ỨNG DỤNG ÔTÔMAT HỮU HẠN ĐỂ QUẢN LÝ CÁC THÔNG ĐIỆP CÓ CHỨA BIẾN NHỚ VÀ CÁC BIẾN THỂ CỦA CHÚNG TRONG MÔI TRƯỜNG ĐA NGỮ

APPLICATION AUTOMATA TO MANAGE MESSAGES WITH VARIABLES AND VARIANTS IN MULTILINGUAL SOFTWARE

VÕ TRUNG HÙNG
Trường Đại học Bách Khoa, Đại học Đà Nẵng

TÓM TẮT
Trong bài báo này, chúng tôi giới thiệu một phương pháp mới để quản lý những thông điệp trong các phần mềm đa ngữ. Chúng tôi đề xuất một giải pháp mới sử dụng các ôtômat hữu hạn có điều khiển, được gọi là "ôtômat thông điệp" (MA), để mô hình hóa các thông điệp (có hoặc không chứa các biến nhớ).

ABSTRACT
In this article, we present our method in order to manage messages with variables and variants in multilingual software. We propose the use of loop-free FSAs, called "message automata" (MA), to model messages (with or without variants).

1. GIỚI THIỆU

Trong những chương trình máy tính, chúng ta thường gặp những thông điệp (message) có chứa những biến nhớ (variable). Những biến nhớ này kéo theo những biến đổi bên trong thông điệp, nguyên nhân tạo ra những biến đổi này là do sự tương hợp về giống, số, cách, mức độ lịch sự và vị trí của các biến…. Ví dụ, khi chúng ta viết trong chương trình một thông điệp "$n tập tin đã được xóa" thì trong tiếng Pháp ta phải phân ra hai trường hợp "$n fichier supprimé" ($n=1), "$n fichiers supprimés" ($n=2, 3, …) hoặc trong tiếng Anh "$n file deleted" ($n=1), "$n files deleted" ($n=2, 3, …). Trong trường hợp trên, đối với tiếng Anh và tiếng Pháp việc sử dụng hậu tố -s phụ thuộc vào giá trị của biến n (1, 2, 3…).

Việc xử lý những biến thể như vậy luôn luôn là một vấn đề rất khó khăn trong những hệ thống đa ngữ bởi vì các kiểu biến đổi là rất khác nhau giữa các ngôn ngữ [1]. Ví dụ, tiếng Nga có 3 dạng thức số nhiều của danh từ giống đực: những biến thể của архив (tập tin) là архив, архива, архивов: nếu những số có chữ số cuối là 1 và không phải là 11 thì dùng архив, nếu số cuối bằng 2, 3, 4 và không phải là số 12, 13, 14 thì dùng архива, những trường hợp còn lại thì dùng архивов. Đa số các ngôn ngữ châu Á chỉ sử dụng một dạng thức dành cho danh từ. Dành cho số từ, tiếng Pháp và tiếng Nga có hai biến thể, còn tiếng Anh có 4 biến thể (-st, -nd, -rd, -th).

Những hệ thống như gettext và catgets cho phép "bản địa hóa" (localisation) dễ dàng những tập tin thông điệp nhưng chúng chưa thể xử lý những biến đổi ngôn ngữ bên trong những thông điệp [7]. Một phương pháp thông dụng hiện nay là người ta sử dụng một thông điệp đơn giản "$n fichier(s) supprimé(s)" hoặc thay đổi chương trình xử lý cho từng ngôn ngữ, ví dụ dành cho tiếng Pháp:

```
if (n<2)   printf("%d fichier supprimé", n);
else       printf("%d fichiers supprimés", n);
```

và dành cho tiếng Nga:

```
if (n<2) || ((n%10 =1) && (n>20))
     printf("%d архив извлекал", n);
else if (n>=2 && n<=4) || (n>20 && n%10>2 && n%10<=4)
```

```
            printf("%d архива извлекали", n);
          else
            printf("%d архивов извлекали", n);
```

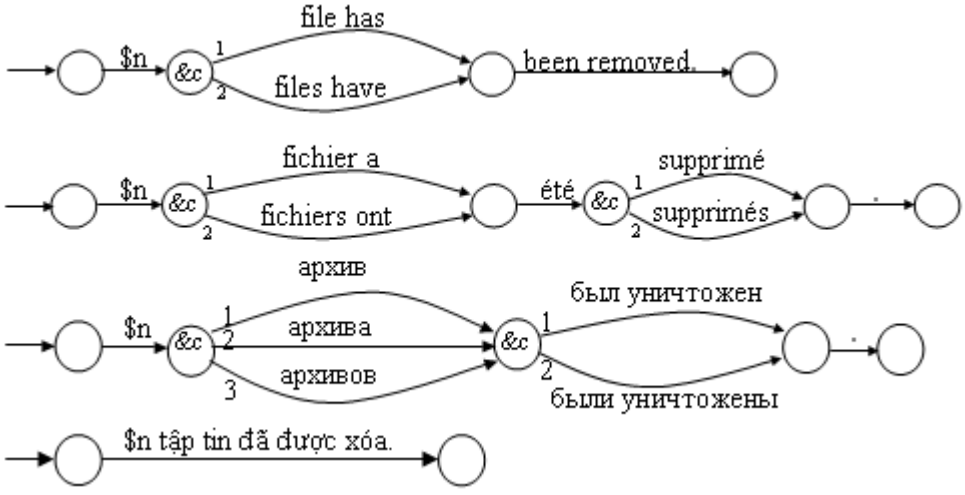
Trong phần lớn các kỹ thuật lập trình hiện nay, người ta hài lòng với việc thay thế những biến của chương trình bởi giá trị của chúng mà không xử lý những biến này và những biến thể tạo ra bởi chúng [10].

Kỹ thuật mà chúng tôi đề xuất được thực hiện như sau. Trước hết, chúng tôi biểu diễn tất cả các biến thể của một thông điệp (luôn là một tập hữu hạn) trong một ôtômat hữu hạn có thể điều khiển được. Tiếp theo, để dịch một thông điệp M trong ngôn ngữ L1 sang ngôn ngữ L2, chúng tôi phát sinh trong L1 một xâu ký tự (định dạng) của mỗi biến thể của M tương ứng các trường hợp có thể ở L2. Tiếp đến, dịch chúng sang L2 và sản sinh trong L2 một ôtômat của thông điệp này.

2. GIẢI PHÁP ĐỀ XUẤT

2.1. Mô hình hóa những thông điệp có chứa biến và biến thể

Automat thông điệp là những ôtômat hữu hạn "được điều khiển". Dành cho mỗi ngôn ngữ, ôtômat có thể là khác nhau [3]. Ví dụ, thông điệp "$n tập tin đã được xóa", có thể được trình bày trong tiếng Anh, tiếng Pháp, tiếng Nga và tiếng Việt dưới dạng những ôtômat như sau:



Hình 1. Những ôtômat thông điệp theo từ

Bắt đầu từ một ôtômat của thông điệp, người ta có thể xây dựng một bảng những biến thể của thông điệp này. Ví dụ, thông điệp "n fichier(s) supprimé(s)" trong tiếng Anh, tiếng Pháp và tiếng Nga được trình bày trong các bảng sau:

Dành cho tiếng Anh và tiếng Pháp, biến &c là được định nghĩa bởi điều kiện $n<=1. Ở đó, hai biến thể (dưới định dạng có thể sử dụng) tương ứng với 2 khả năng:

Bảng 1. Những biến thể của thông điệp trong tiếng Anh và tiếng Pháp

&c	Điều kiện	Tiếng Anh	Tiếng Pháp
1	$n <= 1	$n file has been removed	$n fichier a été supprimé
2	Sinon	$n files have been removed	$n fichiers ont été supprimés

Dành cho tiếng Nga, biến &c được định nghĩa bởi hai điều kiện ($n<2) || (($n%10 =1) && ($n>20)) và ($n>=2 && $n<=4) || ($n>20 && $n%10>=2 && $n%10≤4). Có 3 cung tương ứng ở mỗi nút lựa chọn, người ta giả sử rằng trường hợp thứ 3 dành cho khả năng còn lại (sinon, hai điều kiện 1 và 2 đều sai). Đây là 3 biến thể:

Bảng 2. Những biến thể dành cho thông điệp tiếng Nga

&c	Điều kiện	Tiếng Nga
1	($n<2) \|\| (($n%10 =1) && ($n>20))	$n архив извлекал
2	($n>=2 && $n<=4) \|\| ($n>20 && $n%10>=2 && $n%10≤4)	$n архива извлекали
3	Sinon	$n архивов извлекали

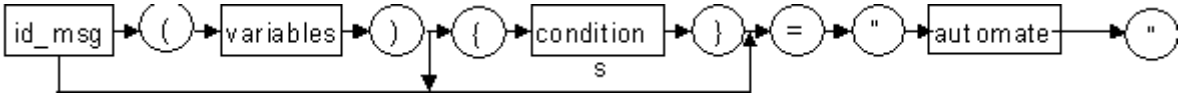
Để mô hình hóa ôtômat trên, chúng tôi định nghĩa những biến điều khiển nguyên &ci bắt đầu luôn luôn bằng 1, tiếp đến tăng lên một đơn vị, nghĩa là nó có giá trị trong miền [1..ni+1], ở đây ni là số cung phát xuất từ nút &ci. Nếu &ci = ni+1, lộ trình tương ứng sẽ kết thúc ở nút này. Ở mỗi một tổ hợp các giá trị của biến điều khiển sẽ tương ứng với một "quĩ đạo" trong ôtômat, nó chứa chỉ một biến thể của thông điệp [3].

2.2. Biểu diễn các thông điệp trong chương trình

Chúng tôi thực hiện một sự phân ly hoàn toàn giữa mã chương trình (code source) và những thông điệp của chương trình này (qua công cụ GetAMsg). Để thực hiện điều này, người ta có thể sử dụng những công cụ của gettext để trích những thông điệp ra khỏi chương trình nguồn [4]. Một tập tin thông điệp (ví dụ: FicMes1_fre.adm) chứa những thông điệp, hoặc chính xác hơn là những ôtômat của các thông điệp, dành cho một ngôn ngữ cho trước (ở đây là tiếng Pháp). Nhiều chương trình thường phải dùng chung cùng một tập tin thông điệp, và một chương trình có thể sử dụng nhiều tập tin thông điệp khác nhau, nhưng chỉ một tập tin là được mở tại một thời điểm.

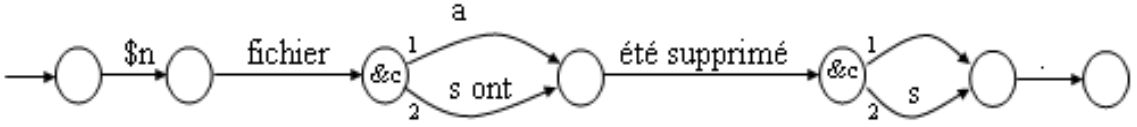
Trong phiên bản hiện tại, mỗi thông điệp là xác định bằng một tên xác định trong tập tin thông điệp, giống như catgets.

Cú pháp tổng quát của một ôtômat thông điệp [6]:



Hình 2. Cú pháp tổng quát của một thông điệp

Ví dụ, thông điệp "$nbfichier fichier(s) a (ont) été compilé(s)." có thể là được trình bày bởi một ôtômat như sau, với &c=1 nếu $nbfichier<2 và ngược lại thì &c=2:



Hình 3. Ví dụ của một ôtômat thông điệp

Nếu định danh của thông điệp này là M1, chúng ta viết trong tập tin thông điệp (với %3d dành cho định dạng số):

```
M1($n_fichier){&c:$n_fichier<2}= "$n_fichier%3d fichier[&c: a | s ont] été compilé[&c:  | s]."
```

Trong một chương trình, chúng ta sử dụng hàm getamsg() với những tham biến bắt đầu với tên và kiểu dữ liệu của biến thông điệp, tiếp đến là những giá trị của các biến này. Ví dụ, ta gọi getamsg("M1 n_", nf) để sản sinh một định dạng không chứa các biến trong những ngôn ngữ lập trình (C/C++, Pascal, Perl, etc.).

Để thực hiện lệnh in trong C, chúng ta viết: 'printf(getamsg("M1 n_", nf));' và trong Pascal 'write(getamsg("M1 n_", nf));'.

Đây là những kiểu biến được định nghĩa trước dành riêng cho GetAMsg. Chúng tôi sử dụng những tiền tố "$o_" để tránh việc khai báo lại, tuy nhiên người sử dụng có thể viết như cách thông thường là "variable: type". Trong phiên bản hiện tại, bảng khai báo biến này là cố định nhưng trong phiên bản đến nó là có thể thay đổi bởi người sử dụng, điều này cũng đồng nghĩa với việc cho phép người sử dụng bổ sung thêm các kiểu biến mới.

Để rút gọn những biểu thức điều kiện cho phép xác định các giá trị điều khiển trong những ôtômat thông điệp, chúng tôi cố định từng kiểu của biến và từng loại ngôn ngữ bởi những tên và giá trị tương ứng với "mỗi trường hợp cụ thể".

Bảng 3. Những trường hợp được định nghĩa trước

Biến	Điều kiện			
	Tiếng Anh	Tiếng Pháp	Tiếng Nga	…
$g_	HE: $g_=m SHE: $g_=f IT: $g_=n	IL:$g_=m ELLE: $g_=f ÇA:$g_=n	ОН:$g_=m ОНА: $g_=f ЭТО:$g_=n	
$p_	FA: $p_≤ 2 POLI: $p ≥ 3	FA: $p_≤ 1 POLI: $p ≥ 2	БЛИЗКИЙ: $p_=1 ВЕЖЛИВЫЙ: $p_≥ 2	
$t_	NONE: $t_=0 MR_MRS:$t_=1 FUNCTION:$t_=2 RANK:$t_=3	SANS:$t_=0 M_MME:$t_=1 FONCTION:$t_=2 GRADE:$t_=3	БЕЗ: $t_=0 (sans titre) ГОСП: $t_=1 (Mme…) ЧИН: $t_=2 (Directeur…) ФУНКЦИЯ:$t_=3 (Dr…)	
$n_	SINGULAR:$n_≤1 PLURAL:$n_≥2	SINGULIER: $n_≤1 PLURIEL: $n_≤2	ОДИН: ($n_<2) ‖ (($n_%10 =1) && ($n_>20)) ДВА: (2≤$n_ && $n_≤4) ‖ ($n_>20 && 2≤$n_%10&& $n_%10)≤4) ПЯТЬ: ($n_>0)&&(5≤$n_%10) &&(($n_%10=0)‖ (%≤$n_%10 && $n_%10≤9))	
$a_	BABY:$a_≤5 CHILD: 5<$a_≤13 TEEN: 13≤$a_<20 ADULT:20≤$a_≤60 SENIOR:60<$a_≤80 OLD: $a >80	BEBE:$a_≤5 ENFANT:5<$a_≤13 ADO: 13≤$a_<20 ADULTE:20≤$a_≤60 SENIOR:60<$a_≤80 VIEUX: $a >80	РЕБЁНОК: $a_≤5 ДИТЯ: 5<$a_≤13 ЮНОША: 13≤$a_<20 ВЗРОСЛЫЙ:20≤$a_≤60 ПОЖИЛОЙ: 60<$a_≤80 СТАРИК: $a >80	
$c_	JULIAN: $c_=1 GREGORIAN:$c_=2 JAPANESE: $c_=3 BUDDHIST: $c_=4 ISLAM: $c_=5 …	JULIEN: $c_=1 GREGORIEN:$c_=2 JAPONAIS: $c_=3 BOUDDHISTE:$c_=4 ISLAM: $c_=5 …	ДЖУЛИАН: $t_=1 ГРИГОРИАНСКИЙ:$t_=2 ЯПОНСКИЙ: $t_=3 БУДДИСТ: $t_=4 ИСЛАМ: $t_=5 …	
$h_	AM: $h_ ≤ 12 PM: 12 < $h_≤ 24	AM: $h_ ≤ 12 PM: 12 < $h_≤ 2	БЕЗ (hh)	
$o_	FIRST:$o_%10=1 SECOND:$o_%10=2 THIRD:$o_%10=3 OTHERWISE: sinon	PRE: $o_%10=1 AUTRES: sinon	ОЙ: ($o_%10=2 || $o_%10=6 || $o_%10=7 || $o_%10=8) && ($o_<12 || o_<20) ЫЙ: sinon	

Những định nghĩa này được trình bày trong một tập tin dạng TXT và có thể được sửa đổi bởi người sử dụng. Người sử dụng cũng có thể thay đổi những tên được sử dụng trong các điều kiện, thay đổi số các trường hợp, thay đổi biểu thức điều kiện, và bổ sung những thông tin tương ứng cho các ngôn ngữ mới.

Ví dụ, chúng ta sử dụng (M, F, N) dành cho giống đực, giống cái và giống trung, nhưng người ta có thể thay thế những định danh như (IL, ELLE, ÇA) trong tiếng Pháp, bởi (HE, SHE, IT) trong tiếng Anh, hoặc (OH, OHA, ЭТО) trong tiếng Nga. Người ta cũng có thể thay đổi các biểu thức điều kiện, ví dụ, phân biệt 3 trường hợp dành cho số từ trong tiếng Pháp (0, 1, và >1).

2.3. Phát triển một công cụ quản lý các thông điệp

Chúng tôi đã phát triển một công cụ thử nghiệm để quản lý các thông điệp đa ngữ và đặt tên là GetAMsg. GetAMsg được trình bày như một thư viện của ngôn ngữ C. Nhờ vào những công cụ javah, h2pas và h2xs, chúng ta có thể biên dịch một thư viện .h trong C thành những thư viện trong Java, Pascal, Perl và chúng ta có thể sử dụng những thư viện này trong các ngôn ngữ lập trình kể trên. Đây là kỹ thuật sử dụng bởi gettext, và tồn tại rất nhiều những công cụ như vậy dành cho nhiều ngôn ngữ lập trình khác, nhưng chúng tôi còn chưa ứng dụng trên GetAMsg.

Để sử dụng hàm này, những lập trình viên phải tuân thủ một số qui định. Trước hết, người ta trích những thông điệp và đặt chúng trong các tập tin, dưới dạng những ôtômat thông điệp. Cùng lúc đó, đương nhiên chúng ta phải sửa đổi chương trình nguồn để gọi các thông điệp này. Tiếp theo, người ta sử dụng hàm textmsg() để biên dịch từng tập tin thông điệp, ví dụ, FicMes1_fre.adm đến FicMes1_fre.amg.

Chúng ta bổ sung những thiết lập hoặc những chương trình liên quan để lựa chọn ngôn ngữ làm việc:

```
#include <locale.h>
char *setlocale (int category, char *locale )
```

Chúng ta cũng phải thiết lập những biến toàn cục.

Việc sửa đổi mã nguồn chương trình để gọi các thông điệp dựa trên hàm getamsg(), cú pháp của nó như sau:

```
char *getamsg(char *id_msg, expression-1, expression-2, ..., expression-n)
```

- id_msg: định danh của thông điệp và của những kiểu biến cục bộ, cũng như những tham số cần thiết. Ví dụ, "001 n_ o_" thì định danh của thông điệp là 001 và sử dụng hai tham số, tham số thứ nhất kiểu số từ chỉ số lượng và tham số thứ hai thuộc kiểu số từ chỉ thứ tự. Những khai báo này là cần thiết để sử dụng một danh sách những tham số có số lượng không xác định trước.
- expression-i: giá trị của biến cục bộ thứ i.
- Giả sử rằng chúng ta có một ví dụ có sẵn viết trong ngôn ngữ C/C++:

```
printf("Bonjour !");
printf("%d fichier(s) supprimé(s)", nbfichier);
```

Chúng ta khởi tạo 4 tập tin thông điệp cho tiếng Pháp, tiếng Anh, tiếng Nga và tiếng Việt:

- Tập tin thông điệp tiếng Pháp:

```
msg1= "Bonjour !"
msg2($n_fichier){&c: ($n_fichier<2)} =
"$n_fichier fichier[&c:|s] supprimé[&c:|s]."
```

- tiếng Anh:

```
msg1= "Hello !"
msg2($n_file){&c:($n_file<2)}="$n_file file[&c:|s] removed."
```

- tiếng Nga:

```
msg1= " Здравствуйте !"
msg2($n_архив){&c: ОДИН($n_архив), ДВА($n_архив)}=
   "$n_архив архив[&c:|а|ов] извлекал[&c:|и|и]."
```

- và tiếng Việt:

```
msg1= " Xin chào!"
msg2($n_taptin)="$n_taptin tập tin đã xóa."
```

Chúng ta sử dụng một mã nguồn duy nhất được viết như sau:

```
printf(getamsg("msg1"));
     /* msg1: định danh của thông điệp, không co biến */
printf(getamsg("msg2 n_", nbfichier));
     /* msg2: định danh của thông điệp, nbfichier: biến */
```

3. KẾT LUẬN

Chúng tôi đã trình bày một phương pháp cho phép tổ chức và xử lý những thông điệp trong các phần mềm đa ngữ. Giải pháp của chúng tôi đã được thí nghiệm trên công cụ GetAMsg, dựa trên việc sử dụng những ôtômat hữu hạn có điều khiển để trình bày và quản lý những

thông điệp có chứa các biến nhớ và những biến thể của chúng. Phương pháp này cho phép dễ dàng lưu trữ và thu hồi một thông điệp duy nhất tương ứng với một quĩ đạo (trajectoire) xác định trong ôtômat thông điệp.

Ưu điểm của phương pháp này so với các phương pháp khác (gettext, catgets [5]) là việc quản lý những biến thể của thông điệp trong nhiều ngữ cảnh và nhiều ngôn ngữ khác nhau. Công cụ GetAMsg là một tập hợp những chương trình viết trong C. Chúng tôi cũng đã tạo ra các thư viện tương ứng để sử dụng trong C++, Java và Perl.

Trong thời gian đến, chúng tôi đề xuất một sự cải tiến GetAMsg trong việc tham số hóa những kiểu biến cần xử lý, cũng như những điều kiện kèm theo dành cho từng ngôn ngữ.

Chúng tôi cũng đã bắt đầu khai thác khả năng để trình bày một thông điệp chứa những biến nhớ dưới dạng một đồ thị dạng UNL [2], [8], [9] với các biến nhớ và độc lập với những ngôn ngữ. Với phương pháp này, ta có thể biểu diễn một ôtômat thông điệp dưới dạng một đồ thị UNL dành cho tất các ngôn ngữ (bằng bộ chuyển đổi "enconvertisseur" bắt đầu từ một trường hợp của ôtômat thông điệp trong một ngôn ngữ bất kỳ), và gửi đến bộ dịch "desconvertisseur" đồ thị UNL (cùng với những giá trị tương ứng) để nhận được kết quả là các trường hợp tương ứng trong ngôn ngữ đích.

TÀI LIỆU THAM KHẢO

[1] Christian Boitet, 1980 – 90: TAO du réviseur et TAO du traducteur, *trương hè ở Lannion, www-clips.imag.fr/geta/christian.boitet/pages_personnelles/*, bản tiếng Anh trong kỷ yếu của *hội nghị quốc tế ROCling-90*, Taipeh, Đài Loan.

[2] Christian Boitet, A rationale for using UNL as an interlingua and more in various domains, *Proceeding of the First International Workshop on UNL, other Interlanguages and their Applications*, LREC2002, Las Palmas, Tây Ban Nha, 5-2002.

[3] Christian Boitet, Message Automata for Messages with Variants, and Methods for their Translation, *Proceeding of the CICLING 2005*, Mexico, 2-2005, Springer LNCS 3406, các trang 352—371.

[4] U. Drepper, J. Meyering, F. Pinard, B. Haible, *GNU gettext tools, version 0.11.2*, Published by the Free Software Foundation, 4-2002.

[5] IBM Corporation, *International Components for Unicode (ICU) – User's Guide*, html:// oss.software.ibm.com/icu/userguide, 2003.

[6] Võ Trung Hùng, "Méthodes et outils pour utilisateurs, développeurs et traducteurs de logiciels en contexte multilingue", luận án Tiến sĩ tin học, xuất bản bởi Viện Đại học Bách khoa Quốc gia Grenoble (Institut National Polytechnique de Grenoble), 12-2004.

[7] Sun Microsystems Inc., *Building International Applications*, Sun product documentation http://docs.sun.com/db/doc/806-6663-01.

[8] Wang-Ju Tsai: "La coédition langue-UNL pour partager la révision entre langues d'un document multilingue", luận án Tiến sĩ tin học, Université Joseph Fourier, 7-2004.

[9] Uchida Hiroshi, "*The Universal Networking Language beyond Machine Translation",* International symposium on language in cyberspace, 9-2001, Seoul, Hàn Quốc.

[10] David A. Wheeler, *Secure Programming for Linux and Linux HOWTO*, http://www.dwheeler.com/secure-programs/, 3-2003.