Affichage et diffusion sur Internet d'explications en langue arabe de preuves mathématiques

Hanane Naciri – Laurence Rideau
INRIA Sophia Antipolis, BP. 93, 06902 Sophia Antipolis Cedex,France.
{Hanane.Naciri,Laurence.Rideau}@sophia.inria.fr

Résumé: Nous présentons ici une extension de FIGUE, librairie d'affichage bidimensionnel, que nous avons développée pour afficher des textes de preuves mathématiques en langue arabe et les diffuser sur Internet. Nous décrivons les problèmes que pose l'affichage de texte arabe (de droite à gauche) dans un contexte prévu pour l'affichage de texte indo-européen (de gauche à droite), puis l'affichage bidirectionnel mélangeant les deux modes (par exemple du texte arabe incluant du texte indo-européen ou des formules mathématiques). Nous montrons aussi comment cette expérience peut nous servir à diffuser ces textes mathématiques sur Internet, grâce à une extension bidirectionnelle de MathML (dialecte de XML dédié aux mathématiques).
Mots clés: formules mathématiques, texte bidirectionnel, outils de formatage, MathML, explications de preuves en langue arabe, interaction homme machine.
Abstract: We present here our work on FIGUE, a bi-dimensional display library, used to display mathematical proofs using arabic natural language and to broadcast these proofs on the Internet. The problems encountered to display arabic text (right-to-left) in the context of indo-european text (left-to-right) are described. Then we show how we perform bi-directional display mixing left-to-right and right-to-left texts (for example arabic text including indo-european text or mathematical formulas). Moreover, we present how this work can help to broadcast these mathematical texts on the Internet, using a bi-directional MathML extension.

1 Introduction

Jusqu'à présent la diffusion des documents scientifiques se faisait en général en anglais, mais avec l'internationalisation, ces documents circulent dans différentes langues et un grand besoin d'outils de formatage multi-langues est apparu. Ces outils permettent d'afficher des documents comportant différents scripts et langues et doivent gérer automatiquement les changements de direction en cas de langues écrites de droite à gauche comme l'arabe et l'hébreu ou de langues écrites de haut en bas comme le japonais et le coréen. De tels outils de formatage aident au développement d'éditeurs de documents et d'environnements graphiques multi-langues. Nous distinguons deux familles d'outils de formatage: les outils en mode non-interactif (*batch-oriented processors*) comme par exemple le système TEX [7] et les outils WYSIWYG (*What You See Is What You Get*) comme par exemple AMAYA [14], éditeur WYSIWYG de pages Web permettant entre autre une édition structurée de formules mathématiques. Parmi les outils non-interactifs de formatage multi-langues citons le système ArabeTEX [8] permettant d'écrire du texte arabe en TEX et son extension arabemath [9, 10] pour l'écriture de formules mathématiques arabes. Dans le cas WYSIWYG, plusieurs problèmes sont posés pour le traitement du texte multi-langues et sont décrits dans [4, 3]. Le système ditroff/ffortid [13] est un exemple de système WYSIWYG pour le formatage du texte arabe.

Dans cet article, nous décrivons comment nous avons étendu FIGUE [11], notre moteur d'affichage interactif d'objets structurés, pour l'adapter au formatage de droite à gauche du texte en arabe, hébreu ou perse et étudier comment mélanger les directions dans le cas du texte mathématique (texte de droite à gauche, incluant des formules écrites de gauche à droite). Le problème se pose de façon particulière pour les formules mathématiques qui ont des règles d'affichage bidirectionnel qui leur sont propres et différentes de celles appliquées pour du texte simple (linéaire). Cela est dû à la nature bidimensionnelle des formules mathématiques et à l'importance des relations spatiales dans les notations mathématiques.

Nous nous sommes intéressées au lien entre FIGUE et le standard MathML des mathématiques sur Internet [12]. La version actuelle de MathML supporte uniquement l'affichage des formules mathématiques de gauche à droite. Nous collaborons avec le professeur Stephen Watt, invité expert du groupe de travail W3C autour de MathML, pour définir de nouvelles recommandations de la norme MathML pour un affichage bidirectionnel de formules mathématiques. FIGUE est utilisé actuellement pour le développement de PCOQ [2], une interface graphique conçue pour le système de preuves COQ [6]. Nous expérimentons dans PCOQ l'implémentation des explications des preuves en arabe, ce qui permet de tester cette extension de FIGUE pour un affichage bidirectionnel du texte mathématique.

2 Affichage bidirectionnel dans FIGUE

La spécification d'unicode attribue une orientation aux caractères selon leurs propriétés et définit un algorithme (complexe) pour déterminer l'orientation correcte d'un texte. FIGUE se repose sur cet algorithme bidirectionnel d'unicode [1] pour le formatage bidirectionnel du texte linéaire et il l'étend pour manipuler des formules mathématiques dans un contexte bidirectionnel : comment afficher des formules dans le script arabe ou hébreu et comment gérer le changement de direction d'affichage. Pour déterminer les règles d'affichage des formules dans un contexte bidirectionnel, nous nous sommes basés sur les règles appliquées dans l'enseignement des mathématiques dans le monde arabe.

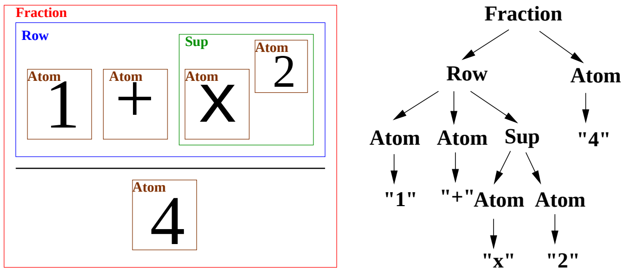
2.1 Représentation des objets structurés bidirectionnels

FIGUE se base sur la notion de boîtes [7] pour représenter tous les objets structurés du document. Il manipule un arbre de boîtes qu'il formate puis affiche. Cet arbre de boîtes peut être obtenu de différentes manières (comme par exemple à partir d'une description XML). Les objets du document sont représentés sous forme d'arbre de boîtes, où chaque nœud est une boîte (constructeur graphique) englobant toutes ses boîtes filles et les feuilles sont des atomes (unités basiques comme les chaînes de caractères). La figure 1 montre un exemple simple de représentation en boîtes d'une formule mathématique.

Dans un contexte bidirectionnel, les boîtes ont un indicateur de direction qui prend comme valeur *LR* (*left-to-right* de gauche à droite) ou *RL* (*right-to-left* de droite à gauche). La direction de chaque objet peut être donnée explicitement ou déduite automatiquement selon des règles d'héritages et selon la direction principale du document. L'objet racine prend la valeur de la direction principale du document (par exemple pour un document arabe la direction principale est *RL*).

Les constructeurs graphiques de base de formatage en FIGUE comme *Horizontal*, *Ver-*

$$\frac{1+x^2}{4}$$



(a) (b) (c)

Figure 1: (a) Formule. (b) Ensemble de boîtes. (c) Arbre de boîtes.

tical et *Paragraph* héritent de la direction de la boîte mère. Le constructeur *Atom* qui représente les feuilles de l'arbre déduit sa direction suivant les propriétés des caractères qui le compose. Les constructeurs mathématiques ont leurs propres règles pour déterminer leur direction (voir section 2.2). Chaque constructeur FIGUE a son propre algorithme de formatage prenant en compte l'indicateur de sa direction propre et spécifiant l'ordre d'affichage de ses fils. Par exemple, l'algorithme de formatage du constructeur *Horizontal* (dispose horizontalement ses boîtes filles) dans le cas bidirectionnel se fait en deux étapes : (1) déterminer l'ordre d'affichage de ses éléments qui peut être différent de l'ordre logique dans la structure d'arbre sous-jacente et qui dépend de sa direction et celle de ses fils. (2) déterminer la position spatiale de chaque boîte fille du constructeur suivant l'ordre d'affichage déjà établi et la nature du constructeur. La figure 2 résume cet algorithme.

Soit d la direction du constructeur *Horizontal* et soit d_i la direction de son fils d'ordre logique i.
pour chaque boîte fille d'ordre i **faire**
si $d_i = d$ (la direction de la boîte est la même que la direction principale)
  alors ordreAffichage[i]= i (l'ordre d'affichage de cette boîte est le même que l'ordre logique)
sinon ($d_i \neq d$)
  debut
    si $d_{i+1} \neq d_i$ (la direction de la boîte suivante $i+1$ est différente de celle de la boîte i)
    alors ordreAffichage[i]= i (l'ordre d'affichage de cette boîte est le même que l'ordre logique)
    sinon
    Soit k t.q $\forall j \in [i,k] d_j \neq d$
    k est l'indice de la dernière boîte qui a une direction différente de la direction principale
    $\forall j \in [i,k]$ ordreAffichage[j] = k-j+i-1
    pour toute les boîtes à partir de l'indice i jusqu'à l'indice k, l'ordre d'affichage est inversé.
  fin

Figure 2: Algorithme du constructeur *Horizontal* pour déterminer l'ordre d'affichage de ses éléments en fonction de leur direction

2.2 Affichage bidirectionnel des formules mathématiques

Dans les textes mathématiques arabes, nous traitons le cas de deux familles de formules mathématiques : les formules à l'égyptienne écrites de droite à gauche conformément au sens de déroulement de l'écriture arabe (voir figure 3.c) et les formules à la marocaine écrites de gauche à droite suivant le modèle européen et qui peuvent contenir du texte ou des symboles arabes (voir figure 3.b). Dans la suite de cet article, nous étudions séparément les deux cas : formules marocaines et formules égyptiennes.

(a) Définition de la valeur absolue: $|x| = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x < 0 \end{cases}$

(b) تعريف القيمة المطلقة : $|x| = \begin{cases} x & x \geq 0 \text{ إذا كان} \\ -x & x < 0 \text{ إذا كان} \end{cases}$

(c) تعريف القيمة المطلقة : د(س) = { س إذا كان س ≥ 0 ، −س إذا كان س > 0 }

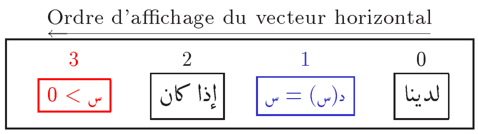
Figure 3: Un texte mathématique définissant la valeur absolue écrit suivant : (a) le modèle français (b) le modèle marocain (c) le modèle égyptien

2.2.1 Les formules mathématiques égyptiennes

Les formules mathématiques arabes à l'égyptienne sont orientées de droite à gauche et composées de symboles spécifiques. Certains symboles sont réfléchis comme dans un miroir (dans [9] on classifie les familles de symboles pour lesquels une telle transformation est possible). Par exemple les symboles $<$, $\in$ deviennent les symboles $>$,$\ni$ et vice-versa. Dans notre étude, nous étendons les constructeurs mathématiques pour les adapter à l'inversion de direction. Les changements de l'algorithme de formatage dépendent de la nature de chaque constructeur. Nous distinguons quatre familles de constructeurs :
Les symboles sont les entités de base comme les signes et les opérateurs mathématiques, les identificateurs ou des nombres (ce sont les éléments *mi*, *mo*, *mn* en MathML). Certains symboles ne changent pas (=، ∗ ، − ، + ، / ، % ، .) tandis que d'autres possèdent, au sein de la même fonte, leur symbole symétrique. Pour ces derniers symboles, nous avons le choix entre intervertir l'unicode des deux symboles (par exemple, le symbole $<$ devient le symbole $>$) ou bien intervertir les noms des deux symboles (*in* devient *ni*).
Les constructeurs horizontaux disposent horizontalement les termes de l'expression comme par exemple les éléments MathML *mrow* (regroupe horizontalement les sous-expressions) et *mtr* (représente la ligne d'une table). Ces constructeurs inversent la disposition de leurs termes suivant les mêmes principes vus dans le cas du texte écrit de droite à gauche. Par exemple l'élément MathML *mrow* suit les mêmes règles d'inversion que celles appliquées pour le constructeur *Horizontal* (l'algorithme est décrit dans la section 2.1).

```
<mrow>
  <mtext> لدينا </mtext>
  <mrow> ... /mrow>
  <mtext> إذا كان </mtext>
  <mrow> ... /mrow>
</mrow>
```

(a)



(b)

Figure 4: (a) La forme MathML de la formule arabe لدينا د(س) = س إذا كان س > 0 (b) L'ordre d'affichage des éléments du vecteur horizontal est inversé.

Les constructeurs verticaux disposent verticalement leurs éléments (par exemple les éléments *mfrac*, *mtable* dans MathML). Les algorithmes de formatage de ces constructeurs sont indifférents à la direction horizontale et ils héritent la direction du constructeur qui les englobe. Par exemple, le formatage de la fraction est indépendant de la direction.

Les constructeurs exposant, indice, racine ont une disposition spatiale importante et qui dépend fortement de la direction principale, par exemple l'indice ou l'exposant sont disposés à gauche de la base dans le cas de la direction droite à gauche (voir figure 5). La racine nième inverse la disposion de ses termes et le dessin du symbole racine est réfléchi.

Formules en mode latin	Formules en mode arabe (égyptien)
$\in, <$	$>\text{،}\ni$
$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$	$\frac{(1+\text{ں}2)(1+\text{ں})\text{ں}}{6} =^2 \text{ں} + \cdots +^2 2 +^2 1$
$M = \begin{pmatrix} 4b & 0 \\ 5 & 4 \end{pmatrix}$	$\begin{pmatrix} 0 & \text{ب}4 \\ 4 & 5 \end{pmatrix} = \text{م}$

Figure 5: Exemple de formules écrites en mode latin et en mode arabe-égyptien.

2.2.2 Les formules mathématiques marocaines

Les formules mathématiques marocaines sont orientées de gauche à droite contrairement au texte arabe. Ces formules peuvent à leur tour contenir du texte arabe. Les symboles mathématiques ne sont pas réfléchis et la disposition des termes des expressions n'est pas inversée. Le problème se pose uniquement dans le cas des constructeurs mathématiques horizontaux où il faut gérer le mélange de directions (si certains termes sont en arabe). Les autres constructeurs ne changent pas (fraction, matrice, ...). Nous donnons quelques règles pour déterminer la direction des éléments horizontaux :

- Un vecteur horizontal formate une séquence d'éléments suivant l'algorithme bidirectionnel vu dans la section 2.1. Seul les éléments textuels déterminent la direction du vecteur horizontal. Ces éléments sont pris par leur ordre logique et le premier élément textuel détermine la direction du vecteur horizontal.

- Les opérateurs et les identificateurs (les éléments *mo* et *mi* en MathML) sont traités comme des éléments neutres par rapport à la direction horizontale. Ils héritent la direction de leur père (boîte qui les englobe).

En appliquant ces règles sur l'arbre représentant la formule, nous déterminons la direction d'affichage dans chaque noeud de l'arbre. Ensuite, chaque noeud applique son propre algorithme de formatage. La figure 6 montre un exemple de formule arabe écrite suivant le modèle marocain et sa représentation MathML. L'élément *mrow* prend la direction de son premier élément textuel *mtext* et qui est la direction de droite à gauche (application de la première règle vu ci-dessus). Une conséquence directe de l'application de ces règles est l'importance de l'imbrication des éléments *mrow* en présence des éléments textuels *mtext* pour déterminer l'affichage des formules.

2.2.3 Affichage bidirectionnel dans MathML

La version actuelle de MathML admet uniquement l'affichage des formules mathématiques de gauche à droite suivant le modèle latin. Nous nous basons sur l'extension de FIGUE pour proposer de nouveaux développements de la norme MathML pour l'affichage bidirectionnel de formules mathématiques. Pour adopter un algorithme bidirectionnel pour les formules

لتكن f الدالة المعرفة بمايلي: $f(x) = ax^2$ حيث $a \in \mathbb{R}$

(a)

```
<mrow>
  <mtext> لتكن </mtext>
  <mi> f </mi>
  <mtext> الدالة المعرفة بمايلي: </mtext>
  <mrow> ... </mrow>
  <mtext> حيث </mtext>
  <mrow> ... </mrow>
</mrow>
```

(b)

Figure 6: (a) Texte mathématique, modèle marocain et (b) sa représentation MathML.

mathématiques, MathML doit définir un nouveau attribut *lang* identifiant la langue ou le modèle de la formule. Par exemple, la valeur de cet attribut est *arabe-marocain* dans le cas des formules marocaines. Pour chaque élément MathML, cet attribut peut être défini explicitement, ou déduit. Une fois cet attribut défini, il sera interprété pour ajuster le formatage des formules en fonction du modèle choisi (par exemple appliquer les règles d'affichage spécifiques au modèle marocain).

L'algorithme d'affichage bidirectionnel requiert un attribut *dir* pour définir la direction primitive d'écriture. La direction d'une formule tout entière peut être définie par l'élément racine *math* où déduite selon la direction primitive d'écriture du texte du document entier. A travers les propriétés des éléments et leurs attributs de langue et de direction, MathML offrira le moyen de contrôler le changement de direction du texte et des formules mathématiques, et contiendra un algorithme bidirectionnel d'écriture pour les formules mathématiques. L'implémentation de cet algorithme bidirectionnel par des navigateurs Web supportant MathML, permettra une large diffusion de documents scientifiques sur Internet en différentes langues. Cela permettra ainsi de diffuser sur Internet le développement de preuves en différentes langues (par exemple, diffuser des explications de preuves en langue arabe sur Internet).

3 Application: explication de preuves en langue arabe dans PCOQ

PCOQ fournit un environnement de travail graphique et interactif écrit en JAVA pour le système de preuve COQ. Le but principal de PCOQ est d'aider au développement de preuves à grande échelle et de fournir une interface conviviale facilitant la création de preuves pour les utilisateurs du système de preuve COQ. FIGUE est utilisé dans PCOQ pour l'affichage des commandes et des formules mathématiques manipulées par l'interface.

Les explications de preuves en langue naturelle

PCOQ offre une fonctionnalité intéressante : la production de textes en langue naturelle à partir de preuves formelles [2, 5]. Cette fonctionnalité permet de présenter les preuves d'une façon proche des démonstrations mathématiques habituelles et de les rendre plus accessibles. A partir de la structure d'arbre représentant la preuve dans COQ, PCOQ produit du texte correspondant à chaque noeud de cet arbre de preuve. Dans le cas d'un noeud dont la formule logique n'est pas encore traitée, la phrase "Imaginez une preuve de" suivie de cette formule est choisie. Pour les autres types de noeuds, une phrase décrivant la tactique (commande) appliquée à ce noeud est générée. Par exemple, pour la commande

"Induction n.", nous générons le texte français "Procédons par récurrence sur n: Cas 0: ... Cas (s): ...".

Aujourd'hui dans PCOQ, il existe des explications en langue française et anglaise. Nous expérimentons l'implémentation des explications en arabe, ce qui permet de tester notre algorithme d'affichage bidirectionnel dans FIGUE pour écrire du texte de droite à gauche et même mélanger avec des formules mathématiques écrites de gauche à droite.

Les explications de preuves en arabe

La production d'un arbre d'explications de preuves en arabe peut se faire de deux façons : la première consiste à produire un arbre textuel arabe à partir de l'objet preuve et la deuxième façon consiste à reprendre l'arbre textuel anglais (produit par un autre module) pour le traduire en un arbre textuel arabe. Dans les deux cas, une fois que l'arbre textuel arabe produit, nous appliquons des règles de transformations sur cet arbre pour produire l'arbre de boîtes FIGUE (décrivant le formatage correspondant).

Nous nous contentons dans un premier temps de traduire l'arbre textuel anglais en un autre arbre textuel arabe en appliquant des règles simples de traduction. Cette traduction mot à mot n'est pas toujours adaptée pour bien exprimer des preuves en langue arabe. Nous étudions comment produire un arbre textuelle en arabe directement à partir de l'objet représentant la preuve.

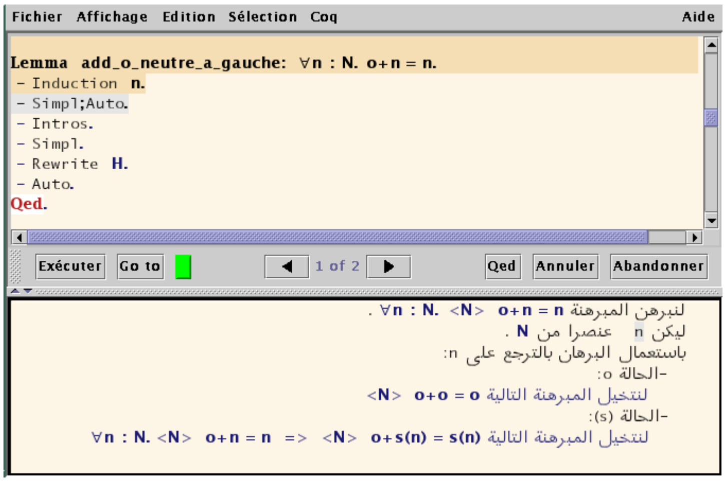


Figure 7: Un exemple de session PCOQ où les explications de preuves sont en arabe.

La figure 7 montre un exemple de session PCOQ où les explications de preuves sont en arabe. La première fenêtre de cette session PCOQ représente les commandes et le script de la preuve et la deuxième fenêtre affiche l'explication en arabe générée automatiquement à partir de la preuve[1]. Cet exemple est la preuve en COQ d'une propriété de l'addition dans l'ensemble $\mathbb{N}$ des entiers de Peano. Cet ensemble est défini par récurrence et il est constitué d'une constante *o* et de ses successeurs obtenus par la fonction s. Dans cet exemple, nous utilisons la tactique *Induction* qui lance une preuve par récurrence sur son argument, qui

[1]La traduction en anglais du texte de la même preuve est : Let us prove $\forall n : N0 + n = n$. Let n be an element of N. By induction on n: -Case 0: Imagine a proof of $0 + 0 = 0$. -Case (s): Imagine a proof of $\forall n : N0 + n = n \Rightarrow 0 + s(n) = s(n)$.

doit être quantifié universellement dans le but. Le texte d'explication de cette preuve en arabe s'écrit de droite à gauche et les formules mathématiques de gauche à droite (modèle marocain). Il est possible de changer la langue utilisée pour les explications de preuves durant la même session PCOQ (arabe, anglais, français).

4 Conclusion

Cette étude, illustrée par notre moteur FIGUE, démontre des idées et des concepts qui sont généralisés pour tout affichage de formules mathématiques dans un contexte multi-langues, que ce soit sur Internet ou dans une interface interactive. En ce moment, FIGUE sert comme base d'expérimentation pour établir de nouvelles recommandations de MathML pour l'affichage bidirectionnel des formules mathématiques. Il est aussi utilisé dans notre interface PCOQ pour présenter des explications en langue arabe de preuves mathématiques.

Jusqu'à présent, nous étudions uniquement l'affichage bidirectionnel des formules mathématiques dans le cas des langues écrites de droite à gauche comme l'arabe et l'hébreu ou de langues écrites de gauche à droite comme le français et l'anglais. Il nous reste à généraliser cette étude pour le cas des langues écrites de haut en bas comme le japonais : étudier comment mélanger les différentes directions (LR, RL, haut en bas) dans le même texte et établir de nouvelles règles d'affichage des formules mathématiques écrites verticalement.

References

[1] The Bidirectional Algorithm. http://www.unicode.org/unicode/reports/tr9/bidi.

[2] A. Amerkad, Y. Bertot, L. Pottier, and L. Rideau. Mathematics and proof presentation in pcoq. In *Proceedings of Workshop Proof Transformation and Presentation and Proof Complexities*, Siena, Italie, June 2001.

[3] Joseph D. Becker. Multilingual word processing. *j-SCI-AMER*, 251(1):96–107 (Intl. ed. 82–93), July 1984.

[4] Joseph D. Becker. Arabic word processing. *Communications of the ACM*, 30(7):600–610, July 1987.

[5] Y. COSCOY. A natural language explanation for formal proofs. In *Proceedings of the Int. Conf. on logical Aspects of Computational Linguistics (LACL)*, volume 1328, Nancy, September 1996.

[6] G. Huet, G. Kahn, and C. Paulin-Mohring. The coq proof assistant : A tutorial : Version 6.1. Technical Report RR-0204, Inria, Institut National de Recherche en Informatique et en Automatique, Decembre 1997.

[7] D. Knuth. *The TEXbook*, volume A of *Computers and typesetting*. Addison-Wesley, Reading, MA, USA, 23rd printing, with corrections. edition, 1993.

[8] K. Lagally. ArabTEX — typesetting Arabic with vowels and ligatures. In *EuroTEX 92: Proceedings of the 7th European TEX Conference*, pages 153–172, Brno, Czechoslovakia, September 1992. Masarykova Universita.

[9] A. LAZREK. Aspects de la problématique de la confection d'une fonte pour les mathématiques arabes. Number 39-40. Cahiers GUTenberg, Mai 2001.

[10] A. LAZREK. A package for typesetting arabic mathematical formulas. In *DANTE*, 2001.

[11] H. Naciri and L. Rideau. Affichage interactif, bidimensionnel et incrémental de formules mathématiques. In *Proceedings of the fifth African conference on research in computer science*, Madagascar, Octobre 2000.

[12] H. Naciri and L. Rideau. The mariage of mathml and theorem proving. In *Proceedings of the IAMC'2001 Workshop on Internet Accessible Mathematical Computation*, Western Ontario Canada., July 2001.

[13] J. Srouji and D. Berry. Arabic formatting with `ditroff/ffortid`. *Electronic Publishingemdash Origination, Dissemination, and Design*, 5(4):163–208, December 1992.

[14] I. Vatton. W3c's amaya 4.0 editor/browser. Technical report, W3C, 2000. http://w3c1.inria.fr/Amaya/.