci devraient être exprimées dans un formalisme facile à lire et à comprendre par l'utilisateur. En effet, en cas de nécessité, celui-ci devrait pouvoir modifier les classes de caractères ou les règles d'écriture ou aussi en rajouter d'autres. Par exemple, cette suite d'instructions PERL issues du filtre *Arabjoin* pour l'analyse de contexte de l'arabe [Czyborra R. (1998)], bien que optimales et efficaces, elles sont difficilement lisibles et compréhensibles par un utilisateur non informaticien:

```
$uchar[$i] = $a && $final{$c} && $medial{$b}
|| $final{$c} && $initial{$b}
|| $a && $final{$b}
|| $isolated{$b}
|| $b;
```

Enfin, dans certaines approches [Leisher M., (1998)], les règles d'analyse de contexte peuvent même être des instructions du programme de l'éditeur de texte. Dans ce cas, elles sont, non seulement, difficile à identifier et à comprendre, mais leur mise à jour nécessite la re-compilation des programmes de traitement.

4. Description du modèle d'analyse de contexte proposé

L'architecture générale de l'analyseur de contexte proposé se présente comme suit:

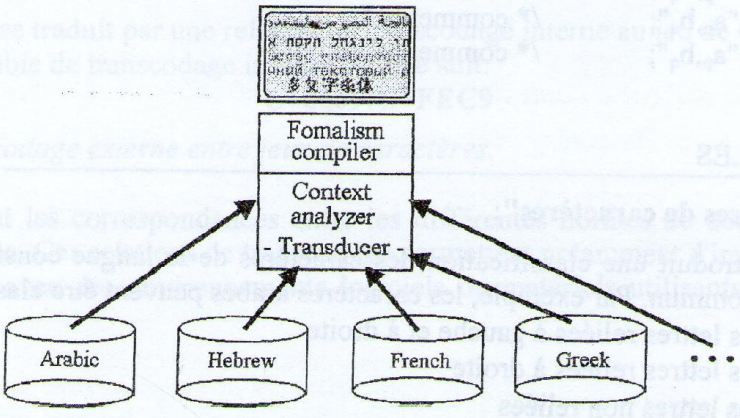


Figure 2. *Architecture générale de l'analyseur de contexte*

L'analyseur de contexte comprend deux principaux composants:

- Formalismes d'analyse de contexte
- Programmes d'analyse de contexte

4.1. Formalismes d'analyse de contexte

A chaque langue est associé un formalisme d'analyse de contexte. La structure des formalismes est très modulaire, en effet chaque formalisme est constitué de plusieurs modules, dont certains peuvent être optionnels. Chaque module est représenté par un fichier de texte ASCII qui est donc indépendant des programmes d'analyse de contexte et qui peut être consulté ou modifié par l'utilisateur. Ces modules sont:

- rules.lg: classes de caractères et règles d'écriture,
- trans.lg: table de translittération des caractères non latins,
- local.lg: table de transcodage interne,
- cconv.lg: table de transcodage externe.

L'extension "**lg**" désigne le code standard de la langue correspondant à la norme de codage des noms des langues à 2 lettres alphabétiques **ISO 639-1988** ("**en**" pour anglais, "**fr**" pour français, etc.).

4.1.1. Classes de caractères et règles d'écriture

Ce module est le plus significatif dans le formalisme associé à la langue car il introduit les règles d'écriture propre à la langue et qui sont basées sur les caractéristiques scripturelles de celle-ci.

La structure de ce module se présente comme suit:

```
# COMMENT SECTION
#

BEGIN CLASSES
  C1={ci, ci+1, ci+2, ...}          /* comments */
  C2={cj, cj+1, cj+2, ...}          /* comments */
  C3={ck, ck+1, ck+2, ...}          /* comments */
  ...
END CLASSES

# Comments
#

BEGIN RULES
  Cn,Cm:n,"ap,bq";                  /* comments */
  Cn,Cm:n,"ap,bq";                  /* comments */
  Cn,Cm:n,"ap,bq";                  /* comments */
  ...
END RULES
```

a. Section "classes de caractères":

Cette section introduit une classification des caractères de la langue considérée en fonction de leur comportement commun. Par exemple, les caractères arabes peuvent être classés comme suit:

- classe des lettres reliées à gauche et à droite
- classe des lettres reliées à droite
- classe des lettres non reliées
- classe des chiffres
- ...

b. Section "règles d'écriture":

Cette section introduit les règles d'écriture contextuelle pour chacune des classes de caractères, ces règles sont issues des caractéristiques scripturelles des langues considérées. Elles sont exprimées dans un formalisme de transducteur à états finis. Chaque règle possède la syntaxe suivante: C_n , C_m : n, "a_p, b_q" ; où:

- C_n (classe contextuelle) est la classe à laquelle appartient le caractère en contexte (près du curseur) dans le texte édité.
- C_m (classe de saisie) est la classe à laquelle appartient le caractère saisi au clavier ou provenant de la structure interne du texte édité.
- n est un marqueur entier (ou combinaison de marqueurs) indiquant le (les) caractère devant être supprimé et remplacé dans le texte (0: aucune suppression, 1: suppression du caractère à droite du curseur, -1: suppression du caractère à gauche du curseur).
- a_p est un pas arithmétique indiquant la distance entre le code du caractère contextuel et le code du glyphe à éditer.
- b_q est un pas arithmétique indiquant la distance entre le code du caractère saisi et le code du glyphe à éditer.

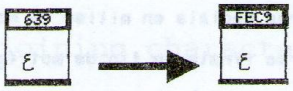
Bien que ce formalisme des règles d'écriture semble difficile à maîtriser au premier abord, mais il reste parfaitement lisible et compréhensible par les utilisateurs connaissant les langues en question, mais au prix, toutefois, d'un certain effort de concentration.

4.1.2. Tables de translitération des caractères non latins

Ces tables sont instanciées lorsqu'un clavier Ascii standard est utilisé pour la saisie de caractères non latins. Elles introduisent la correspondance entre les caractères de la langue considérée et les caractères du jeu Ascii (ISO 646). Des normes de translittération existent pour les langues non basés sur l'alphabet latin (**ISO 233-1984/1993** pour l'arabe, **ISO 259-1984** pour l'hébreu, **ISO/R 843-1968** pour le grec, etc.). Ainsi, on saisit "**a**" pour "α", "**b**" pour "β", "**S**" pour "ﺲ", etc.

4.1.3. Tables de transcodage interne à un jeu de caractères

Ces tables sont instanciées dans un cas assez particulier. Pour mieux comprendre cet aspect, prenons l'exemple de la plage de caractères arabes dans le standard Unicode. Les caractères arabes de base sont représentés dans la plage U+0600 à U+06FF. Afin de prévoir l'analyse de contexte de l'arabe, les formes contextuelles des caractères arabes sont représentées dans la plage de compatibilité (Compatibility Area) U+FE70 à U+FEFF (Arabic Presentation Forms-B). Si on prend le caractère "ARABIC LETTER AIN" codé U+0639 dans la plage des caractères arabes de base, il lui correspond le code U+FEC9 dans la plage de compatibilité pour la même forme graphique.



Cette correspondance se traduit par une relation de transcodage interne au jeu de caractères arabes qui est exprimée dans la table de transcodage interne comme suit:

U+0639:U+FEC9

4.1.4. Tables de transcodage externe entre jeux de caractères.

Ces tables introduisent les correspondances entre les différentes normes de codages des caractères pour une langue donnée. Ces relations de transcodage permettent notamment d'importer, d'exporter ou d'échanger des textes avec des environnements logiciels ou matériels utilisants d'autres normes de codage des caractères.

4.2. Programmes d'analyse de contexte

L'analyseur de contexte contient deux types de programmes:

- Le compilateur de formalismes associés aux langues
- Le moteur d'analyse de contexte

Le compilateur de formalisme a pour rôle de vérifier la structure du formalisme de la langue choisie par l'utilisateur avant son chargement et son utilisation. Le moteur d'analyse a pour rôle de scruter les caractères saisis au clavier ou provenant de la structure interne du texte et d'appliquer les règles d'analyse de contexte contenues dans le formalisme de la langue considérée.

L'écriture de ces programmes en langage Java et la modularité des formalismes des langues permet de porter ou d'adapter l'analyseur de contexte sur différents environnements logiciels et matériels.

5. Applications de l'analyse de contexte

L'analyse de contexte est nécessaire pour un grand nombre de langues notamment celles dont les scripts présentent des combinaisons de caractères, des ligatures ou des glyphes.

5.1. Composition dynamique des caractères accentués

L'analyse de contexte rend la saisie des caractères accentués plus simple et plus intuitive que celle proposée par certains environnements matériels. Par exemple, le caractère accentué français "**ê**" est saisi par la séquence "**e+^**" plutôt que "**compose+e+^**" ou "**^+e**". Bien entendu, un mécanisme d'échappement doit être prévu pour éviter l'accentuation automatique:

- **e + ' => é ;**
- **e + ESC + ' => e' ;** (français)
- **s + s => ß ;**
- **s + ESC + s => ss ;** (allemand)
- etc.

5.2. Génération des glyphes des caractères

Dans certaines langues, les caractères peuvent prendre des formes différentes selon le contexte dans lequel ils apparaissent (position dans le mot, etc.). Par exemple:

- Grec: la lettre σ s'écrit σ en début et en milieu de mot et elle s'écrit ς en fin de mot.

- Arabe:

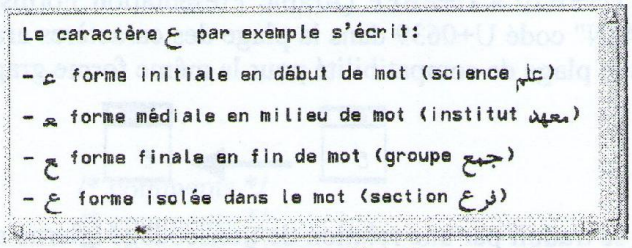


Figure 3. *Variantes contextuelles des caractères arabes (exemple élaboré à l'aide de l'éditeur de texte multilingue MtScript [MtScript, 1996]*

La plupart des normes de codage des caractères (Unicode, ISO-8859-*, etc.) ne prennent pas en compte toutes les variantes graphiques des caractères, celles-ci ayant le même code. C'est le rôle de l'analyseur de contexte de définir les variantes des caractères en sortie (écran, imprimante, etc.) à partir du même code de caractère en entrée (clavier, fichier, etc.).

6. Exemple: analyse de contexte de l'arabe basée sur Unicode

Pour l'analyse de contexte de l'arabe, nous utilisons la classification des caractères et les règles d'écriture définies dans le Standard Unicode (**Joining classes** et **Joining rules**). Les règles d'écriture sont appliquées à l'ensemble des caractères arabes de base définis dans la plage U+0600 à U+06FF (plus exactement U+060C to U+06F9) de l'Unicode 2.1.

L'analyse utilise les variantes contextuelles arabes définies dans la plage de compatibilité (Arabic Presentation Forms-A: U+FB50 à U+FDFF et Arabic Presentation Forms-B: U+FE70 à U+FEFF).

Nous présentons dans ce qui suit un extrait du formalisme des classes de caractères et des règles d'écriture de l'arabe:

```
# Arabic writing rules
#
# This set of rules is compatible with Unicode.
# The rendering rules are applied on the range of the basic
# Arabic characters of Unicode
# (U+0600D to U+06FF / U+060C to U+06F9)
# and use the contextual glyphs of the Arabic Presentation
# Forms-B (U+FE70 to U+FEFF).
#

BEGIN CLASSES

# Control characters
# ------------------
C1={U+0000-U+0007,U+0009-U+001A,U+001C-U+001F}    /* control */
C110={U+0008}                                     /* backspace */
C111={U+007F}                                     /* delete */
C5={U+0020}      /* space */

# Symbols
# -------
C2={U+0021-U+002F,U+003A-U+003E,U+0040,U+005B-U+0060,U+007B-U+007E}
C200={U+061B,U+061F,U+066A-U+066D}     /* Arabic symbols */

# Digits
# ------
C3={U+0660-U+0669}

# Non-joining characters
# ----------------------
C7={U+FE80}      /* hamza */

# Dual joining characters
# -----------------------

# Independent forms of dual joining characters

C60={U+FE89,U+FE8F,U+FE95,U+FE99,U+FE9D,U+FEA1,U+FEA5,U+FEB1,U+FEB5,
     U+FEB9,U+FEBD,U+FEC1,U+FEC5,U+FEC9,U+FECD,U+FED1,U+FED5,U+FED9,
     U+FEDD,U+FEE1,U+FEE5,U+FEE9,U+FEF1}

# Initial forms of dual joining characters

C61={U+FE8B,U+FE91,U+FE97,U+FE9B,U+FE9F,U+FEA3,U+FEA7,U+FEB3,U+FEB7,
     U+FEBB,U+FEBF,U+FEC3,U+FEC7,U+FECB,U+FECF,U+FED3,U+FED7,U+FEDB,
     U+FEDF,U+FEE3,U+FEE7,U+FEEB,U+FEF3}

# Medial forms of dual joining characters

C62={U+FE8C,U+FE92,U+FE98,U+FE9C,U+FEA0,U+FEA4,U+FEA8,U+FEB4,U+FEB8,
     U+FEBC,U+FEC0,U+FEC4,U+FEC8,U+FECC,U+FED0,U+FED4,U+FED8,U+FEDC,
```

```
    U+FEE0,U+FEE4,U+FEE8,U+FEEC,U+FEF4}
# Final forms of dual joining characters

C63={U+FE8A,U+FE90,U+FE96,U+FE9A,U+FE9E,U+FEA2,U+FEA6,U+FEB2,U+FEB6,
     U+FEBA,U+FEBE,U+FEC2,U+FEC6,U+FECA,U+FECE,U+FED2,U+FED6,U+FEDC,
     U+FEDE,U+FEE2,U+FEE6,U+FEEA,U+FEF2}

# Right joining characters
# ------------------------
# Independent-initial forms of right joining characters

C70={U+FE81,U+FE83,U+FE85,U+FE87,U+FE8D,U+FE93,U+FEA9,U+FEAB,U+FEAD,
     U+FEAF,U+FEED,U+FEEF}

# Medial-final forms of right joining characters

C71={U+FE82,U+FE84,U+FE86,U+FE88,U+FE8E,U+FE94,U+FEAA,U+FEAC,U+FEAE,
     U+FEB0,U+FEEE,U+FEF0}

END CLASSES

BEGIN RULES

C*,C*:0,"b";     /* non particular characters: not changed */
C*,C110:-1,"";          /* backspace */
C*,C111:1,"";           /* delete */

C3,C3:+1,"a,b"; /* Numbers */
C*,C5:0,"b";            /* space */
C*,C7:0,"b";            /* hamza */
C*,C60:0,"b";           /* dual joining characters: independent form
*/
C*,C70:0,"b";    /* right joining characters: independent-initial
form */

# Rules for dual joining characters
# ---------------------------------

# state: dual joining character in independent form
C60,C60:+1,"b+1,a+2";       /* dual joining: final */
C60,C70:+1,"b+1,a+2";       /* right joining: final */

# state: dual joining character in initial form
C61,C*:+1,"b,a-2";     /* dual joining independent + any */
C61,C110:-1,"";         /* backspace */
C61,C111:1,"";              /* delete */
C61,C60:0,"b+3";        /* dual joining: medial */
C61,C70:0,"b+1";        /* right joining: medial */

# state: dual joining character in medial form
C62,C*:+1,"b,a-2";     /* dual joining final + any */
C62,C110:-1,"";         /* backspace */
C62,C111:1,"";              /* delete */
C62,C60:0,"b+3";        /* dual joining: medial */
C62,C70:0,"b+1";        /* right joining: medial */
```

```
# state: dual joining character in final form
C63,C60:+1,"b+1,a+2"; /* dual joining: final */
C63,C70:+1,"b+1,a+2"; /* non regular: final */

END RULES
#
```

7. Conclusion

Le formalisme que nous avons décrit ici a déjà été expérimenté sous des formes moins générales et des utilisateurs connaissant certaines langues ont pu mettre à jour ou définir des règles d'écriture pour l'analyse de contexte de leurs langues (distribution de MtScript). Bien entendu, à l'image de l'exemple présenté ici, ce formalise nécessite encore des améliorations. Par exemple, la restriction à une seule classe contextuelle de caractères dans la partie gauche des règles d'écriture (C_n , C_m :) limite les possibilités d'analyse car dans certains cas (formes contextuelles des caractères), les deux caractères (contextuels) voisins du curseur sont modifiés simultanément lors de la saisie d'un nouveau caractère. Il n'en demeure pas moins que ce formalisme est très modulaire, compréhensible par les utilisateurs et adaptable à de nouvelles langues. En outre son développement en langage Java et sa compatibilité avec les normes de codage de caractères en usage en font un outil portable et réutilisable dans des environnements logiciels et matériels différents.

Références

1. Boualem A.M., Harié S. (1997), "MtScript: a multilingual text editor", *Computers and the Humanities*, Vol. 31, No. 2, Kluwer Academic Publishers, pp.135-151.

2. Boualem A.M. (1987), "Un analyseur de contexte multilingue", DEA d'informatique, Université de Nice Sophia Antipolis, INRIA.

3. Czyborra R. (1998), The Arabjoin script for rendering Arabic text. *http://czyborra.com/unicode/arabjoin*

4. ISO/IEC 10646 (1993), ISO/IEC 10646 standard character set, *http://www.iso.ch/*

5. Leisher M. (1998), MUTT, the Multilingual Unicode Text Toolkit (éditeur de textes multilingues), Computing Research Laboratory, New Mexico State University *(ftp://crl.nmsu.edu/CLR/multiling/unicode/).*

6. MtScript (1996), éditeur de textes multilingues développé par M.Boualem et S.Harié au laboratoire Parole et Langage du CNRS, Projet Multext, Université de Provence: *http://www.lpl.univ-aix.fr/projects/multext/MtScript/*

7. Unicode (1996), The Unicode Standard, Version 2.0, The Unicode Consortium, Addison-Wesley Developers Press.