Editorial

Natural Language Processing is a growing field in computer science and engineering, related to both theoretical and applied computational techniques relative to natural language and the use of language knowledge. This issue of the research journal "Polibits" presents four contributions concerning natural language processing. Two of them draw formal models of two categories of knowledge of language: syntax and morphology. The other two papers solve practical computational approaches of language to machine translation and information extraction.

The paper "*Natural Language Syntax Description using Generative Dependency Grammar*" gives a practical solution, based on a Generative Dependency Grammar (GDG), to describe natural language syntax. It also presents the features and what GRAALAN, –declarative computer language in which GDG are implemented– offers.

The paper "*Morpheme based Language Model for Tamil Part-of-Speech Tagging*" presents a POS tagging using a corpus-based approach by formulating a Language Model through morpheme components of words Tamil language. An approach to a language model is also given, in which, in order to estimate the contribution factors, the authors follow generalized iterative scaling technique.

In "*Modeling a Quite Different Machine Translation using Lexical Conceptual Structure*", the author outlines the readability of an Example-Based Machine Translation for any pair of languages by means of the language-independent properties of the lexical conceptual structure (LCS), which is described as a representation of traditional relationships. The author presents LCS-Based Machine Translation from the point of view of a complex adaptive system.

In the paper "*Named Entity Recognition in Hindi using Maximum Entropy and Transliteration*", the authors have explored different features applicable for the Hindi Named entity recognition (NER) task, as well as incorporated some gazetteer lists in the system to increase its performance. A two-phase transliteration methodology is proposed, which is not only applicable for Hindi, but also for other languages.

Additionally, this issue of "Polibits" contains six regular papers addressing research of computer science applied to vision and signal processing, as well as to design of interfaces. In this way, this journal aims at the purpose of spreading the vast discipline of computer science and engineering.

A definition of the simple algorithm for triangulation of the virtual object virtual, as well as an algorithm that allows visualizing of the cutting triangular net and the calculation of the dynamics of the net during the cut are presented in the paper "*Visualización 3D de Deformación y Corte de Objetos Virtuales basada en Descomposición Ortogonal* (*3D visualization of deformation and cut of virtual objects based on orthogonal decomposition*)".

In the paper "*An Extended Video Database Model for Supporting Finer-Grained Multi-Policy and Multi-Level Access Controls*", the authors present a hybrid video database model. They also extend the original hierarchical indexing mechanism to add frames and salient objects at the lowest granularity level in the video tree with the aim to support multi-level access control.

In "*Multiplicador Electrónico para Encoder Incremental* (*Electronic multiplicator for incremental encoder*)", the design and experiments on simulation of the electronic multiplicator for incremental encoders are presented, which purpose is to increment the resolution of the feed back signal using the same encoder.

The term "Distance Object Learning" as a way of learning over a computer network or the Internet about real world entities that are distinguishable from others is used in the paper "*Distance Online Learning and Evaluation Framework*". The Distance Object Learning and Evaluation (DOLE) system concept is presented that uses standards for Learning Object Metadata (LOM), and it is based in part on an earlier version of E-learning Assessment System for Young learners (EASY).

The paper "*Computadoras de Bolsillo como una Alternativa para el Control de Servomotores en Robótica* (*PDA Computers as an Alternative for Servo Motors Control in Robotics*)" proposes an implementation which is related to hardware interface, namely, to the usage of the specialized microcontroller that connects PDA with the servo motor using serial port of the PDA.

In "*Diseño de un Coprocesador Matemático de Precisión Simple usando el Spartan 3E* (*Design of Mathematical Coprocessor of Simple Precision using Spartan 3E*)" the authors show how an implementation of the mathematical coprocessor using VHDL, for its further implementation in FPGA.

Gerardo Sierra
Head of the Group of Linguistic Engineering,
Institute of Engineering,
National Autonomous University of Mexico

Natural Language Syntax Description using Generative Dependency Grammar

Ştefan Diaconescu

Abstract— **The paper presents a practical solution to describe natural language syntax. This solution is based on a Generative Dependency Grammar (GDG). A theoretical definition of these grammars and some of their proprieties is given. GDG are implemented in a declarative computer language GRAALAN (Grammar Abstract Language). The paper shortly present the features of GRAALAN and, after that, a more detailed implementation of natural language syntax description is given. GRAALAN offers for natural language syntactic description some strong features that respond to the following requests: a compact description, the possibility to express the syntax and the agreement and to specify the errors met in a text. The description has also the feature of reversibility. The paper presents some conclusions concerning the using of GRAALAN to describe the syntax (among others natural language features).**

Index Terms—**Dependency grammar, natural language syntax.**

I. Introduction

THE approach of different linguistic chapters in a unified manner was realized so far in many complex systems like EUROTRA [1], EAGLES [2], ROSETTA [20]. These large projects did not produce many successful implementations, but they are very important at least from theoretical point of view. One of the major drawbacks (among others) was the lack of unity among different linguistic chapters approach. Paradoxically, this lack of unity has grown for the worse due to the (successful) standardization effort of the different linguistic chapter representation, because the extremely useful approach of each individual section was not sufficiently correlated with the approach of other linguistic sections [11]. The language GRAALAN (Grammar Abstract Language) that will be very shortly presented in section II of this paper try to integrate many chapters of linguistic description and among these, the syntactic description of a natural language.

A lot of language models and language grammar types were proposed trying to solve the natural language description problem. There are three of the linguistic models that seem to be more successful and used in some applications [18]: TAG – Tree Adjoining Grammar [16], HPSG – Head-Driven Phrase Structure Grammar [17] and LFG – Lexical Functional Grammar [19].

During the last years another idea was more and more analyzed and studied: the dependency. Actually, it is quite an old idea – usually [21] is used as reference but the dependency idea in the grammar is millennial – but new valences and strength became attractive. The present paper is based on some researches that try to make a connection between two directions that seemed to be almost irreconcilable till now: the generative approach and the dependency approach. We present how this connection was done and implemented in the syntactic section of a GRAALAN (section II) in order to find a more adequate language model that could be used in natural language processing and that have the potential to produce many and better applications.

Some theoretical notions that are used to build GRAALAN are presented in the section III: DT - Dependency Trees, AVT - Attribute Value Trees, GDG - Generative Dependency Grammar and GDGF - Generative Dependency Grammar with Features. In section IV, it is presented how GRAALAN is used to describe the syntax under the form of rule sequence that indicates: the syntactic elements, the dependencies between these elements and the agreement between these elements. Finally (section V) some conclusions and the stage of current implementations are presented.

II. GRAALAN: Grammar Abstract Language

GRAALAN is a language (in fact, a meta-language) that allows the description of a natural language and a correspondence between two natural languages. It contains some features that can be used to describe different natural language chapters (sections):

a) Alphabet Section defines the codes of the signs used to represent and describe the natural language. In this section the following information can be put: phonetic alphabet description (using, for example IPA – International Phonetic Alphabet [13]), normal alphabet and special characters (using. for example, UNICODE [14]), groups of characters (diphthongs or triphthongs, etc.) that contain the correspondences between some sequences of normal alphabet and phonetic alphabet, alphabetic classes (vowel class, consonant class, etc.). This section can describe also some special notation systems like those used by Japanese or Chinese languages.

b) Lexicon Section defines morphemes (roots, prefixes, suffixes, prefixoids, suffixoids, etc.), words (lemmas, some inflected forms of a word that accompanies the lemmas in an ordinary dictionary, for example, plural form of a noun), wordforms (some inflected form of another word that usually appears in a dictionary), multiword expression (MWE are groups of words represented as a DT - Dependency Tree),

Manuscript received July 10, 2008. Manuscript accepted for publication October 20, 2008.

This work was supported in part by SOFTWIN SRL, Bucharest, Romania.

Stefan Diaconescu is with the SOFTWIN SRL, Bucharest, Romania (e-mail: sdiaconescu@softwin.ro).

morphologic analytic structures, some typical syntactic structures (taken from the syntactic description), etc. For each lexicon entry some information belonging to the following types are present: semantic information (gloss, synonyms, antonyms, paronyms, hipernyms, hyponyms, connotations, homonyms, meronyms, etc.), etymology (original language, original form, transliteration of the original form), syllabification (euphonic, phonetic and morphologic), morphology (inflection situation, inflection rule identification, and segmentation), etc.

c) Syllabification Rules Section defines the syllabification rules for: euphonic syllabification (when the word is written with the normal or special alphabet), phonetic syllabification (when the word is written with the phonetic alphabet), morphologic syllabification (that respects the morphologic structure of the word). The elements of a word "separated" by syllabification (or not) are: the normal alphabet characters, groups (diphthongs, triphthongs, etc.) described in Alphabet Section (phonetic groups), some special characters, other constitutive elements (morphemes) described in Lexicon Section (morphologic groups).

d) Morphology Section defines morphologic categories and values. It is in fact an AVT - Attribute Value Tree (see section III.B), where attribute nodes are morphologic categories and value nodes are morphologic category values. Some information is attached with each type of node. For example, information attached to the attribute note is: the category name, the abbreviation of the category name, the indication if the category is inflected or not, (eventually) the name of a procedural program. Information attached to the attribute values are: the category value name, the abbreviation of the category value name, indication if it belongs to a lemma (or not), indication if it belongs to a lexicon entry (or not), (eventually) the name of a procedural program.

e) Inflection Rules Section defines the rules that can be used to generate the inflected forms. Lemma (from the lexicon) indicates a Compound rule. A compound rule is a list of basic rules. A basic rule contains an AVT where each leaf has one or more associated elementary inflection rules. An elementary inflection rule contains: a condition (logical expression) that indicates when the transformation sequence must be used, a transformation sequence (insert, delete, replace words or characters) acting on normal alphabet, a transformation sequence (insert, delete, replace words or characters) acting on phonetic alphabet form, an AVT for analytic forms, relations in a DT (dependency tree, see section III.A) for analytic forms.

f) Inflection Forms Section defines the inflected forms of the language. It contains an entry for an inflected form. An entry contains: the inflected form written using the normal alphabet, the inflected form written using the phonetic alphabet, the reference of the word in the lexicon whose inflected form is the current entry, the characterizing of the inflection situation (i.e., an AVT with lexical categories and lexical categories values), how the inflected form is syllabified in different situations: euphonic, phonetic, morphologic and at the end of the line (hyphenation).

g) Syntax Section defines the syntax rules (this section will be detailed in the following sections of the paper).

h) Bilingual Correspondences Section defines the correspondences between two languages for MWE (Multi Word Expression) correspondences [7] (it contains transformation rules based on dependency tree form of MWE, where nodes can be invariable elements, partial variable elements, total variable elements), word correspondences (particular cases of the MWE correspondences where both MWEs have only one word), syntactic structure correspondences (a particular case of MWE correspondences where the nodes can be non-terminals), morphologic analytic structure correspondences (a particular case of MWE where the correspondences is established between analytic inflection forms), morphologic sub-tree correspondences (a particular case of MWE too, that expresses the correspondences between a source morphologic sub-tree and a target morphologic sub-tree).

III. Graalan Theoretical Background

A. Dependency Tree

A generative dependency tree [3] is a 6-tuple $DT = \{N, T, P, A, SR, CR\}$ where:

- N - is the set of non-terminals $n: n\ (i_1, i_2, ...)$, $i_j \in SR$
- T - is the set of the terminals $t: t(i_1, i_2, ...)$, $i_j \in SR$
- P - is the set of pseudo-terminals $p: p\ (i_1, i_2, ...)$, $i_j \in SR$
- A - is the set of procedural actions $a: a(i_1, i_2, ...)$, $i_j \in SR$
- SR - is the set of subordinate relations $sr: sr(i_1)$, $i_1 \in N \cup T \cup \mathrm{P} \cup A \cup\ CR$
- CR - is the set of the coordinate relations $cr: cr(f_1, f_2, ... / s_1, s_2, ...)$, $f_i \in N \cup T \cup P \cup A \cup CR$, $s_i \in \mathrm{SR}$ ($f_1, f_2, ...$ are named fixed entry and $s_1, s_2, ...$ are named supplementary entry).

The non-terminals N are syntactic categories that can be described having a name and a structure.

The terminals T are words that can be found in the lexicon or can be obtained by applying some flexional rules on words from the lexicon.

The pseudo-terminals P are non-terminals that contain only terminals. When we will describe a dependency tree or a grammar we will not cover all the words from the lexicon because in this case the number of rules from the grammar can be too big. So, we can say that some non-terminals that we name pseudo-terminals (for example, some nouns or some verbs) will never be described in the grammar, but they are found in the lexicon.

The procedural actions (or "actions") A are the set of the routines that can be used to represent a certain portion of the text that we analyze. For example, a number represented like a sequence of digits or a mathematical formula or even an image with a certain significance that appear in a text can be "replaced" in grammars or dependency trees by a certain procedural action.

The subordinate relations *SR* are relations between a governor and a subordinate from the point of view of syntactic role in a phrase (for example the relation between a verb and a complement).

The coordinate relation *CR* are relations between two or many (but usually two) syntactic parts of a phrase, for example, the relation between *"read"* and *"write"* in the phrase *"I read and write."*. The coordinated elements are represented by the fixed entries. A coordinate relation can also be a governor for the elements that came eventually on its supplementary inputs (that means that the set of coordinated elements form a governor for the elements that come on the supplementary inputs).

A dependency tree can be represented using the graphical symbols from Fig. 1.

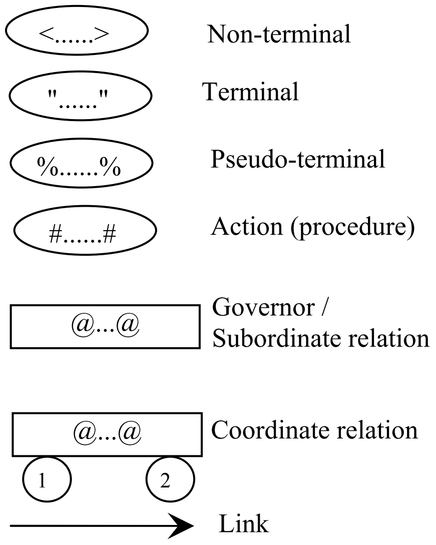


Fig. 1. Graphical DT symbols.

TABLE I.
LINKS IN A DEPENDENCY TREE

Link source	Link target				
	NTPA	GR	CR (supp. entry)	CR (fixed entry)	None
NTPA		1		2	7
GR	3		6		
CR		4		5	8
None	9		10		

The conditions respected by the links in a dependency tree are represented in Table I (there are 10 allowed situations) where we noted:

- NTPA: Non terminal (*N*) or Terminal (*T*) or Pseudo terminal (*P*) or Action (*A*).
- GR: governor/subordinate relation;
- CR: coordinates relation.

In a graphical representation:

- NTPA has maximum one input and maximum one output;
- GR has one input and one output;
- CR has maximum one output, zero, one or many supplementary input and a fixed number of fixed entry (we will consider only two fixed entry).

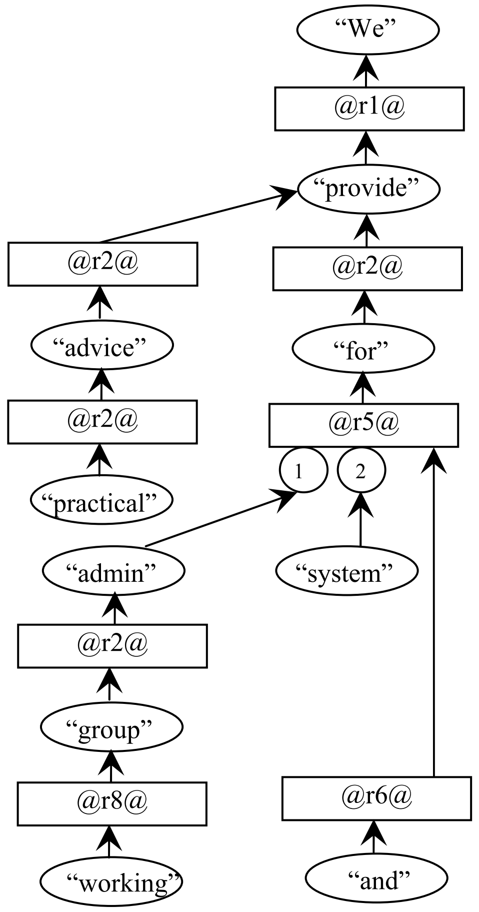


Fig. 2. Example of dependency tree.

We will consider also that the dependency tree is connected. As we can see, in this case, only one NTPA or one coordinate relation can have not output. This NTPA or coordinate relation will be named *head*.

The dependency trees will be used to define generative dependency grammar (see section III.C) and generative dependency grammar with features (see section III.D).

B. Attribute Value Tree

An attribute value tree (AVT) [4] [9] is used to describe morphologic or syntactic structures. It is in fact a list of attributes, each attribute having one or many values and each attribute value having associated one or many attributes. It can be defined as follows, using EBNF - Extended Backus-Naur Form from [22] without capital letter / lower case letter regular expression distinction:

[1] avt ::= ('{' S? label ':' S? attribute+ S? '}') | ('{' S? label S? '}') | ('{' S? attribute+ '}') | (attribute+)

[2] attribute ::= '[' S? <attribute content> S? ']'
[3] <attribute content> ::= (label ':' S? featureContent) | featureContent | label
[4] featureContent ::= attributeName S? '=' S? attributeValueList
[5] attributeValueList ::= attributeValueElement (S? ',' S? attributeValueElement)*
[6] attributeValueElement ::= attributeValueName (S? avt)*
[7] attributeValueName ::= label (S label)*
[8] label ::= labelChar (label)*
[9] labelChar ::= '_' | '-' | '.' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
[10] S ::= (#x20 | #x9 | #xD | #xA)+

Here *S* is any sequences of space, new line, carriage return or line feed characters.

We can see that in the description of an AVT we can use some labels that define some sub trees: labels for attributes lists (rule **[1]**) and labels for attribute content (rule **[3]**). These labels can be used in others parts of the tree and in this manner the tree is represented more compact.

A more formal definition of an AVT is given in [9]. The AVTs have different useful properties like: paths in the AVT, EC (Exclusive Combinations) in the AVT, equivalence, well formed AVT, ordering, intersection, difference, union, factoring, normalization, unifiability, unification. Among these properties, unifiability and the unification are the most important. They are used in the generation process for generative dependency grammar with features (see section III.D).

C. *Generative Dependency Grammar*

A generative dependency grammar is an 8-tuple $GDG = \{N, T, P, A, SR, CR, nt_0, R\}$ where:

- *N, T, P, A, SR, CR* are defined like in section III.A.
- nt_0 - belongs to N and is named root symbol.
- *R* - is the set of numbered rules of the form *(i)* $n_i ::=(p_i, q_i)$, $n_i \in N$, p_i is a sequence of elements from N $\cup$ T $\cup$ P $\cup$ A, q_i is a dependency tree having nodes from p_i and oriented links (relations) from SR $\cup$ CR.

In a GDG we can make generation that will build in the same time surface texts and dependency trees.

We give in the following an example of a grammar that can generate a phrase like:

"*We provide practical advice for system and working group administrators.*"

(1) <phrase> ::= (<nominal group> <verbal group>, <nominal group>(r1 (<verbal group>())))
(2) <nominal group> ::= ("we", "we"())
(3) <verbal group> ::= (<verb> <complement> <complement'>, <verb>(r2(<complement>()), r3(<complement'>())))
(4) <complement> ::= (<attribute> <noun>, <noun>(r4(<attribute>())))
(5) <complement> ::= ("for" <coordination>, "for"(<coordination>()))
(6) <coordination> ::= (<member> "and" <member'>, r5(<member>(), <member'>() / r6("and"())))
(7) <member> ::= (<noun>, <noun>())
(8) <member> ::= (<attribute> <noun>, <noun>(r7(<attribute>())))
(9) <attribute> ::= (<attribute> <noun>,<noun>(r8(<attribute>())))
(10) <attribute> ::= (<noun>, <noun>())
(11) <attribute> ::= (<adjective>, <adjective>())
(12) <attribute> ::= ("practical", "practical"())
(13) <verb> ::= ("provide", "provide"())
(14) <noun> ::= ("advice", "advice"())
(15) <noun> ::= ("system", "system"())
(16) <noun> ::= ("administrator", "administrator"())
(17) <noun> ::= ("group", "group" ())
(18) <adjective> ::= ("working", "working"())

Using this grammar we can generate the surface text as follows:

(19)(1, 2) <phrase> ::= ("we" <verbal group>, "we"(r1(<verbal group>( ))))
(20)(19,3) <phrase> ::= ("we" <verb> <complement> <complement'>, "we"(r1(<verb> (r2(<complement>( )), r3 (<complement'>( )))))
(21)(20,13) <phrase> ::= ("we" "provide" <complement> <complement'>, "we"(r1("provide"(r2(<complement>( )), r3(<complement'>( )))))
(22)(21,4) <phrase> ::= ("we" "provide" <attribute> <noun> <complement'>, "we"(r1("provide"(r2(<noun>(r4(<attribute>( )))), r3(<complement'>( )))))
(23)(22,5) <phrase> ::= ("we" "provide" <attribute> <noun> "for" <coordination>, "we"(r1("provide"(r2(<noun>(r4(<attribute>( )))), r3("for"(<coordination>( ))))))
(24)(23,12) <phrase> ::= ("we" "provide" "practical <noun> "for" <coordination>, "we"(r1("provide"(r2(<noun>(r4("practical"( )))), r3("for"(<coordination> ( ))))))
(25)(24,14) <phrase> ::= ("we" "provide" "practical" "advice" "for" <coordination>, "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(<coordination> ( ))))))
(26)(25,6) <phrase> ::= ("we" "provide" "practical" "advice" "for" <member> "and" <member'>, "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5 (<member>( ), <member'>( ) / r6("and"( ))))))))
(27)(26,7) <phrase> ::= ("we" "provide" "practical" "advice" "for" <noun> "and" <member'>, "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5 (<noun>( ),

<member'>( ) / r6("and" ( ))))))))
(28)(27,15) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" <member'>, "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5 ("system"( ), <member'>( ) / r6("and"( ))))))))
(29)(28,8) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" <attribute> <noun>, "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5("system"( ), <noun>(r7(<attribute>( ))) / r6("and"( ))))))))
(30)(29,16) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" <attribute> "administrator", "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5("system"(), "administrator"(r7(<attribute>( ))) / r6("and"( ))))))))
(31)(29,9) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" <attribute> <noun> "administrator", "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5("system"( ), "administrator"(r7(<noun>(r8(<attribute>( ))))) / r6("and"( ))))))))
(32)(31,17) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" <attribute> "group" "administrator", "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5("system"( ), "administrator"(r7("group"(r8(<attribute>( ))))) / r6("and"( ))))))))
(33)(32,11) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" <adjective> "group" "administrator", "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5("system"( ), "administrator"(r7("group"(r8(<adjective>( ))))) / r6("and"( ))))))))
(34)(33,18) <phrase> ::= ("we" "provide" "practical" "advice" "for" "system" "and" "working" "group" "administrator", "we"(r1("provide"(r2("advice"(r4("practical"( )))), r3("for"(r5("system"( ), "administrator"(r7("group"(r8("working"( ))))) / r6("and"( ))))))))

The final production we obtained contains in the left side of the right side the surface text and in the right side of the right side the dependency tree (represented in Fig. 2).

The GDG allows obtaining a structure from an unstructured text. This structure can be used in different purposes, for example in translation process, in defining correspondences between two languages [7].

D. General Dependency Grammar with Features

A GDG with feature structure is a GDG where each *ntpa* can have associated an AVT. The AVT associated with the non-terminal from the left side of the rules have always only indexed attributes.

<u>*Example*</u>

Let us have the next phrase in Romanian language: "*Ploile* (the rains) *vǎratice* (of summer) *sunt* (are) *cǎlduțe* (lukewarm)" that means "*The summer rains are lukewarm*". We will not use all the grammatical categories involved in the analysis of this phrase but only few as an illustration.

Usually, the phrase to be analyzed is first of all annotated i.e. each word will have attached his lemma and a particular AVT (that have only one value for each attribute). Each word can have many interpretations. For example "*sunt*" can represent the third person plural (*are*) or the first person singular (*am*). Though, for the sake of simplicity, we will consider only one interpretation for each word.

The annotated phrase will be:

"Ploile" *ploaia* [class = noun] [gender = feminine] [number = plural] "vǎratice" *vǎratic* [class = adjective] [gender = feminine] [number = plural] "sunt" (a) *fi* [class = verb] [person: III] [number = plural] [mode = indicative] [voice = active] [time = present] "cǎlduțe" *cǎlduț* [class = adjective] [gender = feminine] [number = plural]

We marked the lemmas using italics.

A GDG with features that can generate this phrase can be as follows:

(1) <phrase> ::= (<nominal group> [gender = masculine, feminine, neuter] [number = singular, plural] [person = I, II, III] <compound nominal predicate> [gender = masculine, feminine, neuter] [number = singular, plural] [person = I, II, III], <nominal group>(@r_1@(<compound nominal predicate> ())))
(2) <nominal group> [gender = masculine, feminine, neuter] [number = singular, plural] [person = I, II, III] ::= (%noun% [class = noun] [gender = masculine, feminine, neuter] [number = singular, plural] %adjective% [class = adjective] [gender = masculine, feminine, neuter] [number = singular, plural], %noun%(@r_2@(%adjective% ())))
(3) <compound nominal predicate>[gender = masculine, feminine, neuter] [number = singular, plural] [person = I, II, III] ::= (%verb% [class = verb] [gender = masculine, feminine, neuter] [number = singular, plural] [mode = indicative] [voice = active] [time = present, future, imperfect past] %adjective% [class = adjective] [gender = masculine, feminine, neuter] [number = singular, plural], %verb%(@r_3@(%adjective% ())))

As we can see, we used pseudo terminals for nouns, verbs, adjectives, so this grammar can generate a set of phrases.

IV. Natural Language Syntax Description in GRAALAN

A. General Structure

The description of the syntax in GRAALAN [8] [10] will use GDG and AVT presented in section III. The language where we are describing the syntax must respect the following conditions:

a) Syntax: The description language will allow the description in a detailed and compact form of the manner to combine words in phrases respecting the rules of a natural language grammar.

b) Dependencies: We accept here that the dependency aspects are reduced to the mode different parts of a phrase are in relation one another (coordinate and governor/subordinate relations).

c) Agreement: By agreement [5] we will understand the mode different part of speech "match" one another when they are in certain dependency relations from the point of view of the values of different morphologic or syntactic categories.

d) Errors: The natural language syntax description must allow indicating the errors (at least the most frequent ones) that can be found in phrases. The bad built phrases must be recognized (in a certain measure) and marked as being incorrect.

e) Reversibility: By reversibility we will understand the property of the description language to be used to convert a source (surface) text in a deep structure (the dependency tree, in our case) and to convert the deep structure into the surface text.

We will give here an informal definition of natural language syntax description in GRAALAN. A more detailed definition of natural language syntax description in GRAALAN is given in the next sections.

A GRAALAN syntax description is a sequence of labeled rules. A rule has two parts: the left part and the right part. The left part of a rule contains a non terminal and an AVT. The AVT contains syntactic / morphologic categories with their values. The right part of a rule contains one or many Alternants. An alternant is formed by a set of subsections: the syntactic subsection, the dependency subsection and the agreement subsection.

a) The syntactic subsection is a sequence of (eventually labeled) one or many NTPAs. Each NTPA can have associated information about how this NTPA is linked with others NTPA by certain relations from the dependency subsection (these relations are indicated by their labels in the dependency subsection). There are three lists concerning the relations: coordinated list (CL), subordinated list (SL) and government list (GL).

Each NTPA can have associated an AVT describing syntactic / morphologic categories with their values.

b) The dependency subsection contains the description of the relations between the NTPA from the syntactic subsection (referred by their labels). There are two types of relations:

The subordinate relation SR is a relation between two N, T, P, A, or a coordinate relation CR. One of the two elements is considered to be the governor (that governs by SR) and the other the subordinate (that is governed by SR).

The coordinate relation CR is a relation between (usually) two N, T, P, A (that are said to be coordinated by CR), and eventually one or many SR (by which the CR is considered to be a governor for others N, T, P, A, or CR).

c) *The agreement subsection* contains a list of agreement rules. An agreement rule is a conditional expression expressed between the categories values of the NTPAs from the syntactic subsection. It can indicate some actions like error messages or how the analyze will be continued after an agreement error is found.

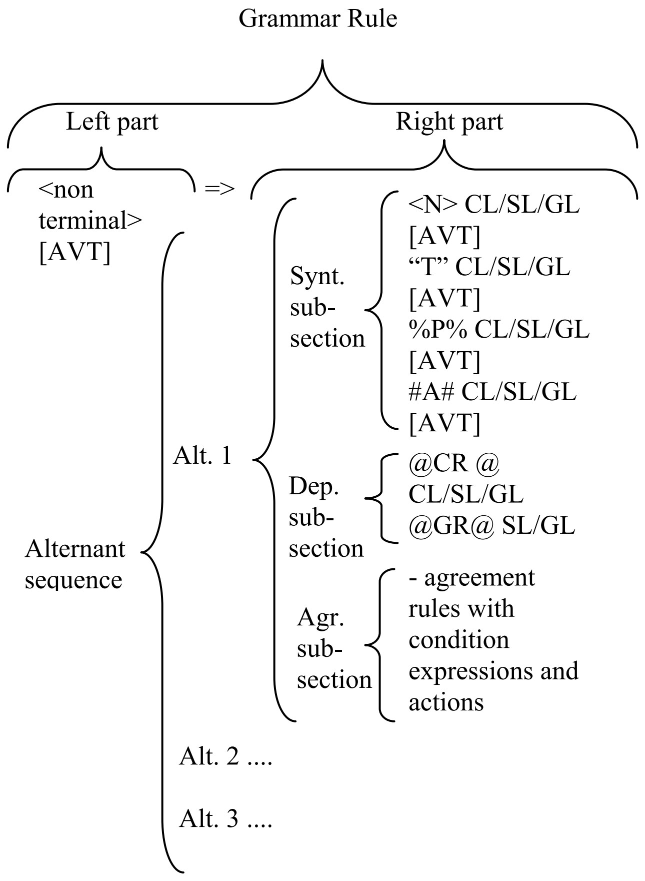


Fig. 3. Syntax rule elements

B. Graalan Syntactic Section Header

The syntactic section of a GRAALAN description is a sequence of rules preceded by a header. The description of the header in EBNF is the following:

[1] syntaxSection ::= 'Section' S 'syntax' S sectionHeader syntax S 'end' S 'of' S 'section'
[2] sectionHeader ::= (sourceLanguage, exploitationLanguage, sourceDirection, exploitationDirection)

A header is formed by elements that refer the source language and exploitation language.

[3] S ::= (#x20 | #x9 | #xD | #xA)+

Here *S* is any sequences of spaces, new line, carriage return or line feed characters.

[4] sourceLanguage ::= 'Source' S 'language' S language S
[5] exploitationLanguage ::= 'Exploitation' S 'language' S language S
[6] sourceDirection ::= 'Source' S 'direction' S ('left' | 'right')

[7] exploitationDirection ::= 'Exploitation' S 'direction' S ('left' |'right')
[8] language ::= ('RUM' | 'FRA' | 'FRA' | 'SPA' | 'RUS, ...')

We understand by *direction* the mode to scan the source text: *right* - the scan is done from left to right; *left* - the scan is done from right to left. The language is indicated according to [15].

[9] syntax ::= (rule S)+
[10] rule ::= 'Rule' S label ':' S '<' S? name S? '>' S? attribute* S? '::=' S? (('Alternant' S label ':' S? alternantContent)+ | ('Alternant' S alternantContent))

The alternant labels must be unique in the rule.

[11] name ::= label (S label)*
[12] label ::= labelChar (label)*
[13] labelChar ::= '_' | '-' | '.' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
[14] alternantContent ::= (syntacticSubsection ((dependencySubsection agreementSubsection?) | agreementSubsection)?)

An alternant can contain the three types of subsection (syntactic, dependency and agreement) in different combinations.

C. *Syntactic Subsection*

The syntactic subsection of a GRAALAN syntax rule contains information about a set of NTPAs that must be found in the analyzed / generated source text, in the corresponding sequence. The description of the syntactic subsection in EBNF is the following:

[15] syntacticSubsection ::= 'Syntax' S ((notRelationedNTPA S tpaRelationalList*)+) | ((notRelationedNTPA S nRelationalList*)+)
[16] notRelationedNTPA ::= (label ':' S?)? (('<' S? name S? '>') | ('"' terminal '"') | ('%' S? name S? '%') | ('#' S? label S? '#')) S attribute* (S ntpaBehaviour)*
[17] terminal ::= char (terminal)*
[18] char ::= &label; | &code;|...

Here *char* can be &label; or &code; or any character defined in GRAALAN Alphabet section (not described in this paper).

[19] code ::= '#x' hexaString
[20] hexaString ::= hexaChar (hexaString)*
[21] hexaChar ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F'
[22] ntpaBehaviour ::= ntpaBehaviourElement (S ntpaBehaviourElement)*
[23] ntpaBehaviourElement ::= '!' | (('OK' | 'KO') S? '=' S? ('OK' | 'KO')) | ('Message' S? = S? label) | ('Context' S? = S? '(' S? label (S label)* S? ')'))

Each element from *ntpaBehaviourElement* that characterises the NTPA behavior can appear only once in an *ntpaBehaviour*.

An NTPA that is head can have associated a feature named “cut” and indicated by “!”. This means that the head loses the characteristic of head. The corresponding NTPA will not have relations with other NTPAs. The grammar must be written in order to obtain a connected dependency tree. One alternant will have only one head. When we use another rule to substitute a non terminal in the current alternant, the different links of the substituted non-terminal will be applied to the head of the used rule.

The others elements that appear in *ntpaBehaviourElement* form the error sequence. In such a sequence the *Message* is mandatory. The error message itself is indicated by a *label* that defines the error text in another GRAALAN section (the GRAALAN Message Section that is not presented in this paper).

The condition ('OK' | 'KO') S? '=' S? ('OK' | 'KO') indicates the error triggering. The left of “=” indicates how NTPA treatment is terminated and the right side of “=” indicates the error condition, see Table II.

TABLE II
ERROR TRIGGERING FOR A NTPA

How NTPA treatment is terminated	Error condition	Continue as it was not an error
OK	*OK*	Yes (i)
OK	*KO*	No
KO	*KO*	Yes(ii)
KO	*OK*	No

The two cases when the treatment is continued as it were not an error has the following significance:

i) We found an error but this error is described as it was correct.

<u>*Example*</u>
(A simplified description.)

<imperative phrase>::=
    <vocative nominal group> "," <imperative verbal group> "!"|
    <vocative nominal group> <imperative verbal group >"!"

The first alternant corresponds to a phrase of the form (in Romanian):
"Domnilor, vorbiți mai încet !" (Gentlemen, shut up!)
(with a comma after the vocative).
The second alternant corresponds to a phrase of the form (in Romanian):
"Domnilor vorbiți mai încet!" (Gentlemen shut up!)
(without a comma after the vocative, therefore incorrect).
Though we have an error, the sense remains clear.

During the analysis of this incorrect phrase, the second alternant will return an OK value after the terminal "!". If we attach in the second alternant an error condition we will can indicated the error apparition:

```
Rule R1: <imperative phrase>::=
    Alternant A1:
        Syntax
                Label1:<vocative nominal group>
                        Coordinate Label5(1)
                Label2: ","!
                Label3: <imperative verbal group>
                        Coordinate Label5(2)
                Label4: "!"!
        Dependencies
                Label5: @vocative relation@(2)
    Alternant A2:
        Syntax
                Label1: <vocative nominal group>
                        Coordinate Label4(1)
                Label2: <imperative verbal group>
                                OK = OK Message =
                        ErrorMessage
                        Coordinate Label4(2)
                Label3: "!"!
        Dependencies
                Label4: @vocative relation@(2)
```

ii) There are situations when an NTPA must return OK but, if it returns KO, we have an error that must be identified as it is. Let us suppose a grammar fragment that must recognize phrases formed by a verbal group or by two verbal groups linked by "and".
(A simplified description.)

```
<phrase>::= <nominal group> <verbal group>"."|
    <nominal group><verbal group> "and"<verbal group>"."
```

In such a grammar, after an "and" was detected, a <verbal group> must be found. We can attach to the non terminal <verbal group> that is written after "and" an error sequence that must be triggered to KO return by this non terminal (giving an error message). A phrase like "*He came and.*" must produce this error message

The (simplified) description could be, for example:

```
Rule R1: <phrase>::=
    Alternant A1: <nominal group> <verbal group> "."
    Alternant A2: <nominal group> <verbal group> "and"
                <verbal group> KO = OK Message =
        Error101
                "."
```

A more detailed description:

```
Rule R1 <phrase>::=
    Alternant A1:
        Syntax
                Label1: <nominal group> Governor Label4
                Label2: <verbal group> Subordinate Label4
                Label3: "."!
        Dependencies
                Label4: @nominal / verbal relation@
    Alternant A2:
        Syntax
                Label1: <nominal group> Governor Label7
                Label2: <verbal group> Coordinate
        Label6(1)
                Label3: "and"!
                Label4: <verbal group>
                    KO = OK Message = Error101
                    Coordinate Label6(2)
                Label5 "."!
        Dependencies
                Label6: @"and" coordination@(2)
                        Subordinate Label7
                Label7: @nominal / verbal relation@
```

[24] tpaRelationalList ::= ('Governor' S labelList) | ('Subordinate' S labelList) | ('Coordinate' label S? '(' S? ('1' | '2' S?) ')')
[25] nRelationalList ::= ('Governor' S labelList) | ('Subordinate' S labelList) | ('Coordinate' label S? '(' S? ('1' | '2' S?) ')') | ('External' labelList)
[26] labelList ::= label (S label)*

A NTPA with relations is an NTPA that is linked with other NTPAs by relations (governor, subordinate, coordinate).

The *Subordinate* list of an NTPA can contain only the label of a *Subordinate* relation from the *Dependency* subsection.

The *Coordinate* list of an NTPA can contain only one label of a coordinate relation (that will be referred on a fixed entry) from *Dependency* subsection.

The *Governor* list of an NTPA can contain one or many labels of subordinate relations from the *Dependency* subsection.

For the same NTPA: the *Subordinate* list and *Governor* list can coexist, the *Governor* list and the *Coordinate* list can coexist, and the *Subordinate* list and the *Coordinate* list are exclusive.

The *External* list of can appear only for a nonterminal. It refers relations from *Dependency* subsection that appear wit the attribute *Definition* or *Reference* [6]. We must respect the discipline that, in the process of applying the rules (the generation of a new rule from two rules) always the external "definition" must appear before the corresponding external "reference". An external definition can appear only in a relational list of a non terminal, because only a non terminal can carry them further to someone that needs it.

D. Dependency Subsection

The dependency subsection of an alternant contains information about the relations that are defined between the NTPAs from the syntactic subsection of the alternant. The description of the dependency subsection in EBNF is the following:

[27] dependencySubsection ::= 'Dependencies' S (coordinateRelation | subordinateRelation)+
[28] coordinateRelation ::= label ':' S? ((('Reference' S) | ('Definition' ' S ')) ? '@' S? name S? '@' S? '(' S? '2' S? ')' (S? '!')? (('Subordinate' S label)? | ('Coordinate' S label S? '(' S? ('1' | '2') S? ')')?) ('Governor' (S label) | (S label S? '(' S? ('1' | '2') S? ')')+)?

A coordinate relation label must be unique among the alternant labels.

The *'Reference'* key word indicates (if present) that the current coordinate relation is not defined in the current rule but it is referred from another rule.

The *'Definition'* key word indicates (if present) that the current coordinate relation is defined in the current rule and it will be referred from other rules.

A coordinate relation can be followed by cut (“!”) because a coordinate relation can be head and using cut this feature will be discarded.

The *Subordinate* list of a coordinate relation can contain only one label of a subordinate relation from *Dependency* subsection.

The *Coordinate* list of a coordinate relation can contain only one label of a coordinate relation from *Dependency* subsection (referred on a fixed entry).

The *Governor* list of a coordinate relation can contain one or more labels of governor / subordinate relation from *Dependency* subsection. The current coordinate relation can be:

- governor for other NTPAs or coordinate relations (using a governor / subordinate relation indicated by a label not followed by a number in parenthesis); this means that the links from these governor / subordinate relations will come on the current coordinate relation on supplementary inputs);
- “governor” for other NTPAs or coordinate relations (using a label followed by a number of a fixed entry); this means that the links from these NTPAs or coordinate relations will appear on the corresponding fixed entry of the current coordinate relation.

For the same coordinate relation: the *Subordinate* list and the *Coordinate* list are exclusive and each of them can coexist with *Governor* list.

[29] subordinateRelation ::= label ':' S? ((('Reference' S) | ('Definition' ' S ')) ? '@' S? name S? '@' (('Subordinate' S label)? , ('Governor' S label)?)

A subordinate relation label must be unique among the alternant labels.

A subordinate relation has always one entry; so, we do not need to specify the number of entries. In fact, the presence of the entry number in coordinate relation indicates the fact that it is a coordinate relation.

The 'Reference' and 'Definition' key words have the same significance as for coordinateRelation.

The *Subordinate* list of a subordinate relation can contain only one label of a coordinate relation from the *Dependency* subsection (where it will go on a supplementary input) or of an NTPA from the *Syntax* subsection.

The *Governor* list of a subordinate relation contains only one label of an NTPA from the *Syntax* subsection or of a coordinate relation from the *Dependency* subsection.

Observation 1: Because a link has two extremities, it can be indicated by the any of its ends or by both. There are many ways to indicate a link according to its type. It is advisable to make the grammar description in such a way that a link appear only once (to the element from where the link leaves or to the element where the link arrives). See TABLE III where we use the notations:

GR = Governor/Subordinate Relation
CR = Coordinate relation
SL = Subordinate List
GL = Governor List
CL = Coordinate List

TABLE III
INDICATION OF LINKS

Link type	Link source (label A)	Link target (label B)	How the link is indicated	
			1	2
1	NTPA	GR	B in SL of A	A in GL of B.
2	NTPA	CR (on fixed entry)	B in CL of A (with fixed entry number of B)	A in GL of B (with fixed entry of B)
3	GR	NTPA	B in SL of A	A in GL of B
4	GR	CR (on supp. entry)	B in SL of A	A in GL of B
5	CR	GR	B in SL of A	A in GL of B
6	CR	CR (on fixed entry	B in CL of A (with fixed entry number of B)	A in GL of B (with fixed entry number of B)

Observation 2: In the Table IV it is indicated that we can put in different relational lists function of the element that the list belongs to.

Example

```
… ::= …<non terminal 1>
                    {Sequence:
                      (gender = masculine, feminine, neutral)
                      (number = singular, plural)}
          <non terminal 2> {Sequence}
          <non terminal 3> {Sequence}
```

TABLE IV
THE RELATIONAL LIST CONTENT

List type	List owner	List content
GL	NTPA	The labels of one or many governor / subordinate relations that have outputs going to the current NTPA.
	GR	The label of a coordinate relation or of an NTPA that have outputs going on the input of the current governor / subordinate relation.
	CR	The labels of one or many governor / subordinate relation (that have outputs going to the supplementary input of the current coordinate relation) and / or labels of some NTPAs or other coordinate relation that have outputs going on fixed entries of the current coordinate relation. In this case, the corresponding number of fixed entry is indicated too.
SL	NTPA	The label of only one governor / subordinate relation that have an input where the output of the current NTPA goes.
	GR	The label of a coordinate relation (where the output of the current governor / subordinate relation will go on a supplementary input) or of an NTPA (that have an entry where the output of the current governor / subordinate relation will go).
	CR	The label of a governor / subordinate relation that has an input where the output of the current coordinate relation will go.
CL	NTPA	The label of only one coordinating relation (where the output of the current NTPA will go on a fixed entry). In this case the number of the fixed entry is also indicated.
	CR	The label of only one coordinating relation (where the output of the current coordinate relation will go on a fixed entry). In this case the number of the fixed entry is also indicated.

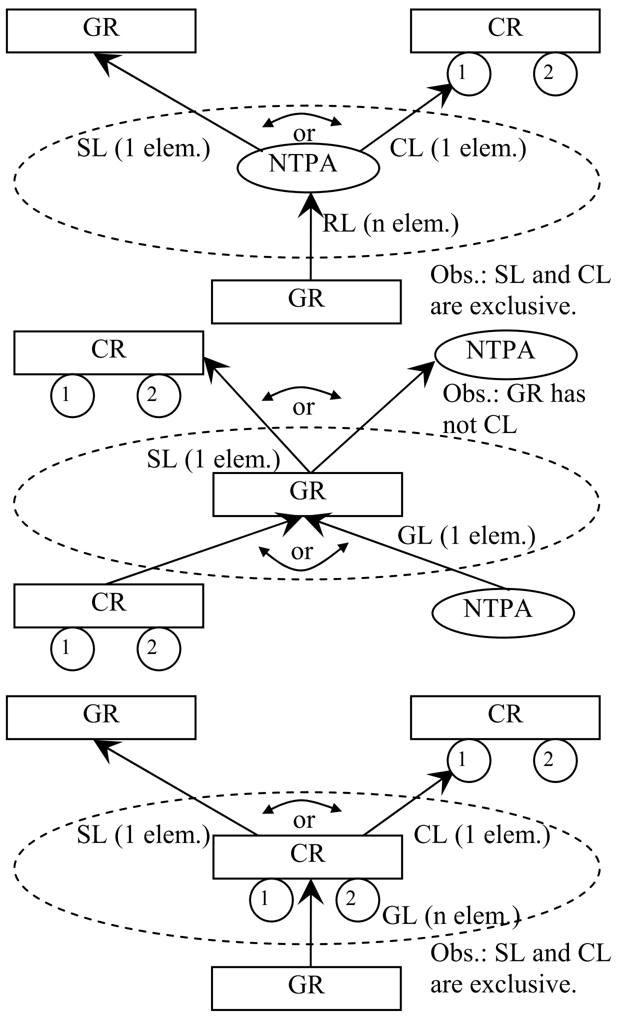


Fig. 4. The content of *Governor*, *Subordinate* and *Coordinate* lists (GL, SL, CL)

[30] attribute ::= notIndexedAttribute | indexedAttribute | '{' S? label ':' S? attribute+ S? '}' | '{' S? attribute+ S? '}' | '{' S? label S? '}'

In this representation, the *label* of an *attribute* sequence allows to compact the rule. If the same attribute sequence appears many times in a rule (in the left side or in the alternants from the right side), then the first apparition of the sequence can be labeled and the following apparitions can be indicated only by using this label. A label of an attribute sequence must be unique in the current rule.

[31] notIndexedAttribute ::= '(' (((S? label ':' S?)? attributeContent) | label) ')'
[32] indexedAttribute ::= '[' (((S? label ':' S?)? attributeContent) | label) ']'
[33] attributeContent ::= category S? '=' S? categoryValue (S? ',' S? categoryValue)*

In this representation, the *label* of an *attributeContent* allows to compact the rule. If the same attribute appears many times in a rule (in the left side or in the alternants from the right side), then its first apparition can be labeled and the following apparitions can be indicated only by using this label. A label of an attribute must be unique in the current rule.

Example

```
… ::= …<non terminal 1>
            (Gender1: gender = masculine, feminine,
    neutral)
            (Number1: number = singular, plural)
    <non terminal 2>
            (Gender2: gender = masculine, feminine)
            (Number2: number = singular)
    <non terminal 3>
            (Gender1) (Number1)
    <non terminal 4>
            (Gender2) (Number2)
```

If an *indexedAttribute* contains a *label* then this label will play the role of an index. If the *indexedAttribute* do not have a *label*, then the *category* from *attributeContent* will play the index role. In any cases, *categoryValue* from *attributeContent* represent all the values that the index can take.

Example

Let us have a set of rules of the form:

```
<complex subjective group>
    [person = I, II, III]
    [number = sg, pl]
    [gender = m, f, n]
    ::=
    Alternant A1:
        <unitary subjective group>
        [person = I, II, III]
        [number = sg, pl]
        [gender = m, f, n]
    Alternant A2:
        <logical subjective group>
        [person = I, II, III]
        [number = sg, pl]
        [gender = m, f, n]
    Alternant A3:
        <distributive subjective group>
        [person = I, II, III]
        [number = sg, pl]
        [gender = m, f, n]
    Alternant A4:
        <correlative subjective group>
        [person = I, II, III]
        [number = sg, pl]
        [gender = m, f, n
```

Considering the combinations for person, number and gender, this rule represents in fact 18 rules.

Example

The same thing can be written more compact as follows:

```
<complex subjective group>
    [e1: persoana = I, II, III]
    [e2: number = sg, pl]
    [e3: gen = m, f, n]
    ::=
    Alternant A1:
        <unitary subjective group> [e1][e2][e3]
    Alternant A2:
        <logical subjective group>[e1][e2][e3]
    Alternant A3:
        <distributive subjective group>[e1][e2][e3]
    Alternant A4:
        <correlative subjective group>[e1][e2][e3]
```

Example

A more important using of the indexing is when the same category must serve as index in many ways in the same alternant.

```
<non terminal1>
    [e1: attribute1 = value11, value12, value13]
    [e2: attribute2 = value21, value22]
    [e3: attribute3 = value31, value32, value33]
    ::=
    Alternant A1:
        <non terminal2>[e1][e2][e3]
        <non terminal3>
        [e4: attribute1 = value11, value12, value13]
        [e5: attribute2 = value21, value22, value23]
        [e6: attribute3 = value31, value32, value33]
        <non terminal4>[e4][e5][e6]
```

In this example, the attributes *e1: attribute1*, *e2: attribute2*, *e3: attribute3* are considered as indexes different from *e4: attribute1*, *e5: attribute2*, *e6: attribute3* (i.e., for example, *<non terminal1>* and *<non terminal2>* must have the same value for *attribute1*, *<non terminal3>* and *<non terminal4>* must have the same value for *attribute1* but the values for *attribute1* can be different in *<non terminal1>* and in *<non terminal3>*, etc.)

[34] categoryValue ::= name S attribute*

We can see that a *categoryValue* can be followed by an *attribute* sequence. In this way, a branching in an attribute value tree is represented. A sequence of "category = value" that pass by such branching points represents a path in the AVT. The syntax must be written in such a way that a path do not have many apparition of the same category.

E. Agreement Subsection

The agreement subsection of an alternant describes the conditions that must be respected by the morphologic / syntactic categories of the NTPA from the syntactic subsection. The description of the agreement subsection in EBNF is the following:

[35] agreementSubsection ::= 'Agreement' agreementRule+
[36] agreementRule ::= 'if' S? '(' S? conditionExpression S? ')' S? alternatives+ ((S? 'else' S? '(' S? agreementRule S? ')'

S?) | (S? 'else' S? agreementRule S?) | (S? 'else' S? '(' S? actionList S? ')'))?
[37] alternatives ::= ('true' S? '(' S? expression S? ')') | ('false' S? '(' S? expression S? ')') | ('not' S? 'applicable' S? '(' S? expression S? ')') | ('not' S? 'determinated' 'S? (' S? expression ')')
[38] expression ::= actionList | agreementRule

A *conditionExpression* can have one of the four truth values. We will use a tetravalent logic that has the following truth values: *TRUE*, *FALSE*, *NOT APPLICABLE*, *NOT DETERMINATE*. Therefore, after *'if' S? '(' S? conditionExpression S? ')' S?* we must have a list of maximum four alternatives and these alternatives must be different. If some alternatives are missing, they can globally be treated using *else*.

Example

Let us have the following sequence:

if (conditionExpression)
true(expression 1)
not applicable(expression 2)
else(expression 3)

Such an expression is read: "if *conditionExpression* is true then execute *expression1* and if *conditionExpression* is *not applicable* then execute *expression2* otherwise (i.e. *conditionExpression* is *false* or *not determinated*) then execute *expression3*".

[39] conditionExpression ::= ('(' S? conditionExpression S? ')' S? logicalOperator S? conditionExpression) | (S? '~' S? '(' S? conditionExpression S? ')' S? logicalOperator S? conditionExpression) | ('(' S? conditionExpression S? ')') | ('~' S? '(' S? conditionExpression S? ')') | (simpleExpression S? logicalOperator S? simpleExpression) | simpleExpression

In order to formulate the logical value of the *conditionExpression* we can use *logicalOperators* (including the negation "~"), parenthesis and operands that are *simpleExpression*.

[40] logicalOperator ::= 'and' | 'or'
[41] simpleExpression ::= ({operand} S? '+ S? {operand} S? '<-' S? {operand}) | ({operand} S? '<-' S? {operand}) | ({operand} s? '->' S? {operand}S? '+' S? {operand}) | ({operand} S? '->' S? {operand})
[42] operand ::= label attribute+

An operand indicates an NTPA that is involved in the agreement. The agreement is usually expressed between a governor and a subordinate. Let us have the example: *"Not only the rain but also the wind corrode the cliffs."* Between "*Not only the rains but also the winds"* as multiple subject and *"corrode"* must be described an agreement. (We can also describe a *sort of* agreement also between *"not only"* and *"but also"* as two parts of a correlation.)

If we have a governor / subordinate relation, then the operator representing the governor will be at left of *"<-"* or at right of *"->"* (the arrow looks at the governor).

An expression of the form *operand*$_1$ + *operand*$_2$ <- *operand*$_3$ or *operand*$_3$ ->*operand*$_1$ + *operand*$_2$ is read: "if *operand*$_1$ has some features a$_1$ (attributes and values: a certain gender, a certain number, etc.) and *operand*$_2$ has certain features a$_2$ then the *operand*$_3$ must have certain features a$_3$.

An expression *operand*$_1$ <- *operand*$_2$ or *operand*$_2$ -> *operand*$_1$ is read: "if *operand*$_1$ has certain features a$_1$ then *operand*$_2$ must have certain features a$_2$".

The *operand* contains a *label* of an NTPA (from the syntactic subsection of the current alternant) involved in the agreement and an *attribute* under the form of an AVT.

An AVT (indicated by *attribute*) of an operand must be unifiable with the AVT associated to the corresponding NTPA (indicated by *label*).

A *simpleExpression* is TRUE when all its operands have AVTs unifiable with the corresponding AVT from the syntactic subsection.

A *simpleExpression* is FALSE when the AVT corresponding to the operands represented the governor is unifiable with the corresponding AVT of NTPA from syntactic subsection and those representing the subordinate are not.

A *simpleExpression* is NOT APPLICABLE if at least one of the operands representing the governor has a not unifiable AVT.

(A value of NOT DETERMINED can appear only by the evaluation of the *conditionExpression* containing simple expressions.)

Example

Let us have two non terminals that appear in syntactic subsection of an alternant:

Label1: <non terminal1>	(a = av1, av2)
	(b = bv1, bv2, bv3)
	(c = cv1, cv2, cv3)
Label2: <non terminal2>	(d = dv1, dv2)
	(e = ev1, ev2, ev3)
	(f = vf1, vf2, vf3)

Let us have the simple expression of the form:

Label1(a = av1, av2)(b = bv2, bv3) -> Label2(e =ev1, ev2)(f = vf2)

During the syntactic analysis, after the current alternant was analyzed and recognized in source text, <non terminal1> and <non terminal2> will have only some of the above attributes/values.

The operand Label1(a = av1, av2)(b = bv2, bv3) will be unifiable with <non terminal1> when this one will have after the syntactic analysis:

- the category a with the values av1 or av2;
- the category b with the values bv1 or bv3.

The operand Label2(e =ev1, ev2)(f = vf2) will be unifiable with the <non terminal2> when this one will have after the syntactic analysis:

- the category e with the values ev1 or ev2;
- the category f with the value vf2.

Example

Let us have two non terminals that appear in syntactic subsection of an alternant:

```
Label1:   <elementary nominal group>
          (negation = affirmative, negative)
          (person = I, II)
          (number = singular)
          (gender = masculine, feminine)
Label2:   <verbal group>
          (negation = affirmative, negative)
          (person = I, II)
          (number = singular)
          (gender = masculine, feminine)
```

Let us have the expression of the form:

```
Label1(person = I) <- Label2(person = I)
or
Label1(persoana = II) <- Label2(person = II)
```

This expression will be *TRUE* when the non terminal with the labels Label1 and Label2 will have the same person (I or II).

Using the indexed representation of the attributes, the expression can be written more compact:

```
Label1[person = I, II] <- Label2[person = I, II]
```

[43] actionList ::= action (S? ',' S? actionList)*
[44] action ::= ('Message' S? '=' S? label) | ('OK' | 'KO') |
('Context' S? = S? '(' S? label (S label)* S? ')')

The *Message* is an error message indicated by a label in another GRAALAN section not described in this paper (where messages in different languages can be found). This message will be displayed during the syntactic analysis of a source text.

The mode OK | KO indicated how the current NTPA situation must be treated:

- KO: negative;
- OK: positive.

Observation 3: An agreement rule between different NTPAs of an alternant make sense only if all these NTPAs have associated attributes with many values in the syntactic subsection of the alternant. If the NTPAs have not attributes or they have attributes but all the attributes have only one value then the agreement problem is solved by the syntactic description itself and we do not need an agreement rule.

The message *Context* is represented by the labels of certain NTPA or relations that can be used in debugging process.

Example

Let us take an agreement expression of the form:

```
if (Label1(person = I) -> Label2(person = I))
    true ( OK )
    else ( Message = Label3, OK)
```

It will be read: “If the NTPA with the label Label1 from the syntactic subsection has the person I and the NTPA with the label Label2 from syntactic subsection has the person I then continue the syntactic analysis, otherwise display the error message with the label Label3 (and that can be for example defined in Message section of GRAALAN language like: "Person number agreement error") and continue after that the syntactic analysis as it was not an error.”

V. Conclusion

We presented a method to describe the syntax of a natural language. The description mode is part of a more general language GRAALAN that allows the description of a natural language or the correspondences between two natural languages. We consider that this kind of description is quite general and fit for almost any natural language.

In order to use the description method, some tools are needed [11]. Among these tools, a very important one is GRAALAN compiler. This compiler analyzes the description GRAALAN text and converts it to XML. The XML information will form an LKB (Linguistic Knowledge Base). The knowledge form LKB can afterwards be used to build different linguistic applications: morphologic analyzer, grammar checker, inflection application, indexing / searching application, lemmatizer, speller, hyphenating application, different kinds of lexicons and dictionaries, different kinds of machine translation applications (human assisted machine translation, computer assisted machine translation, automatic machine translation), etc.

Some tools for GRAALAN are already developed (GRAALAN Macro processor, GRAALAN Compiler, Inflection Forms Tool that allows an automatic / interactive generation of the inflected forms). Some tools are currently in design / implementation stage (Lexicon Tool that allows an automatic / interactive lexicon creation, LINK that checks the coherence of an entire LKB). Some Romanian linguistic knowledge bases are already defined (Alphabet Section, Morphologic Configurator Section, Syllabification Section, Inflection Rules Section), some are partially developed (Lexicon Section, Syntax Section), some will be soon developed (Inflection Forms Section).

We hope that the system built on GRAALAN will be an important tool to elaborate some unified and very general linguistic applications.

References

[1] H. Alshawi, D. J. Arnold, R. Backofen, D. M. Carter, J. Lindop, K. Netter, S. G. Pulman, J. Tsujii, H. Uszkoreit. EurotraET6/1: Rule

Formalism and Virtual Machine Study. Final Report. Commission of the European Communities, 1991.

[2] R. Backofen et al. EAGLES Formalism Working Group Final Report. Expert Advisory Group on Language Engineering Standards, 1996.

[3] S. Diaconescu. Natural Language Understanding Using Generative Dependency Grammar. In: M. Bramer, A. Preece and F. Coenen (Eds), *Twenty second SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence*, pp.439-452, Cambridge UK, Springer, 2002.

[4] S. Diaconescu. Morphological Categorization Attribute Value Trees and XML. In: M. A. Klopotek, S. T. Wierzchon, K. Trojanowski (Eds), *Intelligent Information Processing and Web Mining*, Proceedings of the International IIS: IIPWM'03 Conference, pp. 131-138, Zakopane, Poland, Springer, 2003.

[5] S. Diaconescu. Natural Language Agreement Description for Reversible Grammars. In: T. D. Gedeon, L. C. C. Fung (Eds.), *Advances in Artificial Intelligence, 16th Australian Conference on AI*, pp. 161-172, Perth, Australia, Springer, 2003.

[6] S. Diaconescu. Natural Language Processing Using Generative Indirect Dependency Grammar. In: M. A. Klopotek, S. T. Wierzchon, K. Trojanowski (Eds), *Intelligent Information Processing and Web Mining*, Proceedings of the International IIS, IIPWM'04 Conference, pp. 414-418, Zakopane, Poland, Springer, 2004.

[7] S. Diaconescu. Multiword Expression Translation Using Generative Dependency Grammar. In: J. L. Vicedo, P. Martinez-Barco, R. Muñoz, M. S. Noeda (Eds.), *Advances in Natural Language Processing*, Proceedings of 4th International Conference, ESTAL 2004, pp. 243-254, Alicante, Spain, Springer, 2004.

[8] S. Diaconescu. GRAALAN – Grammar Abstract Language Basics. In: J. M. Jun, B. M. Bae, K. Y. Lee (Eds) *GESTS International Transaction on Computer Science and Engineering*, Vol.10, No.1: Sunjin Publishing Co., 2005.

[9] S. Diaconescu. Some Properties of the Attribute Value Trees Used for Linguistic Knowledge Representation. In: *2nd Indian International Conference on Artificial Intelligence (IICAI-05)*, Pune, INDIA, 2005.

[10] S. Diaconescu. Creation of the linguistic resources using a specialised language. (Crearea resurselor lingvistice cu ajutorul unui limbaj specializat), In C. Forăscu, D. Tufiş, D. Cristea (Eds.), *Workshop on Linguistic resources and tools for Romanian Language Processing*, pp. 39-44, Iassi, Romania, Editura Universităţii A. I. Cuza, 2006.

[11] S. Diaconescu. Complex Natural Language Processing System Architecture. In: Corneliu Burileanu, Horia-Nicolai Teodorescu (Eds.), *Advances in Spoken Language Technology*, pp. 228-240, Bucharest, Romania: The Publishing House of the Romanian Academy, 2007.

[12] EAGLES Formalism Working Group Final Report, Version of september 1996.

[13] IPA International Phonetic Association. Handbook of the International Phonetic Association, A Guide to the Use of the International Phonetic Alphabet. Cambridge, UK: Cambridge University Press, 2005.

[14] ISO/IEC 10646. Information technology -- Universal Multiple-Octet Coded Character Set (UCS). Geneva, International Organization for Standardization, 1992.

[15] ISO 639 (E). Code for the representation of names of languages. Geneva, International Organization for Standardization, 1998.

[16] A. Joshi, L. Levi, L. Takabashi. Tree Adjunct Grammars. *Journal of the Computer and System Sciences*, 1975.

[17] C. Pollard, I. Sag. Head-Driven Phrase Structure Grammar. Stanford: CSLI & Chicago: U Chicago Press, 1994.

[18] S. Kahane. Grammaire d'Unification Sens-Texte. Vers un modèle mathématique articulé de la langue. Document de synthèse, Paris, France: Univ. Paris 7, 2002.

[19] R. Kaplan, J. Bresnan. Lexical Functional Grammar. A Formal System for Grammatical Representation. In: J. Bresnan (ed), *The Mental Representation of Grammatical Relations*: Massachusetts USA: MIT Press, 1982

[20] J. Landsbergen. Isomorphic grammars and their use in the ROSETTA translation system. In: *Machine Translation Today: The State of the Art*, Edinburgh UK: Edinburgh University Press, 1987.

[21] L. Tesnière. Éléments de syntaxe structurelle, Paris France: Klincksieck, 1959.

[22] W3C. Extensible Markup Language (XML) 1.0, Recommendation. 10-Feb-98, pp. 24-25, 1998.

Morpheme based Language Model for Tamil Part-of-Speech Tagging

S. Lakshmana Pandian and T. V. Geetha

Abstract—**The paper describes a Tamil Part of Speech (POS) tagging using a corpus-based approach by formulating a Language Model using morpheme components of words. Rule based tagging, Markov model taggers, Hidden Markov Model taggers and transformation-based learning tagger are some of the methods available for part of speech tagging. In this paper, we present a language model based on the information of the stem type, last morpheme, and previous to the last morpheme part of the word for categorizing its part of speech. For estimating the contribution factors of the model, we follow generalized iterative scaling technique. Presented model has the overall F-measure of 96%.**

Index Terms—**Bayesian learning, language model, morpheme components, generalized iterative scaling.**

I. Introduction

Part-of-speech tagging, i.e., the process of assigning the part-of-speech label to words in a given text, is an important aspect of natural language processing. The first task of any POS tagging process is to choose various POS tags. A tag set is normally chosen based on the language technology application for which the POS tags are used. In this work, we have chosen a tag set of 35 categories for Tamil, keeping in mind applications like named entity recognition and question and answering systems. The major complexity in the POS tagging task is choosing the tag for the word by resolving ambiguity in cases where a word can occur with different POS tags in different contexts. Rule based approach, statistical approach and hybrid approaches combining both rule based and statistical based have been used for POS tagging. In this work, we have used a statistical language model for assigning part of speech tags. We have exploited the role of morphological context in choosing POS tags. The paper is organized as follows. Section 2 gives an overview of existing POS tagging approaches. The language characteristics used for POS tagging and list of POS tags used are described in section 3. Section 4 describes the effect of morphological context on tagging, while section 5 describes the design of the language model, and the section 6 contains evaluation and results.

II. Related Work

The earliest tagger used a rule based approach for assigning tags on the basis of word patterns and on the basis of tag assigned to the preceding and following words [8], [9]. Brill tagger used for English is a rule-based tagger, which uses hand written rules to distinguish tag entities [16]. It generates lexical rules automatically from input tokens that are annotated by the most likely tags and then employs these rules for identifying the tags of unknown words. Hidden Markov Model taggers [15] have been widely used for English POS tagging, where the main emphasis is on maximizing the product of word likelihood and tag sequence probability. In essence, these taggers exploit the fixed word order property of English to find the tag probabilities. TnT [14] is a stochastic Hidden Markov Model tagger, which estimates the lexical probability for unknown words based on its suffixes in comparison to suffixes of words in the training corpus. TnT has developed suffix based language models for German and English. However most Hidden Markov Models are based on sequence of words and are better suited for languages which have relatively fixed word order.

POS tagger for relatively free word order Indian languages needs more than word based POS sequences. POS tagger for Hindi, a partially free word order language, has been designed based on Hidden Markov Model framework proposed by Scutt and Brants [14]. This tagger chooses the best tag for a given word sequence. However, the language specific features and context has not been considered to tackle the partial free word order characteristics of Hindi. In the work of Aniket Dalal *et al.* [1], Hindi part of speech tagger using Maximum entropy Model has been described. In this system, the main POS tagging feature used are word based context, one level suffix and dictionary-based features. A word based hybrid model [7] for POS tagging has been used for Bengali, where the Hidden Markov Model probabilities of words are updated using both tagged as well as untagged corpus. In the case of untagged corpus the Expectation Maximization algorithm has been used to update the probabilities.

III. Parts of Speech in Tamil

Tamil is a morphologically rich language resulting in its relatively free word order characteristics. Normally most Tamil words take on more than one morphological suffix; often the number of suffixes is 3 with the maximum going up to 13. The role of the sequence of the morphological suffixes attached to a word in determining the part-of-speech tag is an interesting property of Tamil language. In this work, we have identified 79 morpheme components, which can be combined to form about 2,000 possible combinations of integrated suffixes. Two basic parts of speech, namely, noun and verb, are mutually distinguished by their grammatical inflections. In

Manuscript received May 12, 2008. Manuscript accepted for publication October 25, 2008.

S. Lakshmana Pandian and T. V. Geetha are with Department of Computer Science and Engineering, Anna University, Chennai, India (lpandian72@yahoo.com).

Tamil, noun grammatically marks number and cases. Tamil nouns basically take on eight cases. The normal morphological derivatives of Tamil nouns are as follows:

$Stem_{Noun}$ *+ [Plural Marker]+[Oblique]+[Case Marker]*

The normal morphological derivative of Tamil Verb is as follows

$Stem_{Verb}$*+[Tense Marker]+[Verbal Participle Suffix]+[Auxiliary verb +[Tense Marker]+[Person, Number, Gender]*

In addition, adjective, adverb, pronoun, postposition are also some stems that take suffixes. In this work, we have used a tagged corpus of 4,70,910 words, which have been tagged with 35 POS categories in a semiautomatic manner using an available morphological analyzer [5], which separates stem and all morpheme components. It also provides the type of stem using lexicon. This tagged corpus is the basis of our language model.

IV. Effect of Morphological Information in Tagging

The relatively free word order of Tamil normally has the main verb in a terminating position and all other categories of words can occur in any position in the sentence. Pure word based approaches for POS tagging are not effective. As described in the previous section, Tamil has a rich multilevel morphology. This work exploits this multi-level morphology in determining the POS category of a word. The stem, the pre-final and final morpheme components attached to the word that is the words derivative form normally contribute to choosing the POS category of the word. Certain sequence of morpheme can be attached to the certain stem types. Moreover, the context in which morphological components occur and its combinations also contribute in choosing the POS tag for a word. In this work, language models have been used to determine the probabilities of a word derivative form functioning as a particular POS category.

TABLE I.
List of tags for Tamil and their description

	Tag	Description
1.	N	Noun
2.	NP	Noun Phrase
3.	NN	Noun + noun
4.	NNP	Noun + Noun Phrase
5.	IN	Interrogative noun
6.	INP	Interrogative noun phrase
7.	PN	Pronominal Noun
8.	PNP	Pronominal noun
9.	VN	Verbal Noun
10	VNP	Verbal Noun Phrase
11	Pn	Pronoun
12	PnP	Pronoun Phrase
13	Nn	Nominal noun
14	NnP	Nominal noun Phrase
15	V	Verb
16	VP	Verbal phrase
17	Vinf	Verb Infinitive
18	Vvp	Verb verbal participle
19	Vrp	Verbal Relative participle
20	AV	Auxiliary verb
21	FV	Finite Verb
22	NFV	Negative Finite Verb
23	Adv	Adverb
24	SP	Sub-ordinate clause conjunction Phrase
25	SCC	Sub-ordinate clause conjunction
26	Par	Particle
27	Adj	Adjective
28	Iadj	Interrogative adjective
29	Dadj	Demonstrative adjective
30	Inter	Intersection
31	Int	Intensifier
32	CNum	Character number
33	Num	Number
34	DT	Date time
35	PO	Post position

V. Language Model

Language models, in general, predict the occurrence of a unit based on the statistical information of unit's category and the context in which the unit occurs. Language models can differ in the basic units used for building the model, the features attached to the units and the length of the context used for prediction. The units used for building language models depend on the application for which the model is used. Bayes' theorem is usually used for posterior probability estimation from statistical information. Bayes' theorem relates the conditional and marginal probabilities of stochastic events *A* and *B* as follows

$$\Pr(A/B) = \frac{\Pr(B/A)\Pr(A)}{\Pr(B)} \quad \alpha \quad L(A/B)\Pr(A) \tag{5.1}$$

where *L(A|B)* is the likelihood of *A* given fixed *B*.

Each term in Bayes' theorem is described as follows.

Pr(A) is the *prior probability* or *marginal probability* of *A*. It is "prior" in the sense that it does not take into account any information about *B*.

Pr(B) is the prior or marginal probability of *B*, and acts as a *normalizing constant.*

Pr(A|B) is the *conditional probability* of *A*, given *B*. It is also called the posterior probability because it is derived from or depends upon the specified value of *B*.

Pr(B|A) is the conditional probability of *B* given *A*.

In this paper, we have used a language model that considers the lexical category of the stem along with morphological components of a word in order to determine its POS tag.

In case the word is equal to a stem then its POS category is the same as the stem lexical category. In case the word consists of a stem and a single morphological component then the language model is designed by considering both the lexical category of the stem and the morphological component. It is given by the equation 5.2:

$$P\left(\frac{pos}{root\_type, e_l}\right) = \alpha_1 p\left(\frac{pos}{root\_type}\right) + \alpha_2 p\left(\frac{pos}{e_l}\right) \quad (5.2)$$

where $P\left(\frac{pos}{root\_type, e_l}\right)$ gives the probability of the word being tagged with the particular pos tag given a particular stem type and a particular morphological ending e_l. The two factors used for this probability calculation are $p\left(\frac{pos}{root\_type}\right)$, the prior probability of a particular pos tag given a particular stem type and $p\left(\frac{pos}{e_l}\right)$, the prior probability of a particular pos tag given a particular morphological ending e_l. In addition, α_1 and α_2 are contribution factors and $\alpha_1 + \alpha_2 = 1$.

In order to calculate the prior probability $p\left(\frac{pos}{e_l}\right)$ we use equation 5.3:

$$P\left(\frac{pos}{e_l}\right) = \frac{P(pos)\, p\left(\frac{e_l}{pos}\right)}{P(e_l)} \quad (5.3)$$

In equation 5.3 $P(pos)$ is the posterior independent probability of the particular pos in the given corpus.

$p\left(\frac{e_l}{pos}\right)$ is the posterior probability of the final morphological component given the particular pos tag calculated from the tagged corpus. $P(e_l)$ is the posterior independent probability of final morphological component calculated from the tagged corpus. This probability is calculated using equation 5.4:

$$P(e_l) = \sum_{i=1}^{k} p(pos_i)\, p\left(\frac{e_l}{pos_i}\right) \quad (5.4)$$

$P(e_l)$ is calculated as a summation of all possible pos type k . $P(e_l)$, the product of $p(pos_i)$, the posterior probability of the given pos type *i* in the corpus and $p\left(\frac{e_l}{pos_i}\right)$, the posterior probability of the final morpheme component given the pos tag *i*.

In case the word whose pos tag is to be determined consists of more than one morphological component, then the language model is designed by considering three factors: the lexical category of the stem and the pre-final and final morphological component and is given by equation 5.5:

$$P\left(\frac{pos}{root\_type, e_{l-1}, e_l}\right) = \alpha_1 p\left(\frac{pos}{root\_type}\right) + \alpha_2 p\left(\frac{pos}{e_{l-1}}\right) + \alpha_3 p\left(\frac{pos}{e_l}\right) \quad (5.5)$$

where $P\left(\frac{pos}{root\_type, e_{l-1}, e_l}\right)$ gives the probability of the word being tagged with the particular pos tag given a particular stem type and a particular morphological pre-final and final components e_{l-1} and e_l. The three factors used for this probability calculation are $p\left(\frac{pos}{root\_type}\right)$, the prior probability of a particular pos tag given a particular stem type and $p\left(\frac{pos}{e_l}\right)$, the prior probability of a particular pos tag given a particular morphological component e_l. These two factors are similar to the equations 5.1 and 5.2. The second factor $p\left(\frac{pos}{e_{l-1}}\right)$ is the prior probability of a particular pos tag given a particular morphological component e_{l-1}. In addition, α_1 , α_2 and α_3 are contribution factors and $\alpha_1 + \alpha_2 + \alpha_3 = 1$. In order to calculate the prior probability $p\left(\frac{pos}{e_{l-1}}\right)$ we use the equation:

$$P\left(\frac{pos}{e_{l-1}}\right) = \frac{P(pos)\, p\left(\frac{e_{l-1}}{pos}\right)}{P(e_{l-1})} \quad (5.6)$$

In equation 5.6 $P(pos)$ is the posterior independent probability of the particular pos in the given corpus.

$p\left(\frac{e_{l-1}}{pos}\right)$ is the posterior probability of the final morphological component given the particular pos tag calculated from the tagged corpus. $P(e_{l-1})$ is the posterior independent probability of final morphological component calculated from the tagged corpus. This probability is calculated using equation 5.7.

$$P(e_{l-1}) = \sum_{i=1}^{k} p(pos_i)\, p\left(\frac{e_{l-1}}{pos_i}\right) \quad (5.7)$$

$P(e_{l-1})$ is calculated as a summation of all possible pos type *k*. $P(e_{l-1})$, the product of $p(pos_i)$, the posterior probability of the given pos type *i* in the corpus and $p\left(\frac{e_{l-1}}{pos_i}\right)$, the posterior probability of the final morpheme component given the pos tag *i*. The next section describes the calculation of contribution factors.

VI. Estimation of Contribution Factors

Using generalized iterative scaling technique, the contribution factors α_1, α_2 and α_3 are calculated so as to maximize the likelihood of the rest of the corpus. Coarse and fine steps are used for finding the parameters. For the fixed value of α_1, the $\alpha 2$ values are raised and simultaneously α_3 values are decreased, the correctness is showing the characteristic of inverted parabola in first quadrant of the graph. This inherent property is used for designing the following algorithm. Here, we are estimating the parameters in two steps, namely, coarse and fine steps. At coarse step, the value of increment and decrement is in the quantity of 0.1 for estimating coarse optimal value of the parameters. At fine step, the changes are in the quantity of .01 for estimating fine optimal value of parameters.

Algorithm for Estimating α_1, α_2 and α_3

```
Constraint  $\alpha_1 + \alpha_2 + \alpha_3 = 1$
Max =0.0
Optimal _solution = set(0 , 0 , 1 )
    Coarse step
        Initialize  $\alpha_1$=0.0
        Step in increment  $\alpha_1$ by 0.1
        Repeat the following step until $\alpha_1$<=1.0
            Initialize   val =0.0; and  $\alpha_2$=0.0;
            Step in increment  $\alpha_2$ by 0.1
            Repeat the following step until $\alpha_2$<=1.0- $\alpha_1$;
                set  $\alpha_3$=1.0-( $\alpha_1$+$\alpha_2$);
                if  the val  is less than  correctness ($\alpha_1$ , $\alpha_2$ , $\alpha_3$ )
                assign the   val = correctness ($\alpha_1$ , $\alpha_2$ , $\alpha_3$ )
                else  break the inner loop;
            Loop
            if the value of val greater than  max
            Max = val
            Optimal _solution = set ($\alpha_1$, $\alpha_2$, $\alpha_3$)
        Loop
    Fine Step
        Initialize   $\alpha_1$= optimal($\alpha_1$);
        Step in increment  $\alpha_1$ by 0.01
        Repeat the following step until $\alpha_1$<= optimal($\alpha_1$) +0.09
            Assign  val =0.0; and  $\alpha_2$= optimal($\alpha_2$);
            Step in increment  $\alpha_2$ by 0.01
            Repeat the following step until $\alpha_2$<= optimal($\alpha_2$)
                  +0.09- $\alpha_1$
                Assign  $\alpha_3$= 1.0-( $\alpha_1$+$\alpha_2$);
                If the val  is less than correctness ($\alpha_1$ , $\alpha_2$ , $\alpha_3$ ) )
                        then
                            val = correctness ($\alpha_1$ , $\alpha_2$ , $\alpha_3$ )
                        else
                            Break inner loop;
                Loop
            if the value of Max less than Val
                Val = Max
                Optimal _solution = set ($\alpha_1$, $\alpha_2$, $\alpha_3$)
        Loop
    ($\alpha_1$ , $\alpha_2$ , $\alpha_3$ ) ← value( Optimal _solution )
```

In this algorithm, correctness function will return the percentage of words correctly tagged by the language model with the corresponding contribution factors α_1 , α_2 and α_3. Optimal_solution is a set variable to store set of values of α_1 , α_2 and α_3. The function *optimal* with a variable as the parameter obtains the value of the corresponding parameter in the set *Optimal solution.*

Parameter α_1 , α_2 and α_3 for Language Model 1

TABLE II.
CONTRIBUTION FACTORS α1, α2, α3 FOR FINE STEP

α1	α2	α3	Correct (%)
0.30	0.21	0.49	86.67
0.31	0.21	0.48	86.64
0.32	0.22	0.46	86.82
0.33	0.23	0.44	86.55

After coarse step we got the values (0.3, 0.2, 0.5), thus, the estimated values of $\alpha 1 = 0.31$, $\alpha 2 = 0.21$ and $\alpha 3 = 0.48$

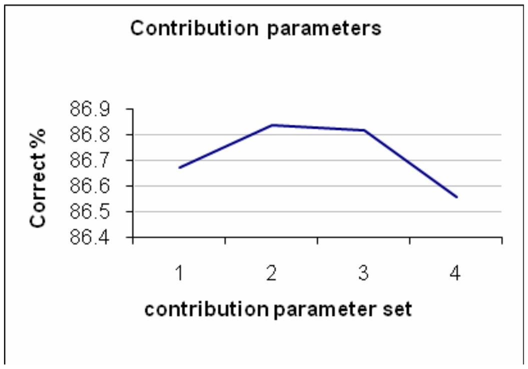


Fig 1. Correctness in % for the parameter set in Table 2.

Fig. 1 shows the graphical representation of Correctness for each set of parameter values as mentioned in Table II.

Parameter α_1 and α_2 for Language Model 2

TABLE III.
CONTRIBUTION FACTOR α1, α2 FOR COARSE STEP

	α1	α2	Correct (%)
1.	0.0	1.0	84.60
2.	0.4	0.6	93.06
3.	0.5	0.5	96.88
4.	0.6	0.4	98.37
5.	0.7	0.3	98.38
6.	0.8	0.2	99.37
7.	0.9	0.1	93.32
8.	1.0	0.0	90.00

There is no further improvement at fine step, so the estimated values are $\alpha 1 = 0.8$ and $\alpha 2 = 0.2$.

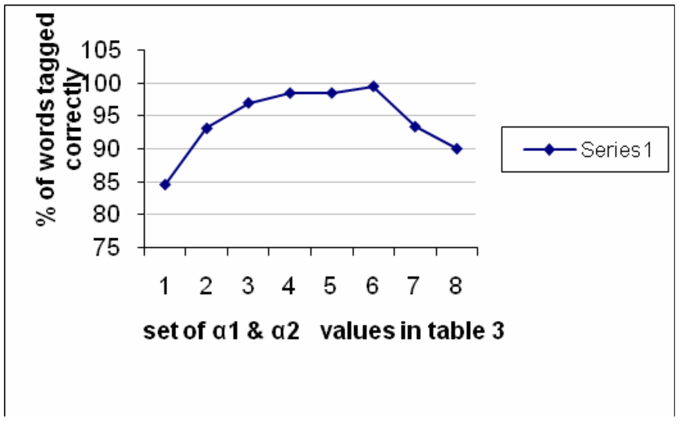


Fig. 2. Correctness in % for the parameter set from Table 3.

Fig. 2 shows the graphical representation of Correctness for each set of parameter values as mentioned in Table III.

VII. Evaluation and Results

The implemented system is evaluated as in Information Retrieval, which makes frequent use of precision and recall.

Precision is defined as a measure of the proportion of the selected items that the system got right.

$$precision = \frac{No.of\ items\ tagged\ correctly}{No.of\ items\ tagged} \quad (6.1)$$

Recall is defined as the proportion of the target items that the system selected.

$$\text{Re}\,call = \frac{No.of\ items\ tagged\ by\ the\ system}{No.of\ items\ to\ be\ tagged} \quad (6.2)$$

To combine precision and recall into a single measure of over all performance, the F-measure is defined as

$$F = \frac{1}{\alpha \frac{1}{p} + (1-\alpha)\frac{1}{R}} \quad (6.3)$$

where P is precision, R is recall and α is factor, which determine the weighting of precision and recall. A value of α is often chosen for equal weighting of precision and recall. With this α value the F-measure simplifies to

$$F = \frac{2PR}{P+R} \quad (6.4)$$

The perplexity is a useful metric for how well the language model matches with a test corpus. The perplexity is a variant of entropy. The entropy is measured by the following equation

$$H(tag\_type) = -\frac{1}{n}\sum_{i=1}^{n} \log(p(tag(w_i)) \quad (6.5)$$

$$perplexity(tag\_type) = 2^{H(tag\_type)} \quad (6.6)$$

The system is evaluated with a test corpus with 43,678 words in which 36,128 words are morphologically analyzed within our tag set. 6,123 words are named entity, while the remaining words are unidentified. The morphologically analyzed words are passed into tagger designed using our language model. The following table and graphs represents the results obtained by the language models for determining the tags.

TABLE IV.
Noun Categories

Postag	Recall	Precision	F-measure	Perplexity
<N>	0.98	0.99	0.99	1.91
<Pn>	0.98	0.98	0.98	2.61
<IN>	1.00	1.00	1.00	1.63
<NN>	0.48	0.97	0.64	8.58
<NP>	0.99	0.87	0.93	3.29
<PnP>	0.81	1.00	0.89	1.91
<INP>	0.69	0.47	0.56	9.18
<VnP>	0.01	1.00	0.02	1.57
<PN>	0.00	0.00	0.00	-
<NNP>	0.15	0.97	0.27	1.87
<NnP>	0.60	0.71	0.65	3.95
<PNP>	0.00	0.00	0.00	-
<Vn>	0.81	0.96	0.88	1.63
<Nn>	0.18	1.00	0.33	3.63

Table IV shows noun category type POS tags in which the tags <VNP>, <PN> and <PNP> are the noun category but its stem is of type verb. Due to this reason, the words of these tag types are wrongly tagged. These cases can be rectified by transformation based learning rules.

TABLE V.
Verb Categories

Postag	Recall	Precision	F-measure	Perplexity
<V>	0.99	0.93	0.95	3.88
<AV/V>	1.00	1.00	1.00	1.00
<FV>	1.00	0.99	0.99	1.00
<NFV>	1.00	0.90	0.95	1.00
<NFV/DT>	0.00	0.00	0.00	-
<Vvp>	0.93	0.92	0.92	2.74
<VP>	0.81	0.87	0.84	3.66
<Vinf>	0.71	0.88	0.79	3.98
<V/Vrp>	0.99	1.00	0.99	19.40
<Vrp>	0.98	0.78	0.87	1.81
<V/Vinf>	0.99	0.97	0.98	1.13
<VC>	0.42	0.91	0.57	3.47
<Vpost>	0.00	0.00	0.00	-

Table V shows verb category type POS tags. A word '*anRu*' in Tamil belongs to <NFV/DT> tag type. It has two type of context with the meaning of (i) *those day* (DT) and (ii) *not* (NFV). These cases have to be rectified by word sense disambiguation rules. Words of Tag type <Vpost> are relatively very few. By considering further morpheme components, these tag types can be identified.

TABLE VI.
OTHER CATEGORIES

Postag	Recall	Precision	F-measure	Perplexity
<DT>	1.00	1.00	1.00	1.00
<cNum>	1.00	1.00	1.00	1.00
<Madj>	1.00	0.99	0.99	1.00
<adj>	0.98	0.99	0.99	3.57
<Dadj>	0.99	1.00	0.99	1.00
<Iadj>	1.00	0.97	0.99	1.00
<PO>	1.00	1.00	1.00	1.00
<conj>	0.96	1.00	0.98	1.00
<par>	1.00	1.00	1.00	1.02
<Int>	1.00	1.00	1.00	1.00
<adv>	0.92	0.97	0.94	3.80
<SP>	1.00	1.00	1.00	9.22
<postAdj>	1.00	0.86	0.92	9.63
<SCC>	1.00	1.00	1.00	1.00
<PostP>	0.97	1.00	0.99	9.07

Table VI shows POS tags of other category types. The occurrence of categories of this type is less as compared with noun type and verb type.

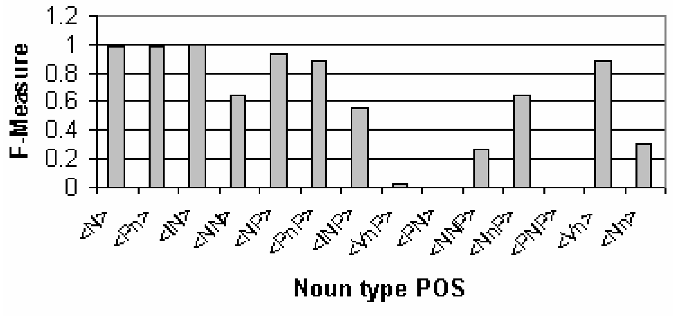


Fig 3. F-measure for POS tag of noun categories.

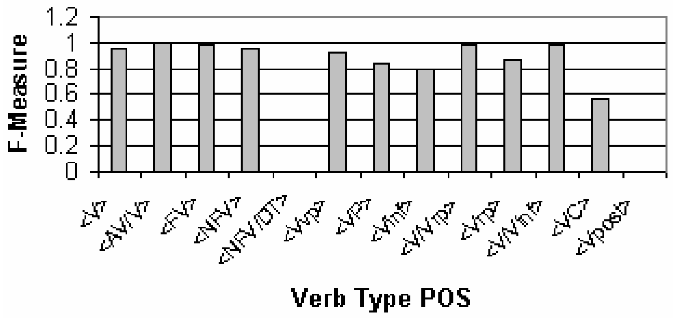


Fig 4. F-measure for POS tag of verb categories.

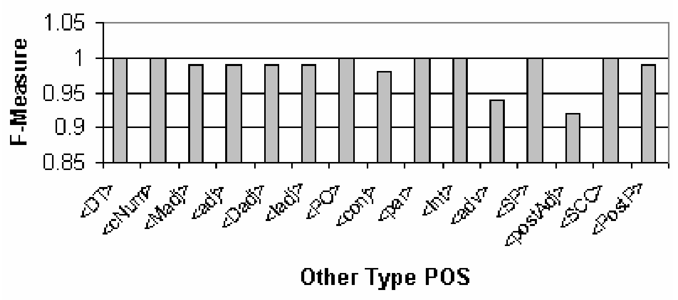


Fig 5. F-measure for POS tag of other categories.

The same test is repeated for another corpus with 62,765 words, in which the used analyzer identifies 52,889 words. 49,340 words are correctly tagged using the suggested language model. These two tests are tabulated in Table VII.

TABLE VII.
OVERALL ACCURACY AND PERPLEXITY.

	Words	Correct	Error	Accuracy (%)	Perplexity
T1	36,128	34,656	1,472	95.92	153.80
T2	36,889	34,840	2,049	94.45	153.96

As the perplexity values for both test cases are more or less equal, we can conclude that the formulated language model is independent of the type of untagged data.

VIII. CONCLUSION AND FUTURE WORK

We have presented a part-of-speech tagger for Tamil, which uses a specialized language model. The input of a tagger is a string of words and a specified tag set similar to the described one. The output is a single best tag for each word. The overall accuracy of this tagger is 95.92%. This system can be improved by adding a module with transformation based learning technique and word sense disambiguation. The same approach can be applied for any morphologically rich natural language. The morpheme based language model approach can be modified for chunking, named entity recognition and shallow parsing.

REFERENCES

[1] Aniket Dalal, Kumar Nagaraj, Uma Sawant, Sandeep Shelke , Hindi Part-of-Speech Tagging and Chunking : A Maximum Entropy Approach. In: Proceedings of the NLPAI Machine Learning Contest 2006 NLPAI, 2006.
[2] Nizar Habash , Owen Rambow ,Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in one Fell Swoop. In: Proceedings of the 43rd Annual Meeting of the ACL, pages 573–580, Association for Computational Linguistics, June 2005.
[3] D. Hiemstra. Using language models for information retrieval. PhD Thesis, University of Twente, 2001.
[4] S. Armstrong, G. Robert, and P. Bouillon. Building a Language Model for POS Tagging (unpublished), 1996.
http://citeseer.ist.psu.edu/armstrong96building.html
[5] P. Anandan, K. Saravanan, Ranjani Parthasarathi and T. V. Geetha. Morphological Analyzer for Tamil. In: International Conference on Natural language Processing, 2002.
[6] Thomas Lehman. A grammar of modern Tamil, Pondicherry Institute of Linguistic and culture.
[7] Sandipan Dandapat, Sudeshna Sarkar and Anupam Basu. A Hybrid Model for Part-of-speech tagging and its application to Bengali. In: Transaction on Engineering, Computing and Technology VI December 2004.

[8] Barbara B. Greene and Gerald M. Rubin. Automated grammatical tagger of English. Department of Linguistics, Brown University, 1971.

[9] S. Klein and R. Simmons. A computational approach to grammatical coding of English words. JACM, 10:334-337, 1963.

[10] Theologos Athanaselies, Stelios Bakamidis and Ioannis Dologlou. Word reordering based on Statistical Language Model. In: Transaction Engineering, Computing and Technology, v. 12, March 2006.

[11] Sankaran Baskaran. Hindi POS tagging and Chunking. In: Proceedings of the NLPAI Machine Learning Contest, 2006.

[12] Lluís Márquez and Lluis Padró. A flexible pos tagger using an automatically acquired Language model. In: Proceedings of ACL/EACL'97.

[13] K. Rajan. Corpus analysis and tagging for Tamil. In: Proceeding of symposium on Translation support system STRANS-2002

[14] T. Brants. TnT - A Statistical Part-of-Speech Tagger. User manual, 2000.

[15] Scott M. Thede and Mary P. Harper. A second-order Hidden Markov Model for part-of-speech tagging. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, pages 175—182. Association for Computational Linguistics, June 20--26, 1999.

[16] Eric Brill. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. Computation Linguistics, 21(4):543- 565, 1995.

Modeling a Quite Different Machine Translation using Lexical Conceptual Structure

Nadia Luiza Dinca

***Abstract*— The goal of this study is to outline the readability of an Example-Based Machine Translation for any pair of languages by means of the language-independent properties of the lexical conceptual structure (LCS). We describe LCS as a representation of traditional dependency relationships and use in experiments an isolated pair of verbs, extracted from Orwell's "1984" parallel English – Romanian texts. We discuss the mental models in terms of specific knowledge structures. Finally, we present LCS-Based Machine Translation from the point of view of a complex adaptive system and present our ongoing work in order to capture the neutral linguistic core of any mental model corresponding to the real world.**

***Index Terms*—Lexical conceptual structure, machine translation, readability, complex adaptive system.**

I. Introduction

THE paradigm of 'translation by analogy', used to characterize the Example-Based Machine Translation, proposes the use of an unannotated database of examples (possibly collected from a bilingual dictionary) and a set of lexical equivalences simply expressed in terms of word pairs. The matching process is focused on checking the semantic similarity between the lexical items in the input sentence and the corresponding items in the candidate example. In fact, an Example-Based Machine Translation database is used for different purposes at the same time: as a source of sentence frame pairs, and as a source of sub-sentential translation pairs.

In this paper, we aim to present a new direction in designing the structural Example-Based Machine Translation. We are expanding the original Nagao's model [1] in order to obtain the translation of a complete sentence by utilizing more than one translation example and combine some fragments of them. Usually, the translation examples are represented as dependency trees with correspondence links between sub-trees. I propose here an issue to replace the traditional representation of these translation examples with lexical conceptual structure (LCS), a kind of a compositional abstraction with language-independent properties that transcend structural idiosyncrasies [2].

For an input sentence, there is a matching expression, naming a pointer to a translation unit, i.e., a lexical conceptual structure to be found in a manually constructed database of examples. The pointer is optionally followed by a list of commands for deletion/replacement/adjunction of nodes dominated by the node pointed to. The replaced or adjoined elements are other matching expressions. The data encapsulation of the translation examples is related to the modularity demands of the sub-sequences that inherit the features of the dominating units.

It is obviously that a machine translation system requires a substantial amount of translation knowledge, typically embodied in bilingual dictionaries, transfer rules, example databases or statistical models. Our approach seeks to obtain as much of this knowledge as possible by expressing translation examples in LCS- dependency trees.

The real value of this LCS–Based Machine Translation is offered by readability, since the machine captures the mental models of any language and therefore, isolates the correspondence links between the translation units.

II. Lexical Conceptual Structure

Traditionally, a translation example contains three parts:

- Dependency tree, adapted for the source language;
- Dependency tree, created for the target language;
- Correspondence links.

In this paper, the dependency trees are replaced with lexical conceptual structures, built by hand, for English-Romanian linguistic project. We have isolated pairs of verbs, extracted from Orwell's "*1984*" parallel English-Romanian texts.

The verbs were characterized from the point of view of syntagmatic and paradigmatic relations, using the verb classes and alternations described by Levin [3] and Visdic [4]**,** using a multilingual ontology editor.

The semantic properties of a lexical item are totally reflected in a number of relations associated to different types of contexts. These affinities developed by a word regarding a context are syntagmatic or paradigmatic. Lexical semantic relations are essentially paradigmatic, even if they can be combined directly with or be based on, some analytical elements or expression of properties, as in WordNet.

According to [5], a lexical conceptual structure is a directed graph with a root, where each root is associated with a special kind of information, including a *type*, a *primitive* and a *field*. The types name are Event, Path, Manner, Property, Thing; the fields refer to Locational, Possessional, and Identificational values. The primitive of a LCS node is splitted into structural primitive (e.g., *go, cause, act*) and constants (e.g., *reduce+ed, slash+ingly, face+ut, caine+este*). For example, the top node in the root LCS of the verb *follow* ("*the eyes follow you*") has the structural primitive ACT_ON in the locational field. Its subject is a star-marked LCS with the restriction of being a type thing. The number "1" specifies the thematic role of the

Manuscript received July 22, 2008. Manuscript accepted for publication October 30, 2008.

Nadia Luiza Dinca is with Research Institute for Artificial Intelligence, Bucharest, Romania (hnadia_luiza@hotmail.com).

agent. The second child node is an argument position and needs to be of type thing, too; its number "2" represents the clue of the theme:

(DEF_WORD: "follow"
LCS: (act_on loc (thing 1) (* thing 2)))*

The format for thematic roles is the following:

1. Any thematic role preceded by an underscore (_) is obligatory.
2. Any thematic role preceded by a comma (,) is optional.
3. Prepositions inside parentheses indicate that the corresponding phrases must necessarily be headed by the specified prepositions.
4. An empty set of parentheses ( ) indicates that there necessarily must be a prepositional head, but it is left unspecified.
5. The difference between the main communication and the incident constructions referring to a second level of speech is marked by indices "1" for the first level, and "2" for the second one.

In the notation we used, the DEF_WORD, THEM_ROLES and LCS represent the Dorr's description attributes [2], [6]. We added TE, CLASS, SYNSET attributes with the values of *translation equivalent*, *semantic* and *synonymic classes*. The LCS specifies a star marker (*) for very explicitly realized argument and modifier. The star marker forces logical constituents to be realized compositionally at different levels.

Consider the sentence (1a). This can be represented as shown in (1b), glossed as (1c):

```
1a. I walk to cinema.
1b. (event go loc
         (thing I+)
        (path to loc
            (thing I+)
            (position at loc (thing I+) (thing cinema+)))
               (manner walk +ingly))
1c. 'I move (location) to the cinema in a walking manner.'
```

The next figure shows the lexicon entry for the contextual sense of the English verb 'walk' with several pieces of information, such as the root form of the lexical item, its translation equivalent, the semantic verb class and the synset, introduced by the fields: DEF_WORD, CLASS, SYNSET and TE. The thematic roles appearing in the root LCS entry are classed in a canonical order that reflects their relative surface order: first available in this case is theme, with the obligatory specification; the last two optional roles are source and goal:

```
(DEF_WORD: "walk"
TE: "merge"
CLASS: "51.3.2"
SYNSET: "walk: 4", "jog: 3",
             "run: 29"
THEM_ROLES: "_th ,src() ,goal()"
LCS: (event go loc ( * thing 2)
                 (( * path from 3) loc (thing 2)
                    (position at loc (thing 2) (thing 4)))
                 (( * path to 5) loc (thing 2)
                    (position at loc (thing 2) (thing 6)))))
```

The field LCS introduces the uninstantiated LCS corresponding to the underlying meaning of the word entry in the lexicon. The top node for '*walk*' has the structural primitive *go* in the locational field. Its subject, marked with a star "*", indicates that the node must be filled recursively with other lexical entries during semantic composition. The only restriction is that the filler must be of type '*thing*'. The second and third child nodes are in argument positions filled with the primitives FROM and TO; the numbers 3 and 5 mark the source and goal particle; the numbers 4 and 6 stand for source and goal.

III. Lexical Conceptual Structure-Based Machine Translation

The main problem concerning an Example-Based Machine Translation is how to use a translation example for translating more than one source sentence. The solution described here uses the lexical conceptual structure as a representation of traditional dependency relationships. The words are introduced in the dictionary with the specification of semantic class, the synset and LCS –for the verb lexical entry– and with the notation of thematic roles and LCS, for all other parts of speech lexical entries (i.e., nouns, pronouns, numbers, adverbs, adjectives, prepositions).

The basic properties of LCS are: idempotence, reflexivity and compositionality:

- *Idempotence*: a LCS multiplied by itself, gives itself as a result. For example, the root LCS for "*follow*" can combine with any members of the semantic class "51.6" ("*watch*: 2", "*observe*: 7", "*follow*: 13") and the resulted LCS has the same characteristics as the original one.
- *Reflexivity* means the act of self-reference. For example, the LCS created for the verb "*divide*" can refer to any members of the synset "*divide*: 1", "*split*: 1", "*split up*: 2", "*separate*: 4", "*dissever*: 1", "*carve up*: 1").
- *Compositionality* states that the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them. It can be considered the most important property because it allows to a translation equivalent to be used in order to translate more than one source sentence.

Let's consider the translation of the following sentence:

(1) He dipped the pen into the ink.

If the translation database contains the translation examples (2) and (3), then we can translate sentence (1) into (4) by imitating examples and combining fragments of them:

(2) He dipped the pen into the blue liquid.
El isi inmuie penita in lichidul albastru.
(3) I bought an ink bottle.
Eu am cumparat o cutie cu cerneala.
(4) El isi inmuie penita in cerneala.

Formally, a translation example consists of three parts:

- Source LCS-tree (ELCS; English Lexical Conceptual Structure);
- Target LCS-tree (RLCS; Romanian Lexical Conceptual Structure);
- Correspondence links.

Each number prefixed by "*e*" or "*r*" represents the identifier of the sub-tree and each node in a tree contains a word (in root form), a thematic role and its corresponding part of LCS. A correspondence link is represented as a pair of identifiers.

If the translation of an identical sentence is not available in the bilingual corpus, the EBMT system makes use of some sort of similarity metric to find the best matching translation examples. Suitable sub-sequences are iteratively replaced, substituted, modified or adapted in order to generate the translation. While the replacement, substitution, modification or adaptation is rule-driven, the mapping of a source segment into an equivalent target segment is guided from translation examples.

According to [1], the concept *matching expression* (ME) is defined as in the following:

```
<ME> ::= [<ID> | <ME - Commands>]
<ME – Commands> ::=
        [ ]
or [<ME – Command> | <ME – Commands>]
<ME – Command> ::=
        [d, <ID>]              %% delete <ID>
or      [r, <ID>, <ME>]  %% replace <ID> with <ME>
or      [a, <ID>, <ME>] %% add <ME> as a child of
                                    root node of <ID>
```

Under these assumptions, the LCS trees (a) can be represented by the matching expression (b):

```
(a) elcs_e ([e11, [dip, cause],
        // TE1
            [e12, [he, _ag, (* thing 1)]],
             [e13, [pen, _th, (go loc (* thing 2))]],
              [e14, [into, _goal (into), ([into] loc (thing 2))],
               [e15, [liquid, _goal (into), (thing 6)]
                [e16, [blue, _goal (into), (thing 6)]]]]])

        rlcs_e ([r11, [inmuia, cause],
             [r12, [el, _ag, (* thing 1)]],
              [r13, [penita, _th, (go loc (* thing 2))]],
                [r14, [in, _goal (in), ([in] loc (thing 2))],
                  [r15, [lichidul, _goal (into), (thing 6)]
                     [r16, [albastru, _goal (into), (thing 6)]]]]

 %% clinks: ([e11, r11], [e12, r12], [e13, r13], [e14, r14],
[e15, r15])

        elcs_e ([e21, [buy, cause_exchange],
        // TE2
             [e22, [I, _ag, (* thing 1)]],
             [e23, [bottle, rec, (go poss (* thing 2))],
          [e24, [ink, _th, (go poss (* thing 2))]]]]])

        rlcs_e ([r21, [cumpara, cause_exchange]
             [r22, [Eu, _ag, (* thing 1)]],
             [r23, [cutie, rec, (go poss (* thing 2))],
          [r24, [cerneala, _th, (go poss (* thing 2))]]]]])

 %% clinks: ([e21, r21], [e22, r22], [e23, r23], [e24, r24])

 (b) [e11, [(r, e15, [e24]), (d, e16)]  // for source language
```

The first step is matching fragments of the input sentence *He dipped the pen into the ink* against a database of real examples. Two sequences can be found: *he dipped the pen into the* and *ink*, respectively. The data encapsulation of the translation unit provides a kind of logical independence of the sub-sequences. Therefore, the complex translation unit, i.e., the entire sentence, is splitted into two sub-units; the first sub-unit encapsulates the second sub-sequence 'into the', while the last has lexicalized only the head and the specifier, and it waits for the noun corresponding to the input modifier.

TE_1 joins TE_2 and the result matches against the input only if the result has the same structure as the source sentence and its arguments have values for the same type, primitive and field as the input. The inheritance develops translation sub-units incrementally through defining new objects in terms of one previously object defined. It is applied only from noun to adjective, from preposition to noun, and the inherited features allow the matching by literals, instead of complex translation units.

In the example above, the preposition node for *into* requires a daughter node with the feature tuple (_goal (into), (thing 6)), instantiated by the lexical items *black liquid*. Also, the item 'black' lexicalizes the daughter node of the noun 'liquid' and it inherits its feature tuple. The inheritance is possible only if the selected literal has the same type as the input needed to match. The LCS-tree created for the second translation example shows the value of theme for the bi lexeme '*ink* ⇔ *cerneala*' and the type 'thing', so the literal is corresponding to the input word.

In the transfer step, the system replaces every identifier in the source matching expression with its corresponding identifier:

SME= [e11, [r, e15, [e24]]
TME= [r11, [r, r15, [r24]

In the composition step, the lexical conceptual structure-tree is composed according to the target matching expression:

```
TME = [r11, [r, r15, [r24]]
TLCS=   ([r1, [inmuia, cause],
            [r2, [el, _ag, (* thing 1)]],
            [r3, [penita, _th, (go loc (* thing 2))]],
            [r4, [in, _goal (in), ([in] loc (thing 2))],
             [r5, [cerneala, _goal (into), (thing 6)]]]]])
%% El isi inmuie penita in cerneala.
```

IV. Discussion

The EBMT idea is to translate by analogy [7], [8]. But what is happening when the translator finds only a synonym of a given input, instead of its matching in different associations? In our opinion, the solution is to consider a new mental model as we explain below.

Any translation means to acquire new knowledge about the real world by taking into consideration the older knowledge organized into mental schemata by the system. The synonymy relation, the classification into semantic verb classes, the verb arity and the thematic roles, the lexical-semantic representation in terms of primitive, fields and types- all represents mental schemata corresponding to different levels of the linguistic representation [9], [10]. Each of these schemata identifies distributed information and a starting point in understanding, learning and transferring the code from a source language to a target language.

When the system confronts with a new situation, i.e., the presence of a synonym, instead of its matching, it must unify all the distributed schemata and organize them into a new mental model, which is, in our case, the lexical conceptual structure tree representation of the translation examples. Therefore, while the schemata are generic pre-compiled knowledge structures, the mental models are specific knowledge structures, built in order to figure a new situation using this generic knowledge.

Lexical forms written in the example side may be a synonym for the matched input word and we must modify the input side before constructing a target structure. The matching is analogue to a reasoning issue in natural language, where an unknown word is translated depending on the associations with known items, participants in the same kind of context and used as pointers for the semantic interpretation of the item given.

Usually, the context acceptance means the words which occur with a lexical item in order to disambiguate it semantically. For this approach, the context refers to the semantic class and synonymy relation of the verb given in the lexical entry (e.g., the context of *jog* is the class “51.3.2.” and the synset “*run*: 29, *jog*: 3, *walk*: 4, *zigzag*: 1, *jump*: 1, *roll*: 12”, instantiated for the sentence *John jogged to school*). Formally, the source verb “a” may be translated into the target verb “b” when:

a. there is the appropriate equivalent in the translation database;

b. there is a source verb “c” which is:

b.1. in a synonymy relation with “a”;

b.2. in the same semantic verb class with “a”;

b.3. the translation equivalent of “a”.

The following example shows how to obtain a translation if the system has a translation example and its context:

Input Sentence: *He runs to school.*

Translation Example:

He jogs to school = El alearga la scoala.

Context:

jog: 3= run: 29

jog: <- class “51.3.2.”

jog: (synset) <- (run: 29, jog: 3, walk: 4, zigzag: 1, jump: 1, roll: 12)

Target Sentence: *He runs to school = El alearga la scoala.*

Even LCS is not a deep knowledge representation; it captures the semantics of a lexical item through a combination of semantic structure (which is something the verb shares with a semantic verb class) and semantic content (which is specific to the verb itself). The semantic structure relies also on the subcategorization level of linguistic representation by the fact that there are three ways a child node relates to its parents: as a subject (maximally one), as an argument, or as a modifier. Considering this relation between different levels of linguistic representation, I can define a well-formedness principle for LCS-based MT:

A translation is well-formed if:

i. There is an appropriate equivalent in the database examples;
ii. There is an appropriate context (semantic verb class and synset) for the input verb;
iii. The LCS children are lexicalized (if there is one minimally) and associated with information including a type, a primitive and a field.

The type of the LCS created for ‘*give*’ is *Event*, its structural primitive is *go*, which appears in many generalized movements, and the field which specifies the domain is *Possessional*. The thematic roles are organized into the grid: "_ag_th_goal(to)". If the sentence contains a form which doesn’t fill all the values of LCS, it won’t be lexicalized and the sentence won’t be generated:

Input: *He gives.*

TE$_1$: *He gives fruits to children.* ⇔ *El da fructe copiilor.*

TE$_2$*: He buys a kilo of fruits.* ⇔ *El cumpara un kilogram de fructe*.

LCS _input :

```
((cause (* thing 1)
  (go poss (* nill)
     ((* to 5) poss (nill) (at poss (nill) (nill))))
 (give+ingly 26))
```

V. Conclusion and Future Work

The LCS-Based Machine Translation can be a powerful linguistic tool because it allows clear readability of the results. Since the information contained in these lexical conceptual structures is language-independent, we consider them an interesting issue to capture the core of a machine translation, which is not centered on any particular pair of language.

In fact, the LCS-Based Machine Translation has the behavior of a *complex adaptive system*, which means:

- *Patterns of activity:* the linguist has to describe the LCS for every verb, considered a lexical entry in translation database;
- *Self-organization*: the structure receives a holistic interpretation, including a type, a primitive and a field;
- *Collective behavior*: the verbal core is extended, so that the root LCS accepts all the verbs which respect the same semantic class.

In conclusion, this paper aims to present the improvement of readability of an Example-Based Machine Translation in terms of lexical conceptual structure. It is only a beginning of a more profound study that will be developed in order to characterize a machine translation without the old centering on the particular pair of languages – source and target languages.

The future work involves the following practical tasks of investigation:

1. Creating LCS-lexicons by taking the most frequent verb pairs in "*1984*" corpus, Romanian and English versions (with a frequency threshold of 5 occurrences, minimally).
2. Considering French as a new language for translation and creating also LCS-tree representations.
3. Organizing the LCS-trees of English, Romanian and French languages into new mental models using the encapsulation and a top ontology. We will try to reduce the language differences and to isolate the neutral linguistic core of any mental model, corresponding to the real world.

References

[1] S. Satoshi and M. Nagao. Toward Memory- Based Translation. In: Proceedings of the 13th conference on Computational linguistics, Finland: Helsinki, 1990, pp. 247-252.

[2] B. J., Dorr. LCS VerbDatabase, Online Software Database of Lexical Conceptual Structures and Documentation. http://www.umiacs.umd.edu/~bonnie/LCS_Database_Documentation.html

[3] B. Levin. English Verb Classes and Alternations - A Preliminary Investigation. The University of Chicago Press, 1993.

[4] http://nlp.fi.muni.cz/projekty/visdic/

[5] A. N. Fazil and B. J. Dorr. Generating a Parsing Lexicon from an LCS-Based Lexicon. In: Proceedings of the LREC-2002 Workshop on Linguistic Knowledge Acquisition and Representation, Spain: Las Palmas, 2002, pp. 43-52.

[6] H. Nizar, B. J. Dorr and D. Traum. Hybrid Natural Language Generation from Lexical Conceptual Structures. Machine Translation, 18:2, 2003, pp. 81—128

[7] M. Carl and A. Way (eds.) Recent Advances in Example-Based Machine Translation. Kluwer Academic Publishers, 2003.

[8] H. Tanaka. Verbal case frame acquisition from a bilingual corpus: gradual knowledge acquisition. In: Proceedings of the 13th conference on Computational linguistics, Kyoto, Japan, 1994, pp. 727-731.

[9] R. Green, B. J. Dorr and Ph. Resnik. Inducing Frame Semantic Verb Classes from WordNet and LDOCE. In: Proceedings of the Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 96-102.

[10] A. N. Fazil and B. J. Dorr. Generating A Parsing Lexicon from an LCS-Based Lexicon. In: Proceedings of the LREC-2002 Workshop on Linguistic Knowledge Acquisition and Representation, Las Palmas, Canary Islands, Spain, 2002, pp. 43-52.

Named Entity Recognition in Hindi using Maximum Entropy and Transliteration

Sujan Kumar Saha, Partha Sarathi Ghosh, Sudeshna Sarkar, and Pabitra Mitra

***Abstract*—Named entities are perhaps the most important indexing element in text for most of the information extraction and mining tasks. Construction of a Named Entity Recognition (NER) system becomes challenging if proper resources are not available. Gazetteer lists are often used for the development of NER systems. In many resource-poor languages gazetteer lists of proper size are not available, but sometimes relevant lists are available in English. Proper transliteration makes the English lists useful in the NER tasks for such languages. In this paper, we have described a Maximum Entropy based NER system for Hindi. We have explored different features applicable for the Hindi NER task. We have incorporated some gazetteer lists in the system to increase the performance of the system. These lists are collected from the web and are in English. To make these English lists useful in the Hindi NER task, we have proposed a two-phase transliteration methodology. A considerable amount of performance improvement is observed after using the transliteration based gazetteer lists in the system. The proposed transliteration based gazetteer preparation methodology is also applicable for other languages. Apart from Hindi, we have applied the transliteration approach in Bengali NER task and also achieved performance improvement.**

***Index Terms*—Gazetteer list preparation, named entity recognition, natural language processing, transliteration.**

I. Introduction

Named entity recognition is a subtask of information extraction that seeks to locate and classify the proper names in a text. NER systems are extremely useful in many Natural Language Processing (NLP) applications such as question answering, machine translation, information extraction and so on. NER systems have been developed for resource-rich languages like English with very high accuracies. But construction of an NER system for a resource-poor language is very challenging due to unavailability of proper resources.

English is resource-rich language containing lots of resources for NER and other NLP tasks. Some of the resources of English language can be used to develop NER system for a resource-poor language. Also English is used widely in many countries in the world. In India, although there are several regional languages like Bengali, Hindi, Tamil, Telugu etc., English is widely used (also as subsidiary official language). Use of the Indian languages in the web is very little compared to English. So, there are a lot of resources on the web, which are helpful in Indian language NLP tasks, but they are available in English. For example, we found several relevant name lists on the web which are useful in Hindi NER task, but these are in English. It is possible to use these English resources if a good transliteration system is available.

Transliteration is the practice of transcribing a word or text in one writing system into another. Technically most transliterations map the letters of the source script to letters pronounced similarly in the goal script. Direct transliteration from English to an Indian language is a difficult task. As our primary objective is to make the available English gazetteer lists useful for the Hindi NER task, we propose a two-phase transliteration, which is capable to do that.

The transliteration module uses an intermediate alphabet, which is designed by preserving the phonetic properties. The English names in the name lists are transliterated to the intermediate alphabet. A Hindi word, when it needs to be checked whether it belongs to a gazetteer list, is also transliterated into the intermediate alphabet. For an English-Hindi word pair, if their transliterated intermediate alphabet strings are the same, then we conclude that the English word is the transliteration of the Hindi word.

In this paper, we have identified suitable features for Hindi NER task. These features are used to develop a Maximum Entropy (MaxEnt) based Hindi NER system. The highest F-value achieved by the MaxEnt based system is 75.89. Then the transliteration based gazetteer lists are incorporated in the system and F-value is increased to 81.12. The improvement in accuracy demonstrates the effectiveness of the proposed transliteration approach.

The proposed transliteration module is applicable to other languages also. We have chosen another language Bengali and applied the transliteration approach for using the English gazetteers in Bengali NER task. Also in Bengali, the addition of the transliteration based gazetteer lists increases the accuracy.

The paper is structured as follows. Varios NER techniques and transliteration systems for different languages are discussed in Section II. In Section III, the architecture of the MaxEnt based Hindi NER system is presented. Then two-

Manuscript received July 10, 2008. Manuscript accepted for publication October 22, 2008.

Sujan Kumar Saha is with Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India (email: sujan.kr.saha@gmail.com).

Partha Sarathi Ghosh is with HCL Technologies, Bangalore, India (email: partha.silicon@gmail.com).

Sudeshna Sarkar is with Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India (email: shudeshna@gmail.com).

Pabitra Mitra is with Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India (email: pabitra@gmail.com).

phase transliteration system is discussed in Section IV. In the next section, the prepared gazetteers and the corresponding experimental results are discussed. The experiments on Bengali NER are summarized in Section VI. Section VII presents the overall discussion. Finally Section VIII concludes the paper.

II. Previous Work

There are a variety of techniques for NER. Two broadly classified approaches to NER are:

- Linguistic approach and
- Machine learning based approach.

The linguistic approach is the classical approach to NER. It typically uses rules manually written by linguists. Though it requires a lot of work by domain experts, a NER system based on manual rules may provide very high accuracy. There are several rule-based NER systems, containing mainly lexicalized grammar, gazetteer lists, and list of trigger words, which are capable of providing F-value of 88-92 for English [9], [13], [18].

The main disadvantages of these rule-based techniques are: they require huge experience and grammatical knowledge on the particular language or domain; the development is generally time-consuming and sometimes changes in the system may be hard to accommodate. Also, these systems are not transferable, which means that one rule-based NER system made for a particular language or domain, cannot be used for other languages or domains.

The recent Machine Learning (ML) techniques make use of a large amount of annotated data to acquire high-level language knowledge. ML based techniques facilitate the development of recognizers in a very short time. Several ML techniques have been successfully used for the NER task. Here we mention a few NER systems that have used ML techniques.

'Identifinder' is one of the first generation ML based NER systems which used Hidden Markov Model (HMM) [2]. By using mainly capital letter and digit information, this system achieved F-value of 87.6 on English. Borthwick used MaxEnt in his NER system with lexical information, section information and dictionary features [3]. He had also shown that ML approaches can be combined with hand-coded systems to achieve better performance. He was able to develop a 92% accurate English NER system. Mikheev *et al.* has also developed a hybrid system containing statistical and hand coded system that achieved F-value of 93.39 [14].

Other ML approaches like Support Vector Machine (SVM), Conditional Random Field (CRF), Maximum Entropy Markov Model (MEMM) are also used in developing NER systems. Combinations of different ML approaches are also used. For example, we can mention a system developed by Srihari *et al.*, which combined several modules, built by using MaxEnt, HMM and handcrafted rules, that achieved F-value of 93.5 [17].

The NER task for Hindi has been explored by Cucerzan and Yarowsky in their language independent NER which used morphological and contextual evidences [5]. They ran their experiments with 5 languages: Romanian, English, Greek, Turkish and Hindi. Among these, the accuracy for Hindi was the worst. For Hindi the system performance has F-value of 41.70 with very low recall 27.84% and about 85% precision. A successful Hindi NER system is developed by Li and McCallum using CRF with feature induction [12]. They automatically discovered relevant features by providing a large array of lexical tests and using feature induction to automatically construct the features that mostly increase conditional likelihood. In an effort to reduce overfitting, they used a combination of a Gaussian prior and early stopping. The training set consisted in 340 K words. Feature induction constructed 9,697 features from an original set of 152,189 atomic features; many are position-shifted but only about 1% are useful. Highest test set accuracy of their system is the F-value of 71.50. The MaxEnt based Hindi NER system developed by Saha *et al.* has achieved F-value of 80.01 [16]. The system has used word selection and word clustering based feature reduction techniques to achieve this result.

Transliteration is also a very important topic and several transliteration systems for different languages have been developed using different approaches. The basic approaches of transliteration are phoneme based or spelling-based. To mention a phoneme-based statistical transliteration system from Arabic to English is developed by Knight and Graehl [10]. This system used finite state transducer that implemented transformation rules to do back-transliteration. A spelling-based model that directly maps English letter sequences into Arabic letters is developed by Al-Onaizan and Knight [1]. There are several transliteration systems for English-Japanese [8], English-Chinese [11], English-Spanish [4] and many other languages to English.

But very few attempts were made to develop transliteration systems for Indian languages to English or other languages. We can mention a transliteration system for Bengali-English transliteration developed by Ekbal *et al.* [7]. They have proposed different models modifying the joint source channel model. In that system a Bengali string is divided into transliteration units containing a vowel modifier or *matra* at the end of each unit. Similarly, English string is also divided into units. Then various unigram, bigram or trigram models are defined depending on consideration of contexts of the units. Linguistic knowledge in the form of possible conjuncts and diphthongs in Bengali and their representations in English are also considered. This system is capable of transliterating mainly person names. The highest transliteration accuracy achieved by the system is 69.3% Word Agreement Ratio (WAR) for Bengali to English and 67.9% WAR for English to Bengali transliteration.

III. MaxEnt Based Hindi NER System

We have used MaxEnt classifier to develop the system. Selection of an appropriate feature set is very important to train a ML based classifier. As language resources and tools are limited in Hindi, we have given the most importance to the features. MaxEnt model has the capability to use different features to compute the conditional probabilities.

In Hindi, there is no capitalization of letters to distinguish proper nouns from other nouns. Capitalization is a very important feature for English as most of the names are capitalized. Due to absence of the capitalization feature, Hindi NER task is difficult. Also, person names are more diverse in Indian languages; many common words are used as names.

In the following sections we discuss the features that we have identified and used to develop the Hindi NER system.

A. Feature Description

The features that we have identified for the Hindi NER task are:

- *Surrounding Words*

As the surrounding words are very important to recognize a NE, previous and next words of a particular word are used as features. As a feature, previous *m* words ($w_{i-m}...w_{i-1}$) to next *n* words ($w_{i+1}...w_{i+n}$) can be treated depending on the training data size, total number of candidate features etc. During experiment different combinations of previous four words to next four words are used as features. These features are multi-valued. For a particular word w_i, its previous word w_{i-1} can be any word in the vocabulary, which makes the feature space very high. Such high-dimensional features do not work well if amount of training data is not sufficient.

- *Binary Word Feature*

The multi-valued feature can be modified as a set of binary feature to reduce the feature space. Class specific lists are compiled taking the frequent words present in a particular position. For example, for the previous word of the *person* class, frequent words are collected in *PrevPerson* list. Such lists are compiled for each class and each position (previous *m* to next *n*). Now *C* binary features replace the word feature for a particular position, where *C* is the number of classes. The word in a particular position is checked whether it is in the corresponding position list for a class or not. Firstly we have prepared the lists blindly by taking the words occurring at least four times in a particular position corresponding to a class.

- *Context Lists*

The idea of binary word feature is used to define the class context features. Context words are defined as the frequent words present in a word window for a particular class. In our experiment we have listed all the frequent words present anywhere in $w_{i-3}...w_{i+3}$ window for a particular class. Then this list is manually edited to prepare the context word list for a class. For example, location context list contains *roda* (road), *rajdhani* (capital), *sthita* (located in), *jakar* (going to) etc. The feature is defined as, for a word w_i, if any of its surrounding words ($w_{i-3}...w_{i+3}$) is in a class context list then the corresponding class context feature is 1.

- *Named Entity Tags of Previous Words*

Named entity (NE) tags of the previous words ($t_{i-m}...t_{i-1}$) are used as feature. This feature is dynamic. The value of the feature for w_i is available after obtaining the NE tag of w_{i-1}.

- *First Word*

If the word is the first word of a sentence, then this feature is set to 1. Otherwise, it is set to 0.

- *Containing Digit*

If a word contains digit(s) then the binary feature *ContainsDigit* is set to 1.

- *Made up of 4 Digits*

For a word *w* if all the characters are digits and having only 4 digits in *w*, then the feature *fourDigit* is set to 1. This feature is helpful for identifying year. A little modification of the feature might give better result. As in our development, we are working in news domain, the years are limited to 1900-2100 in most cases. Then we have modified the feature as if it is a four-digit word and its value is between 1900 and 2100 then the feature value is 1.

- *Numerical Word*

If a word is a numerical word, i.e. it is a word denoting a number (e.g. *tin* (three), *char* (four) etc.) then the feature *NumWord* is set to 1.

- *Word Suffix*

Suffix information is useful to identify the named entities. This feature can be used in two ways. The first and naive one is that a fixed length word suffix of current and surrounding words can be treated as feature. During evaluation, it was observed that this feature is useful and able to increase the accuracy by a considerable amount. Still, better approach is to use suffix based binary feature. Variable length suffixes of a word can be matched with predefined lists of useful suffixes for different classes of NEs. Suffix list of locations is very useful since most of the location names in India end with a specific list of suffixes. Suffix list of locations contains 116 suffixes like, *bad, pur, puram, ganj, dihi* etc.

- *Word Prefix*

Prefix information of a word is also useful. A fixed length word prefix of current and surrounding words can be treated as feature.

- *Parts-of-Speech (POS) Information*

The POS of the current word and the surrounding words are important to recognize names. For this task we have used the POS tagger developed at IIT Kharagpur, India. The tagset of the tagger contains 28 tags. Firstly we have used the POS values of current and surrounding tokens as feature.

All 28 POS tags are not helpful in recognizing names. Nominal and postpositional tags are the most important in name finding in Hindi. Then we have modified the POS tagger to a coarse-grained POS tagger which has only three tags - nominal, postpositional (PSP) and others. These coarse grained POS values of current and surrounding tokens are more helpful for name recognition.

The POS information is also used in another way. Some binary features are defined using the POS information. For example, a binary feature *NominalPSP* is defined as following, if the current token is nominal and the next token is a PSP then the feature is set to 1, otherwise 0.

B. *Maximum Entropy Based Model*

MaxEnt is a flexible statistical model which assigns an output for each token based on its history and features. MaxEnt computes the probability *p(o|h)* for any *o* from the space of all possible outputs *O*, and for every *h* from the space of all possible histories *H*. A history is all the conditioning data that enables to assign probabilities to the space of output. In NER, history can be viewed as all information derivable from the training corpus relative to the current token w_i. The computation of *p(o|h)* depends on a set of features, which are helpful in making predictions about the output.

Given a set of features and a training corpus, the MaxEnt estimation process produces a model in which every feature f_i has a weight α_i. We can compute the conditional probability as [15]

$$p(o \mid h) = \frac{1}{Z(h)} \prod_i \alpha_i^{f_i(h,o)} \qquad (1)$$

$$Z(h) = \sum_O \prod_i \alpha_i^{f_i(h,o)} \qquad (2)$$

The probability is given by multiplying the weights of active features. The weight α_i is estimated by a procedure called Generalized Iterative Scaling (GIS) [6]. This method improves the estimation of weights iteratively. The MaxEnt estimation technique guarantees that, for every feature f_i, the expected value equals the empirical expectation in the training corpus.

For our development we have used a Java based open nlp MaxEnt toolkit[1] to get the probability values of a word belonging to each class. That is, given a sequence of words, the probability of each class is obtained for each word. To find the most probable tag corresponding to each word of a sequence, we can choose the tag having the highest class-conditional probability value.

Sometimes this method results in inadmissible assignment for tags belonging to the sequences that never happen. To eliminate these inadmissible sequences we have made some restrictions. Then we have used a beam search algorithm with beam length 3 with these restrictions. This algorithm finds the most probable tag sequence from the class conditional probability values.

C. *Training Data*

The training data used for this task contains of about *243 K* words with *16,482* NEs, which is collected from the popular daily Hindi newspaper "Dainik Jagaran". In this development, we have considered four types of NEs to recognize. These are *Person* (Per), *Location* (Loc), *Organization* (Org) and *Date*. To recognize entity boundaries, each name class *N* is subdivided into four sub-classes, i.e., *N_Begin*, *N_Continue*, *N_End*, and *N_Unique*. Hence, there are total *17* classes (4 name classes × 4 sub-classes + 1 not-name class). The corpus contains *6,298* Person, *4,696* Location, *3,652* Organization and *1,845* Date entities.

D. *Evaluation*

About 80 different experiments are conducted taking several combinations from the mentioned features to identify the best feature set for the NER task. We have evaluated the system using a blind test file of size 25 K words, which is totally different from the training file. The accuracies are measured in terms of F-measure, which is weighted harmonic mean of precision and recall. Precision is the percentage of the correct annotations and recall is the percentage of the total named entities that are successfully annotated. The general expression for measuring the F-value is: $F_\beta = ((1 + \beta^2)(precision \times recall)) / (\beta^2 \times precision + recall)$. Here the value of β is taken as 1.

First of all, we have used only the current and surrounding words as feature of MaxEnt. We have experimented with several combinations of previous 4 to next 4 words ($w_{i-4}...w_{i+4}$) to identify the best word-window. The results are shown in Table I.

TABLE I.
RESULTS (F-MEASURE) OF MAXENT BASED SYSTEM USING WORD FEATURES

Feature	Per	Loc	Org	Date	Total
w_i, w_{i-1}, w_{i+1}	61.36	68.29	52.12	88.9	67.26
w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}	64.10	67.81	58	92.30	**69.09**
w_i, w_{i-1}, w_{i-2}, w_{i-3}, w_{i+1}, w_{i+2}, w_{i+3}	60.42	67.81	51.48	90.18	66.84
w_i, w_{i-1}, w_{i-2}, w_{i-3}, w_{i-4}, w_{i+1}, w_{i+2}, w_{i+3}, w_{i+4}	58.42	64.12	47.97	84.69	61.27
w_i, w_{i-1}inList, w_{i-2}inList, w_{i+1}inList, w_{i+2}inList	65.37	70.33	47.37	83.72	66.17

From Table I we can observe that word window ($w_{i-2}...w_{i+2}$) gives the best result. When the window size is increased, the performance degrades. List based binary word features are not effective. In the table, the notation *w_{i-n}inList* is used to indicate binary word features for all classes for w_{i-n}. We have already mentioned that the binary word feature matches the word if it presents in a frequent word list which is formed from the training corpus. By analyzing the word lists we have observed that the lists do not contain all the words related to a class. For example, the word *'jakar'* (going to) in the next position helps to conclude that the current word has high probability to be a location name. But the word is *'jakar'* is not in the corresponding list because the word is not occurring in that particular position with high frequency in our training corpus. Manual editing of the lists might help the binary word feature to perform better.

Similar experiments are conducted to find the best feature set for the Hindi NER task. The features described earlier are applied separately or in combination to build the MaxEnt

[1] www.maxent.sourceforge.net

based model. In Table II we have summarized the results. Only the best values of each feature category are given in the table. This result is considered as the *baseline* in this study.

TABLE II.
RESULTS OF MAXENT BASED SYSTEM USING DIFFERENT FEATURES

Feature	***Per***	***Loc***	***Org***	***Date***	***Total***
words, previous NE tags	63.33	69.56	58.58	91.76	69.64
words, tags, prefix(≤4)	66.67	71	58.58	87.8	70.02
words, tags, suffix(≤4)	70	76.92	59.18	88.9	73.5
words, tags, suffix (≤4), prefix(≤4)	70.44	70.33	59.18	90.18	72.64
words, tags, digit information	62.94	69.56	50	91.76	67.63
words, tags, suffix (≤4), digit	70.44	76.92	60.44	93.02	74.51
words, tags, POS (28 tags)	66.67	72.84	60	88.9	71.22
words, tags, POS(coarse-grained)	69.62	80.74	58.7	91.76	75.22
words, tags, POS(coarse-grained), suffix (≤4), digit	72.23	78.1	62.37	93.02	75.67
words, tags, 'nominalPSP', suffix (≤4), digit	72.5	80.74	58.7	93.02	**75.89**

From the table we can observe that some of the features are able to improve the system accuracy separately, but when applied in combination with other features, they cause decreasing of the the accuracy. For example, with the information about the word and tag only we achieve F-value of 69.64. When suffix information is added, F-value is increased to 73.5 and when prefix information is added then F-value of 70.02 is achieved. But when both the suffix and prefix features are combined, then the F-value is 72.64. Prefix information increases the accuracy alone, but when combined with suffix information, it decreases the accuracy instead of increasing it. More complex features do not guarantee the better result. The best accuracy of the system is the F-value of 75.89, which is obtained by using current word, surrounding words (w_{i-1}, w_{i+1}), previous NE tags, suffix information (≤4), digit information (contains digit, four digit, numerical word) and the POS based binary feature *nominalPSP*. Here an interesting observation is, that the best feature set uses the word window (-1 +1), i.e. one previous word and one next word. Using the wider window reduces the performance, though in Table I it was found that window (-2 +2) performs best.

IV. GAZETTEER INFORMATION

Gazetteer lists or name dictionaries are helpful in NER. It is observed that a huge number of organization names end with some specific words like *Inc.*, *Corp.*, *Limited* etc. If all such words can be collected in a list then they can help to recognize the organization names. Again, it is very common that some designations like *prof.*, *minister* etc. and some other qualifiers like *Mr.*, *Dr.*, *Sri* etc. appear before the name of a person. A list containing all such words helps in person name identification. A surname list is also helpful for identifying person names. Similarly location list, organization list, first name list etc. are some helpful gazetteer lists.

Gazetteer lists are successfully used in many English NER systems. Borthwick's 'MENE' has used 8 dictionaries [3], which are: First names (1,245), Corporate names (10,300), Corporate names without suffix (10,300), Colleges and Universities (1,225), Corporate suffixes (244), Date and Time (51) etc. The numbers in parentheses indicate the size of the corresponding dictionaries. As another example, we can mention the hybrid system developed by Srihari *et al*. (2000). The gazetteer lists used in the system are: First name (8,000), Family name (14,000) and a large gazetteer of Locations (250,000). There are many other systems which have used name dictionaries to improve the accuracy.

Being influenced by these systems, we have decided to use gazetteer lists in our system. We have planned to use a few gazetteer lists like, person prefix, corporate suffix, surname, first name, location etc.

Initially we have attempted to prepare the gazetteers from the training corpus. Comparing with similar English dictionaries, it seems that prepared dictionaries might be sufficient for person prefix words, organization suffix words etc. but person first name list, location list etc. are not sufficient for the Hindi NER task. Then we have attempted to use the web sources for creating large gazetteer lists.

As our goal is to develop a NER system for Hindi, we are mainly interested in preparing gazetteers, which will contain mainly places in India, Indian first names and Indian surnames. For that purpose, we have collected the names from several websites. Mainly we have explored some Indian baby name websites to prepare the first name list. Also a lot of names of non-Indian famous personalities who are likely to appear in Indian news, collected from several sources, are added to the first name list. Similarly, we have prepared the location dictionary using Indian telephone directory, postal websites and the web encyclopedia 'wikipedia'. In Table III, we have mentioned the main sources from which we have collected the names.

TABLE III.
SOURCES OF GAZETTEER LISTS

Gazetteer	Sources
First name	http://hiren.info/indian-baby-names http://indiaexpress.com/specials/babynames http://www.modernindianbabynames.com/
Surname	http://surnamedirectory.com/surname-index.html http://en.wikipedia.org/wiki/Indian_name http://en.wikipedia.org/wiki/List_of_most_common_surnames
Location	http://indiavilas.com/indiainfo/pincodes.asp http://indiapost.gov.in http://maxmind.com/app/worldcities http://en.wikipedia.org/wiki

A. Transliteration

The transliteration from English to Hindi is very difficult. English alphabet contains 26 characters whereas the Hindi alphabet contains 52 characters. So the mapping is not trivial. We have already mentioned that Ekbal *et al*. [7] has developed a transliteration system for Bengali. A similar approach can be used to develop a Hindi-English transliteration system. But it requires a bilingual transliteration corpus, which needs huge efforts to built, is unavailable to us. Also using this approach the word agreement ratio obtained is below 70%, which is not a good value for the task.

To make the transliteration process easier and more accurate, we propose a 2-phase transliteration module. As our goal is to make decision that a particular Hindi string is in English gazetteer or not, we need not transliterate the Hindi strings in English or English strings into Hindi. Our idea is to define an intermediate alphabet. Both the English and Hindi strings will be transliterated to the intermediate alphabet. For two English-Hindi string pair, if the intermediate alphabet is same then we can conclude that one string is the transliteration of the other.

First of all we need to decide the alphabet size of the intermediate state. When several persons write a Hindi name in English, all the English string may not be same. For example a Hindi name "*surabhii*" when written in English, may be written as several ways, like *surabhi, shurabhi, suravi, suravee, shuravi* etc. So, it is very difficult to transliterate properly. Preserving the phonetic properties we have defined our intermediate alphabet consisting of 34 characters. To indicate these 34 characters, we have given unique character-id to each character which ranges from 51# to 84#. As special characters and digits are very rare in person and location names, all the special characters are mapped to a single character with character-id 99# and all the digits are mapped to 98#.

B. *English to Intermediate Alphabet Transliteration*

For transliterating English strings into the intermediate alphabet, we have built a phonetic map table. This map table maps an English n-gram into an intermediate character. A few entities of the map table are shown in Table IV.

TABLE IV.
A PART OF THE MAP-TABLE

English	Intermediate	English	Intermediate
A	51#	EE, I	53#
OO, U	54#	B, W	55#
BH, V	56#	CH	57#
R, RH	76#	SH, S	77#

The procedure of transliteration is as follows.

Procedure 1: Transliteration English-Intermediate
Source string – English, Output String – Intermediate.

```
1. Scan the source string (S) from left to right.
2. Extract the first n-gram (G) from S. (n = 4)
3. Search it in the map-table.
4. If it is found, insert its corresponding intermediate
   state entity (I) into target string M. M → M + I.
   Remove G from S. S → S – G.
   Go to step 2.
5. Else, set n = n – 1.
   Go to step 3.
```

Using this procedure, English string '*surabhii'* will be transliterated to 77#54#76#51#56#53#. If we check the transliteration for '*shuravi'*, it is transliterated into intermediate string in the same manner.

C. *Hindi to Intermediate Alphabet Transliteration*

This is done in two steps. At the first step, the Hindi strings (which are in Unicode) are transliterated into *itrans*. *Itrans* is representation of Indian language alphabets in terms of ASCII. Since Indian text is composed of syllabic units rather than individual alphabetic letters, *itrans* uses combinations of two or more letters of English alphabet to represent an Indian language syllable. However, there are multiple sounds in Indian languages corresponding to the same English letter and not all Indian syllables can be represented by logical combinations of English alphabet. Hence, *itrans* uses some non-alphabetic special characters also in some of the syllables. The difficulty in converting the Unicode Hindi string to *itrans* is that the conversion mapping of Unicode to *itrans* is many to one. A map table[2], with some heuristic knowledge, is used for the transliteration. Our example Hindi word '*surabhii'* is converted into '*sUrabhI'* in *itrans*.

At the next step, the *itrans* string is transliterated into the intermediate alphabet using a similar procedure of transliteration. Here we use a similar map-table containing the mappings from *itrans* to intermediate alphabet. This procedure will transliterate the example *itrans* word '*sUrabhI'* to 77#54#76#51#56#53#.

D. *Accuracy of the Transliteration System*

The transliteration system is evaluated by using a bilingual corpus containing 1,070 English-Hindi word pairs most of which are names. 980 of them are transliterated correctly by the system. So, the system accuracy is 980×100/1070 = 91.59%.

This transliteration approach is applicable for some other languages also.

V. USE OF GAZETTEER LISTS IN MAXENT BASED HINDI NER

We have prepared the gazetteer lists directly from the corpus or from the web using the transliteration process discussed in the above section. The lists collected from the web are transliterated and stored in the intermediate form. One way of using the gazetteer information is to directly search a token if it is in the list. If it is present then we make the decision that the word belongs to that particular class. But this cannot resolve ambiguity as a particular token may present in more than one list and confusion arises. We have used the gazetteer information as a feature of MaxEnt. In the following we have described the prepared gazetteer lists and the corresponding features in details.

[2] www.aczoom.com/itrans

A. Gazetteer Lists

– *Month name, Days of the Week*

If the word is one of *January, February, . . ., December*, (*baishakh, jyashtha, . . ., chaitra* (month names of Hindi calendar)), then the feature *MonthName* is set to 1. If it is one of *Monday, Tuesday*, . . ., *Sunday (sombar, mangalbar, . . ., rabibar*,..) then the feature *DayWeek* is set to 1.

– *Corporate Suffix list*

Corporate Suffix List (CSL) contains most frequently occurring last words of organization names collected from the training data. CSL is made up of *limited*, *corp*., *inc*, *institute*, *university* etc. The size of the list is 92 entries. For a word w_i, if any of the words from w_{i+1} to w_{i+n} is in CSL, then a feature *CorpSuf* is set to 1.

– *Person Prefix List*

It contains the designations and qualifiers that occur before person names and are collected from the training data. Examples of some prefixes are, *sri* (Mr.), *kumari* (mrs.), *mantri* (minister), *adhyaksha* (chairman) etc. The list contains 123 prefixes.

Note that person prefix words are not the part of the person names, while corporate suffixes are part of the organization names. For a word w_i, if any of the words from w_{i-m} to w_{i-1} is in person prefix List, then a feature *PerPref* is set to 1.

– *Common Location*

This list contains the words denoting common locations. Common location words like *jila* (district), *nagar* (town/city), *roda* (road) etc. have high probability to occur at the end of a location name. 70 such words are collected in the Common Location List (CLL). Then the binary feature *ComLoc* is defined as, it takes value 1 for a word w_i if its next word presents in CLL.

– *Location List*

17,600 location names are gathered in the Location List (LL). LL is converted using the transliteration and stored in intermediate form. LL is processed into a list of unigrams (e.g., *Kolkata, Japan*) and bigrams (e.g., *New Delhi, New York*). The words are matched with unigrams and sequences of two consecutive words are matched against bigrams to get the feature value of the binary *LocList* feature.

– *First Name List*

This list contains 9,722 first names collected from the web. Most of the first names are of Indian origin. The feature *FirstName* is defined as, if the word w_i is in the list, then the feature is set to 1, otherwise 0.

– *Middle Name List*

A list is compiled containing the common middle names in India, for example, *kumar, chandra, nath, kanta* etc. This list contains 35 entries.

– *Surname List*

This is a very important list which contains surnames. As our objective is to develop a Hindi NER, we are most interested in Indian surnames. We have prepared the Surname List (SL) from different sources containing about 1,500 Indian surnames and 200 other surnames. A binary feature *SurName* is defined according to whether the word is in SL.

B. Evaluation

In Table V, we have shown the results of the NER system after incorporating the gazetteer lists. To observe the effectiveness of the prepared gazetteer lists in Hindi NER, we have added the lists with the baseline system.

TABLE V.
RESULTS OF MAXENT BASED SYSTEM USING GAZETTEER LISTS

Feature	Per	Loc	Org	Date	Total
Baseline: words, tags, suffix (≤4)	70	76.92	59.18	88.9	73.5
words, tags, suffix, CorpSuf	70	78.1	72.3	88.9	76.92
words, tags, suffix, DayWeek, monthName,	70	76.92	59.18	95.83	74.16
words, tags, suffix, PersonPrefix	72.5	76.92	59.18	88.9	74.09
words, tags, suffix, SurName, PerPref, FirstName, MidleName	77.2	78.1	59.18	88.9	76.34
words, tags, suffix, LocList, ComLoc	70	82.81	61.05	88.9	75.41
words, tags, suffix, all gazetteers	75.86	81.29	74.8	95.83	80.2
Baseline: words, tags, nominalPSP, suffix (≤4), digit	72.5	80.74	58.7	93.02	75.89
words, tags, nominalPSP, suffix, digit, all gazetteers	77.2	82.81	76.35	95.83	**81.12**

To observe the changes in accuracy, we have selected two feature sets from the baseline system (as in Table II): {current word, surrounding words, previous NE tags, suffix≤4} and {current word, surrounding words, previous NE tags, suffix≤4, digit information, nominal PSP}. The first feature set achieves F-value of 73.5 and the second one achieves F-value of 75.89, which is the best baseline feature set.

After adding the gazetteer lists, F-value has increased to 80.2 for the first feature set and 81.12 for the second. Also from the table we observe that the addition of a gazetteer list for a particular class (C_j) mostly increases the accuracy of C_j. For example, when the person gazetteer lists (e.g. person prefix list, surname list, first name list etc.) are incorporated, F-value of the person class has increased to 77.2 from 70. Change in accuracy of the other classes is minor. The highest F-value achieved by the developed Hindi NER system is 81.12.

VI. EXPERIMENTS ON BENGALI NER

The proposed two-phase transliteration approach is used successfully to make the English gazetteer lists useful in the Hindi NER task. The proposed approach is also applicable to other resource-poor languages. To study the effectiveness of the approach in another language we have chosen Bengali. As Hindi and Bengali alphabets are very similar, we needed a little effort to transfer the transliteration module from Hindi to Bengali.

Our primary objective is not to develop a 'good' Bengali NER system, but to experiment the effectiveness of the transliteration approach in Bengali NER task. We first developed a Bengali NER system using a small training corpus which is used as baseline. Then the transliteration module is modified to make the collected English gazetteer lists useful for the Bengali. These gazetteer lists are incorporated in the system and the improvement in accuracy is observed.

A. *Training Corpus*

The training corpus used for the Bengali NER task is much smaller than the Hindi corpus. The corpus contains only 68 K words. Three named entity classes are considered: Person, Location and Organization. The corpus contains 1,240 person names, 1,475 location names and 490 organization names.

B. *Transliteration Module*

Collected English gazetteer lists are then transliterated into Bengali. English to intermediate alphabet transliteration of the gazetteer lists is already done during the experiments in Hindi. Using a Bengali map-table, the Bengali words are transliterated to *itrans*. We have already mentioned that the alphabets of Bengali and Hindi are similar, so the Hindi module for the transliteration from *itrans* to intermediate is used for Bengali without any modification.

The accuracy of the Bengali transliteration is measured using a smaller bilingual test corpus containing 400 word pairs. The accuracy of transliteration for Bengali is 89.3%.

C. *Features for Bengali NER*

The feature set used for the Bengali NER development is mentioned in the following.

- Surrounding words (two previous and two next),
- NE tags of previous words,
- Affix information (all affixes up to a fixed length and list based),
- Root information of the words,
- POS information.

Most of the features are used in similar ways as used in the Hindi NER task. The feature *root information* is not used in Hindi NER development, but it is very important in Bengali NER. In Bengali, several affixes are often added to the names inflecting them. For example, a person name "*Sachin*" is inflected in Bengali as, *sachinra* (plural, the group in which *Sachin* belongs to), *sachiner* (*of Sachin*), *sachinke* (*to Sachin*), *sachinda (brother Sachin*), *sachinbabu* (*Mr. Sachin*) etc. As these affixes are added to the names, sometimes identification of inflected names becomes very difficult. To identify the inflected names we have extracted the 'root' information of the words and used them as features of MaxEnt. In Hindi, such affixes generally present separately from the names as 'postpositions', so root information is not much useful.

D. *Experimental Results*

MaxEnt classifier is used for the experiments. The training corpus and the mentioned features are used to develop the baseline system. The system is evaluated using a test corpus containing 10 K words. The baseline system has achieved the highest F-value of 62.81. After that the transliteration based gazetteer lists are incorporated. Then F-value of the system has increased to 69.59. The results are summarized in Table VI.

TABLE VI.
RESULTS OF THE BENGALI NER SYSTEM

Feature	Per	Loc	Org	Total
words, tags	56.9	56.13	56.67	56.55
words, tags, affix	58.01	59.05	57.28	58.24
words, tags, affix, root information	62.21	60.46	57.94	60.6
words, tags, affix, root, POS information	64.39	62.5	60.2	62.81
words, tags, affix, root, POS information, all gazetteers	70.42	69.85	67.58	**69.59**

VII. DISCUSSION

Named entity recognition is an important task. ML based approach for NER task requires sufficient annotated data to build the system. Gazetteer lists are often used to increase the performance of a NER system. For resource-rich languages, such resources are available, but for resource-poor languages these resources are scarce. Useful gazetteer lists are not available in these languages, though sometimes they are available in other languages (like English). If such lists are transliterated from other language into the target language, they become useful. We have proposed a two-phase transliteration methodology for the task.

Direct transliteration is difficult, so we have proposed a two-phase transliteration. Here an intermediate alphabet is defined. The strings from both languages (say, Hindi and English) are transliterated into the intermediate alphabet to make the decision that a string (Hindi) is in the gazetteer lists (English) or not. The main advantages of the proposed approach are:

- This is a character-gram mapping based (using map-tables) approach, where no training data (bilingual corpora) is required.
- The approach is very simple and fast.
- This is easily transferable to other language.
- The accuracy of transliteration is high.

The disadvantages of the approach are:

- The English strings are not transliterated to the target language. Here only the decision is taken whether a target word (Hindi) is in the English name list or not.
- The module is specially built for the NER task. It is not widely applicable to other NLP tasks.

The accuracy of transliteration is 91.59% for Hindi and 89.3% for Bengali. The major cases where the transliteration approach fails are, presence of homophones (pronounced similarly but one word is name but the other is not-name), word level changes (e.g., India is written as '*bharat*' in Indian languages, New Delhi as '*nayi dilli*'), dropping of internal vowels ('*surabhi*' is sometimes written/pronounced as '*surbhi*' – '*a*' is dropped) etc.

Suitable features are identified and MaxEnt is used to build the baseline NER systems for Hindi and Bengali using the identified features. Baseline accuracies for Hindi and Bengali are F-value of 75.89 and 62.81 respectively. A few gazetteer lists are collected from the web, which are in English, are incorporated in the system using the transliteration module and performance improvement is observed. F-values are increased to 81.12 for Hindi and 69.59 for Bengali. The accuracy for Bengali is much lower compared to Hindi because the training corpus size for Bengali is only 68 K words, whereas in Hindi the corpus contains 243 K words.

VIII. Conclusion

ML based approach requires annotated data and other resources to build a NER system. We have identified the suitable features for the Hindi NER task. We observed that some relevant gazetteer lists, which are very useful for improving the performance of the NER system, are available in English. To make the English name lists useful for Hindi, we have proposed a two-phase transliteration methodology. The available English gazetteer lists are used successfully in the Hindi NER system using the proposed transliteration approach. We have also examined the effectiveness of the transliteration approach on Bengali NER task.

Use of larger training data would increase the overall accuracy of the system. Also we hope that use of larger gazetteer lists will increase the accuracy of the system.

References

[1] Al-Onaizan Y. and Knight K. 2002. Machine Transliteration of Names in Arabic Text. In: *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages.*

[2] Bikel D. M., Miller S, Schwartz R and Weischedel R. 1997. Nymble: A high performance learning name-finder. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 194-201.

[3] Borthwick A. 1999. A Maximum Entropy Approach to Named Entity Recognition. *Ph.D. thesis, Computer Science Department, New York University*.

[4] Crego J. M., Marino J. B. and Gispert A. 2005. Reordered Search and Tuple Unfolding for Ngram-based SMT. In: *Proceedings of the MT-Summit X,* Phuket, Thailand, pp. 283-289.

[5] Cucerzan S. and Yarowsky D. 1999. Language independent named entity recognition combining morphological and contextual evidence. In: *Proceedings of the Joint SIGDAT Conference on EMNLP and VLC 1999,* pp. 90-99.

[6] Darroch J. N. and Ratcliff D. 1972. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, pp. 43(5):1470-1480.

[7] Ekbal A., Naskar S. and Bandyopadhyay S. 2006. A Modified Joint Source Channel Model for Transliteration. In *Proceedings of the COLING/ACL 2006*, Australia, pp. 191-198.

[8] Goto I., Kato N., Uratani N. and Ehara T. 2003. Transliteration considering Context Information based on the Maximum Entropy Method. In: *Proceeding of the MT-Summit IX,* New Orleans, USA, pp. 125–132.

[9] Grishman R. 1995. Where's the syntax? The New York University MUC-6 System. In: *Proceedings of the Sixth Message Understanding Conference*.

[10] Knight K. and Graehl J. 1998. Machine Transliteration. *Computational Linguistics,* 24(4): 599-612.

[11] Li H., Zhang M. and Su J. 2004. A Joint Source-Channel Model for Machine Transliteration. In: *Proceedings of the 42nd Annual Meeting of the ACL*, Barcelona, Spain, (2004), pp. 159-166.

[12] Li W. and McCallum A. 2003. Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction. In: *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3): 290–294.

[13] McDonald D. 1996. Internal and external evidence in the identification and semantic categorization of proper names. In: *B.Boguraev and J. Pustejovsky (eds), Corpus Processing for Lexical Acquisition*, pp. 21-39.

[14] Mikheev A, Grover C. and Moens M. 1998. Description of the LTG system used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference*.

[15] Pietra S. D., Pietra V. D. and Lafferty J. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 19(4): 380-393.

[16] Saha S. K., Mitra P. and Sarkar S. 2008. Word Clustering and Word Selection based Feature Reduction for MaxEnt based Hindi NER. In: *proceedings of ACL-08: HLT*, pp. 488-495.

[17] Srihari R., Niu C. and Li W. 2000. A Hybrid Approach for Named Entity and Sub-Type Tagging. In: *Proceedings of the sixth conference on applied natural language processing*.

[18] Wakao T., Gaizauskas R. and Wilks Y. 1996. Evaluation of an algorithm for the recognition and classification of proper names. In: *Proceedings of COLING-96*

Visualización 3D de Deformación y Corte de Objetos Virtuales basada en Descomposición Ortogonal

Gabriel Sepúlveda, Vicente Parra y Omar A. Domínguez

***Resumen*—Se muestra la implementación gráfica de un nuevo modelo para la renderización de fuerzas de contacto durante la interacción háptica dentro de ambientes virtuales 3D para deformación y corte de objetos virtuales con propiedades dinámicas y superficiales complejas. Se define un algoritmo simple para la triangulación de un objeto virtual, empleando algoritmos clásicos para la detección de colisiones. Se desarrolló un algoritmo para la visualización del corte de la malla triangular así como para el cálculo de las dinámicas de la malla durante el corte. Se presentan imágenes de la plataforma utilizando OpenGL y Visual C++, para la parte gráfica y el dispositivo Falcon para la retroalimentación háptica.**

***Palabras clave*—Dispositivo háptico, renderización 3D, deformación y corte.**

3D Visualization of Deformation and Cut of Virtual Objects based on Orthogonal Decomposition

***Abstract*—We present graphic implementation of a novel model for contact force rendering during the haptic interaction in 3D virtual environment. It is applied during cut and deformation of virtual objects with complex dynamic and surface properties. We define the simple algorithm for triangulation of the virtual object using the classic algorithm of detection of collisions. The algorithm is proposed that allows visualizing of the cutting triangular net as well as the calculation of the dynamics of the net during the cut. We present the corresponding images using OpenGL and Visual C++ for the graphic module and Falcon device for haptic feedback.**

***Index Terms*—Haptic device, 3D rendering, deformation and cut.**

I. Introducción

DURANTE la interacción con objetos virtuales a través de dispositivos hápticos, un aspecto importante a considerar es la renderización de las fuerzas de contacto, para ello se han desarrollado numerosas teorías y algoritmos dentro de los que destacan: el diseño de restricciones necesarias para la simulación de tejido [1], el uso de herramientas virtuales [2], el uso de elementos locales (LEM) [3]. Así como los algoritmos empleados para la generación de texturas como [4] y [5]. Pero estos algoritmos utilizados en la renderización háptica consideran relativamente poco la visualización de la tarea que se está llevando acabo, además de necesitarse por lo menos dos de ellos para generar simultáneamente propiedades dinámicas del objeto y textura en su superficie.

Es por lo anterior que este documento presenta un nuevo algoritmo para la renderización gráfica de objetos virtuales deformables y bajo corte, utilizando como base el algoritmo de descomposición ortogonal utilizado para la generación de fuerzas de contacto a través de un dispositivo háptico [6], este algoritmo permite la generación simultánea de propiedades dinámicas y propiedades superficiales para los objetos virtuales.

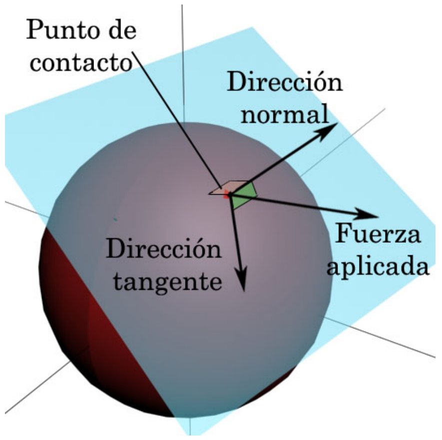


Fig. 1. Descomposición ortogonal de la fuerza aplicada a una superficie definida por una esfera en coordenadas articulares.

II. Descomposición ortogonal

El algoritmo de descomposición ortogonal presentado en [6] permite la descomposición de la dinámica del dispositivo háptico en dos dinámicas independientes y ortogonales, la primera se utiliza para generar propiedades dinámicas de los objetos virtuales, tales como deformación elástica lineal y de tipo exponencial [7], esta dinámica es empleada en la renderización de las fuerza de contacto, deformación y corte de los objetos virtuales. La segunda dinámica generada por la descomposición ortogonal se emplea en la generación de fuerzas de fricción simples (viscosa-Coulomb) o avanzadas (GMS) [8]. La primera dinámica es normal al punto de

Manuscrito recibido el 20 de agosto del 2008. Manuscrito aceptado para su publicación el 3 de noviembre del 2008.

Gabriel Sepúlveda, Departamento de Ingeniería Eléctrica, CINVESTAV, México DF (email: gsepulveda@cinvestav.mx).

Vicente Parra, Grupo de Robótica y Manufactura Avanzada, CINVESTAV, Saltillo, México (email: vparra@cinvestav.mx).

Omar A. Domínguez, Centro de Investigación en Tecnologías de Información y Sistemas, UAEH, Pachuca, México (email: omar_arturo@uaeh.edu.mx).

contacto con el objeto virtual y la segunda es tangente al punto de contacto como se muestra en la Fig. 1.

El algoritmo de descomposición ortogonal presentado en [6] permite generar fuerzas de deformación exponencial como las presentes durante la manipulación de tejido orgánico real [7], fuerzas presentes durante el corte de una muestra de tejido [9] con una gran fidelidad al utilizar dispositivos hápticos de altas prestaciones como el Phantom Premium 1.0 de la compañía *Sensable* y con una fidelidad moderada con dispositivos como el Falcon de la compañía *Novint*.

III. Mallado de la muestra virtual

Para realizar caracterizaciones biomecánicas se toman muestras de tejido y sobre ellas se realizan las pruebas, para nuestros experimentos dentro de un ambiente virtual 3D se generó una muestra virtual con forma de prisma triangular (ver Fig. 6). La muestra virtual se genera a partir de un triángulo definido por sus tres vértices, posteriormente se genera un mallado por subtriangulación iterativa a partir del primer triángulo siguiendo la base del algoritmo empleado para generar el fractal de Serpinsky, en el cual cada triángulo es subdividido en tres triángulos como se muestra en la Fig 2A y 2B (triángulos negros). El algoritmo implementado dentro de nuestra plataforma extiende el algoritmo anterior para generar el cuarto triángulo (el triángulo blanco del centro en la Fig. 2B, y así lograr una triangulación completa a partir del triángulo original.

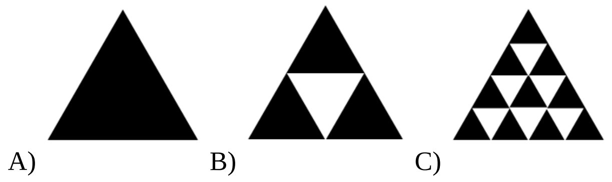


Fig. 2. Triangulación generada a partir de subtriangulación iterativa.

La muestra posee una característica denominada resolución, dicha característica permite definir el número de iteraciones que se realizará el algoritmo definiendo así la cantidad de triángulos de acuerdo a la Tabla 1.

TABLA I
CANTIDAD DE TRIÁNGULOS DE LA MALLA DE ACUERDO A LA RESOLUCIÓN DEL OBJETO VIRTUAL

Resolución	**Triángulos en la malla**
1	1
2	4
…	…
i	$4^{(i-1)}$

Dentro de la estructura de la muestra virtual se definen los vértices que forman el contorno del objeto, obteniendo así los puntos a partir de los cuales se generan las paredes del objeto, generando un mallado triangular para las paredes como se muestra en la Fig. 3, el numero de subdivisiones verticales depende de la resolución del objeto virtual.

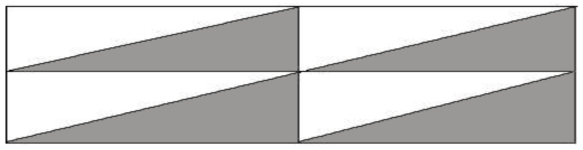


Fig. 3. Triangulación empleada en las paredes de la muestra virtual.

IV. Mallado inercial

Para generar la percepción visual de un movimiento continuo a lo largo de la superficie de la muestra virtual se generó un mallado inercial. Un mallado inercial es un conjunto de puntos a los cuales se les asigna una masa, interconectados por resortes y amortiguadores a lo largo de las aristas que los unen, como se muestra en la Fig. 4, lo anterior produce el efecto deseado: cuando un nodo de la malla es desplazado, el cálculo de la dinámica debida a los resortes y amortiguadores produce una fuerza sobre sus vecinos, la cual produce desplazamiento en ellos, por lo cual se genera el efecto que permite interpretar la malla como un cuerpo continuo y deformable.

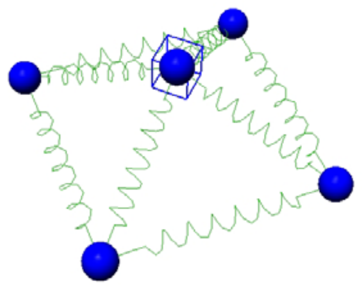


Fig. 4. Malla inercial, las esferas representan lo nodos del mallado, y las conexiones representan los resortes y amortiguadores que conectan a cada uno de los nodos.

La dinámica de la malla inercial es la sumatoria de las dinámicas de cada uno de *n* nodos de la malla. A cada *i*-ésimo nodo de la malla se le asigna una masa m_i y un conjunto de valores b_{ij} y k_{ij} que representan los valores del amortiguador y del resorte respectivamente entre el nodo i y el nodo *j* por último la variable F_i que representa la fuerza externa aplicada sobre el nodo *i*, esto se representa mediante la siguiente ecuación:

$$m_i \ddot{X}_i + \sum_{i=1}^{n} b_{ij}(\dot{X}_i - \dot{X}_j) + \sum_{i=1}^{n} k_{ij}(X_i - X_j - X_{ij0}) = F_i \quad (1)$$

donde $X_i, \dot{X}_i, \ddot{X}_i$ representa el desplazamiento, velocidad y aceleración del nodo *i* en coordenadas cartesianas respectivamente, X_{ij0} representa la longitud inicial del resorte que une el nodo *i* con el nodo *j*.

Para resolver la ecuación diferencial presentada en (1) se utilizó como integrador el algoritmo de Runge-Kuta de 4° orden.

V. Detección de colisiones y generación de fuerzas durante deformación

Para generar las fuerzas de contacto presentes durante la interacción utilizando un dispositivo háptico y a través del algoritmo de descomposición ortogonal como se muestra en [6] se requiere generar una ecuación implícita, la cual define una geometría con la cual se interactúa a través del dispositivo háptico, para el desarrollo de este trabajo se consideró la ecuación de un plano como la geometría de interacción descrito por la ecuación:

$$Ax + By + Cz + D = 0 \tag{2}$$

donde A, B, C, D son escalares constantes, x, y, z son las tres coordenadas cartesianas.

Para generar la geometría de interacción durante la manipulación de la muestra virtual, se requiere un algoritmo para la detección de colisiones. El algoritmo empleado para la detección de colisiones entre la herramienta virtual y el mallado es el conocido como colisión entre esfera y triángulo, este algoritmo se emplea situando pequeñas esferas a lo largo de la superficie de la herramienta que puede entrar en contacto contra el mallado, después se realiza una búsqueda exhaustiva entre los triángulos del mallado de la muestra y las esferas de la herramienta (conocidas como Proxy), cuando se detecta la colisión se procede de la siguiente forma:

1. Se realiza la detección de colisiones.
2. Se obtienen las coordenadas de los tres vértices del triángulo con el cual se generó la colisión.
3. A partir de los vértices se calcula la ecuación del plano al cual pertenece el triángulo.
4. Con la ecuación del plano se calcula la fuerza de interacción que será enviada al dispositivo háptico así como el desplazamiento debido al movimiento del operador.
5. Una vez calculado el desplazamiento del triángulo se realiza el cálculo de la dinámica de la malla de acuerdo al modelo inercial, logrando la difusión del movimiento a lo largo del mallado.

Los pasos mencionados se muestran en la Fig. 5. Es importante señalar que la fuerza calculada en el paso 4 es la fuerza de interacción normal al punto de contacto (ver Fig. 1) la cual se interpreta a través del dispositivo háptico como la dinámica del objeto virtual por ejemplo su elasticidad y su dinámica ante el corte.

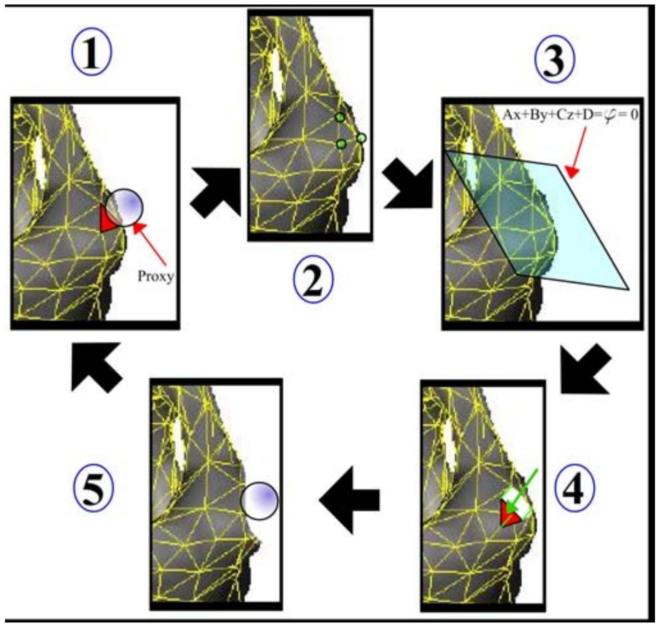


Fig. 5. Pasos para la detección de colisiones, renderización de fuerzas y deformación del mallado de la muestra virtual.

Durante la deformación de la muestra virtual debida a compresión la normal del plano apunta hacia afuera del objeto virtual, logrando con ello una fuerza en dirección contraria a la compresión percibiendo el operador rigidez o elasticidad del objeto virtual como se muestra en la Fig. 6.

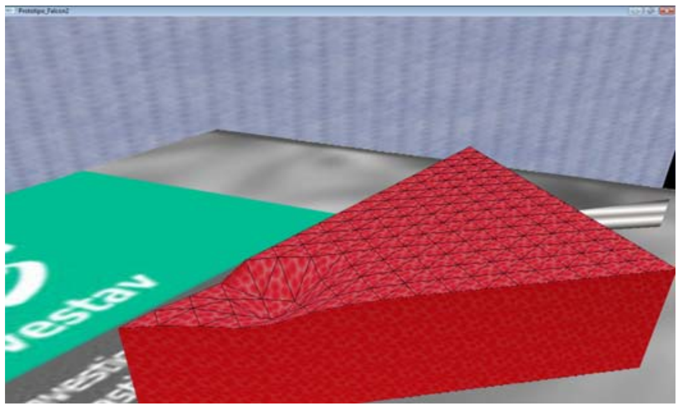


Fig. 6. Pantalla de la aplicación mostrando la compresión de la muestra virtual.

Durante la interacción en la cual se estira el mallado, lo que correspondería a tomar con unas pinzas la muestra virtual, la normal del plano de interacción apunta hacia adentro de la muestra para generar una fuerza que se oponga a la distensión, esto se muestra en la Fig. 7.

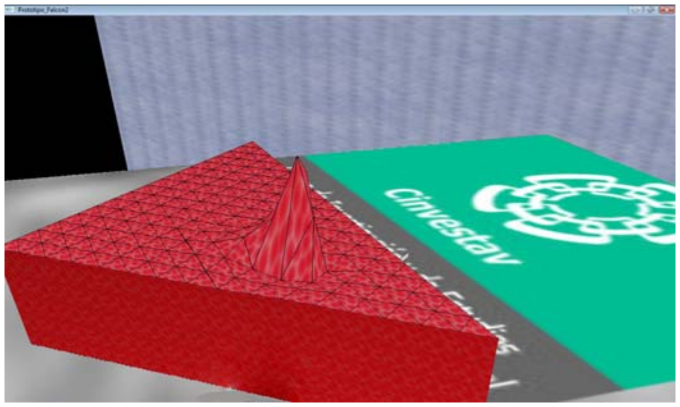


Fig. 7. Pantalla de la aplicación mostrando la distensión de la muestra virtual.

VI. Algoritmo para el corte

Para la renderización de las fuerzas durante el corte de la muestra virtual mediante el dispositivo háptico se emplearon los algoritmos descritos en [6], donde se muestra como ejemplo de corte la dinámica de una muestra de hígado de cerdo durante el corte. En [6] se generan las fuerzas de interacción pero no se presenta algún algoritmo que genere la misma retroalimentación de forma visual.

Para generar la visualización del corte se desarrollo un algoritmo que toma en cuenta la estructura triangular de la muestra virtual (ver Fig. 8) permitiendo cortes en dirección vertical ascendente (paralelos al eje *y*). El algoritmo consta de los siguientes pasos:

1. Se detecta el desplazamiento del dispositivo háptico durante la deformación de la muestra virtual hasta que se alcanza el punto de ruptura del tejido, con lo cual se cambia del modo de interacción de deformación a modo de interacción en corte.
2. Una vez en modo de interacción de corte se calcula de acuerdo al punto de contacto con la malla y a la dirección del desplazamiento una línea que define la dirección del corte Fig. 8 A.
3. Se realiza un cálculo del número de triángulos que tengan dos o más vértices a la derecha de la línea de corte, es decir que su coordenada *x* siempre esté a la derecha de la línea de corte. Dichos triángulos serán eliminados del mallado de la muestra virtual, como se muestra en la Fig. 8 B.
4. Una vez eliminados los triángulos del mallado se procede a recalcular el contorno del objeto para generar las nuevas paredes del objeto.

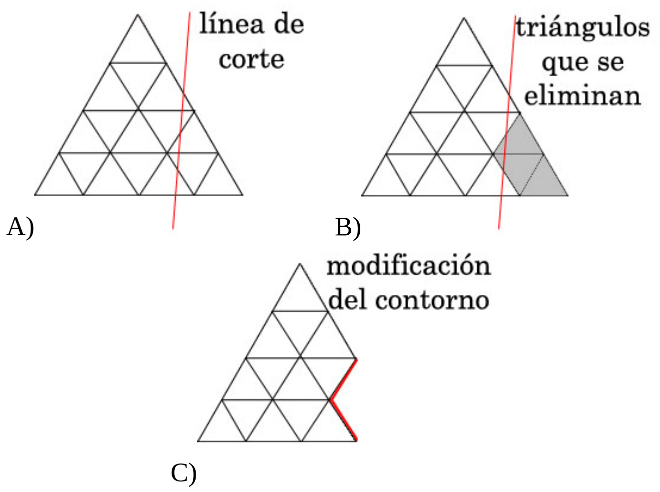


Fig. 8. Pasos realizados por el algoritmo para la visualización del corte de una muestra virtual.

El conjunto de nodos de la nueva muestra es reducido cambiando los elementos empleados en (1), al eliminar conexiones de resortes y amortiguadores. Los nodos, caras y paredes eliminados son removidos de la escena virtual. En la Fig. 9 se muestra una pantalla de la aplicación desarrollada donde se ha realizado un corte de la muestra virtual.

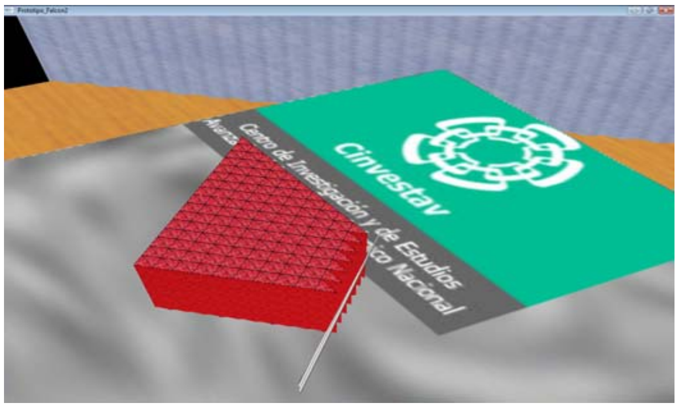


Fig. 9 Pantalla que muestra la aplicación después de haber realizado un corte.

VII. Plataforma experimental

La plataforma experimental empleada consta de una computadora portátil o laptop HP Pavillion dv2000 con procesador Intel Core2 Duo a una velocidad de 1.5 GHz con 2 Gb de memoria RAM y una tarjeta de video Mobile Intel 965 con un monitor LCD de 14 pulgadas. El sistema operativo es Windows Vista Ultimate. El ambiente de programación empleado fue OpenGL y sus librerías de herramientas GLU para la renderización del ambiente virtual 3D. Visual C++ 2007 como lenguaje de programación para la lógica del programa, los modelos matemáticos, los integradores empleados y las funciones de escritura de archivos.

El dispositivo háptico empleado en esta plataforma experimental fue el Falcon de la compañía Novint. La plataforma experimental se muestra en la Fig. 10.

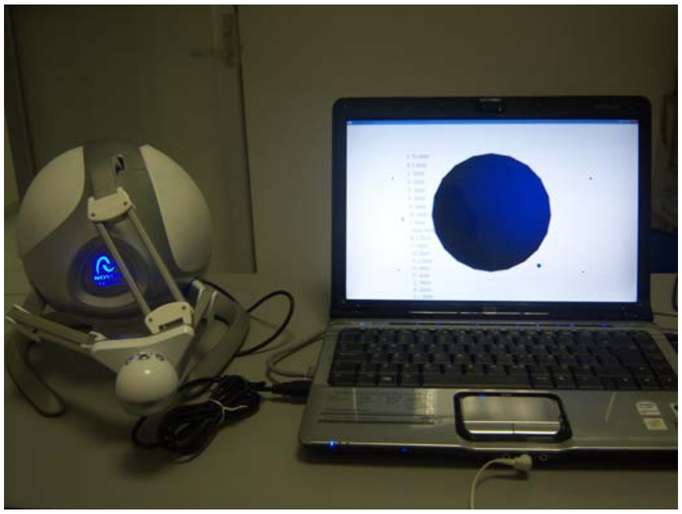


Fig. 10. Plataforma experimental, dispositivo háptico y computadora.

VIII. Conclusiones

Se presentó un algoritmo para la generación de muestras virtuales de tejido a partir de una triangulación iterativa. El algoritmo permite la renderización de fuerzas de contacto mediante un dispositivo háptico así como la visualización durante interacción en deformación y corte de la muestra. El algoritmo aporta un método para generar las geometrías necesarias para emplear el algoritmo para la renderización háptica de fuerzas basado en la descomposición ortogonal. El algoritmo permite únicamente cortes verticales de la muestra virtual, por lo que debe ser modificado para permitir cortes en cualquier dirección. El algoritmo aquí presentado se puede implementar en aplicaciones hápticas para la simulación de cirugía, entrenadores hápticos para estudiantes de medicina, así como para la interacción en mundos virtuales dinámicos.

Referencias

[1] Oliver R. Astley and Vincent Hayward. Design constraint for haptic surgery simulation. IEEE ICRA, 2000 San Francisco CA.

[2] Colgate, J. E., Stanley, M. C., and Brown, J. M. Issues in the Haptic Display of Tool Use. In: IEEE/RSJ, Proceedings of International Conference on Intelligent Robots and Systems, Pittsburgh, 1995, pp. 140-145.

[3] Teeranoot Chanthasopeephan Jaydev P. Desai and Alan C. W. Lau. 3D and 2D Finite Element Analysis in Soft Tissue Cutting for Haptic Display. In: Proceedings of ICAR, 2005, pp. 360-367.

[4] Omar Arturo Domínguez Ramírez. Design and integration of a Realistic Haptic Interface. CINVESTAV, Mechatronic Section, PhD Thesis, México, 2005.

[5] Gianni Campion and Vincent Hayward. Fundamental Limits in the Rendering of Virtual Haptic Textures. McGill Univerity, Montreal, Canada.

[6] Gabriel Sepúlveda, Vicente Parra, Omar Domínguez. Noninear Haptic Rendering of Deformation and Cutting Based on Orthogonal Decomposition. Journal “Research in Computing Science”, Vol. 35, 2008, pp. 51-61.

[7] Iman Brouwer *et al*. Measuring In Vivo Animal Soft Tissue Properties for Haptic Modeling in Surgical Simulation. University of California, USA, 2001.

[8] Farid Al-Bender, Vicent Lampaert and Jan Swevers. The Generalized Maxwell-Slip Model: A Novel Model for Friction Simulation and Compensation. IEEE Transaction on Automatic Control, Vol. 50, 2005, pp. 11.

[9] Mohsen Mahvash and Vincent Hayward. Haptic Rendering of Cutting: A Fracture Mechanics Approach. McGill University, Quebec, Canada, 2001.

An Extended Video Database Model for Supporting Finer-Grained Multi-Policy and Multi-Level Access Controls

Nguyen Anh Thy Tran and Tran Khanh Dang

***Abstract*—The growing amount of multimedia data available to the average user has reached a critical phase, where methods for indexing, searching, and efficient retrieval are needed to manage the information overload. Many research works related to this field have been conducted within the last few decades and consequently, some video database models have been proposed. Most of the modern video database models make use of hierarchical structures to organize huge amount of videos to support video retrieval efficiently. Even now, among open research issues, video database access control is still an interesting research area with many proposed models. In this paper, we present a hybrid video database model which is a combination of the hierarchical video database model and annotations. In particular, we extend the original hierarchical indexing mechanism to add frames and salient objects at the lowest granularity level in the video tree with the aim to support multi-level access control. Also, we give users more solutions to query for videos based on the video contents using annotations. In addition, we also suggest the original database access control model to fit the characteristics of video data. Our modified model supports both multiple access control policies, meaning that a user may be affected by multiple polices, and multi-level access control, meaning that an authorization may be specified at any video level. Theoretical analyses and experimental results with real datasets are presented that confirm the correctness and efficiency of our approach.**

***Index Terms*—Video database security, video database model, content-based video retrieval, access control, multimedia database.**

I. Introduction

THE field of multimedia systems has experienced an extraordinary growth during the last decade. Among many visible aspects of the increasing interest in this area is the creation of huge digital libraries accessible to users worldwide. These large and complex multimedia databases must store all types of multimedia data, e.g., text, images, animations, graphs, drawings, audio, and video clips. Video information plays a central role in such systems, and consequently, the design and implementation of video database systems have become a major topic of interest.

With the huge amount of video information stored in archives worldwide, video databases have been researched for many years to introduce efficient ways to manage this kind of data. Below are some criteria that a video database should satisfy:

– The first thing that should be satisfied is how to organize efficiently raw video data. Videos are gathered from various sources with different formats so they need to be normalized to a standard form before being stored. In addition, these videos should also be compressed to reduce storage space because of their inherently huge sizes. Furthermore, video database also extracts key features such as key frames, salient objects, etc. to achieve high performance video content-based retrieval.

– Secondly, the video database access control scheme should be integrated with the database indexing structure in order that video database access control can be achieved more effectively. Since video database access control schemes should exploit semantic visual concepts and not low-level visual features, these database indexing units should correspond to the relevant semantic visual concepts.

– Thirdly, the flexibility and efficiency of transmitting video data through networks are an important consideration because most video databases are deployed over network environments.

– Finally, control over the security of a video database system is important. Videos can be illegally accessed while being transferred over the network, or accessed directly into the database. This is vital for the important video databases such as national video data stores.

To achieve the above requirements, this paper proposes a video database system that supports content-based retrieval and multi-level access control with different policies. This video database system is illustrated in Fig. 1. It contains two main components, *video analyzing* module and *query processing* module.

As can be seen in Fig. 1, the videos come from various sources with different formats so they firstly need to be analyzed. This step consists of three major tasks:

– Partitioning video into several video shot (shot boundary detection), extracting key frames and salient

Manuscript received May 11, 2008. Manuscript accepted for publication October 22, 2008.

This work was supported in part by Advances in Security & Information Systems (ASIS) Lab, Faculty of Computer Science & Engineering, HCMUT, Vietnam.

N. A. T. Tran is with the KMS Company, Ho Chi Minh City, Vietnam (e-mail: thytran@kms.com.vn).

T. K. Dang is with the Faculty of Computer Science & Engineering, HCMC University of Technology, VNUHCM, Ho Chi Minh City, Vietnam (phone:+84-98-3173334, e-mail: khanh@cse.hcmut.edu.vn).

objects of each shot (key-frame and salient object extraction).

– Classifying and clustering video shots.
– Indexing video database using semantic clusters.

These tasks are handled by the *video analyzing* component while the *query processing* component is responsible for controlling access to the database. The access control model should be flexible and reliable. This means that users should have multiple methods to retrieve the desired videos but they should only be able to access those to which they have been authorized.

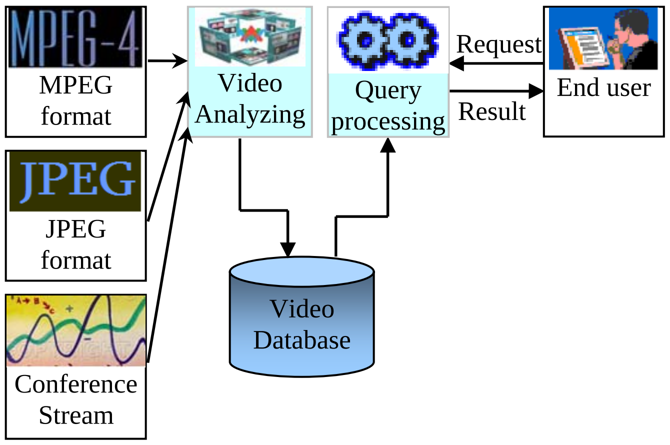


Fig. 1. A video database system structure.

The rest of this paper is organized as follows: In Section II, we briefly introduce the most crucial related work. In section III, after basics of video data processing is presented, we introduce a new hierarchical video database schema with more granularity levels. Section IV introduces the proposed algorithms to manage finer-grained access controls to the proposed video database. In section V, we present and discuss the implementation of a system prototype and experimental results of our proposed video database and access control model. Finally, section VI gives concluding remarks and introduces future work.

II. Related Work

Several efforts have been made to construct video database models to achieve flexible query and reliable access control. Basically, such efforts include the common data model for video information and hierarchical video data models [11], [2]. The first model did support content-based query using annotations, but was not suitable for large video databases since its structure was "flat", meaning every video was at the same level. This restriction led to a problem in that users could not browse and navigate to find desired videos. In contrast, the hierarchical model proposed by Bernito *et al.* organized videos into semantic clusters within a tree structure. This helped to resolve the semantic gap between the low-level visual features and the high-level semantic visual concepts. However, this model lacked annotations and so although browsing requirements can be satisfied, retrieval options were not flexible.

With regards to video database access control, although there are some proposed models that support multi-level video access controls, on the whole, they do not allow users to specify authorizations at frame and object levels [1], [2]. In [2] and [3], Bernito *et al.* suggested a mechanism to manage access control to hierarchical video objects and another method to support multiple access control policies. However, combining them into a unique model is not a simple task because of authorization conflicts. Consequently, in this paper, we propose a full-fledged model that supports both a multi-policy and multi-level access control mechanism.

III. Video Database Models

In this section, after major video processing steps are presented, we introduce an extended storage model for video data that supports both semantic visual concepts clustering and flexible content-based retrieval. This newly introduced video database model can serve as a solid basis for materializing flexible access control mechanisms in a single video database system, which will be presented in section IV.

A. Video Processing

This paper does not intend to provide detailed information about video processing, but still we will provide some basic background information to offer a context for the proposal of a new video database model. Firstly, video formats are discussed in relation to compressed and uncompressed videos. Secondly, video shot detection methods to split a whole video into a sequence of meaningful video shots are presented. Thirdly, methods to extract video key features which will be utilized by users when searching through the database are introduced and finally, some video shot classification methods used to classify video shots into clusters are presented.

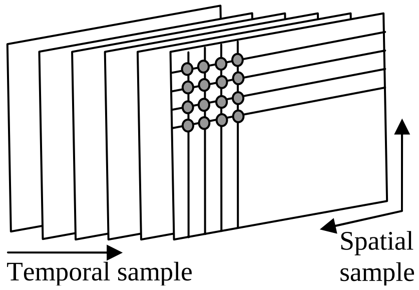


Fig. 2. Spatial and temporal sampling of a video sequence.

B. Digital Video Formats

Digital video is a representation of a natural (real-world) visual scene, sampled spatially and temporally (cf. Fig. 2). A scene is sampled at a point in time to produce a frame. Sampling is repeated at intervals, called spatial sampling, (e.g. 1/24 second intervals) to produce a moving video signal. In each scene, a frame is sampled at certain points, called pixels, positioned on a square or rectangular grid. At each pixel, stored information often includes its color like RGB (Red–Green–Blue) method or its color and luminance like YCbCr method [14], [15].

Videos are often compressed prior to being stored and decompressed before being displayed on the user screen. There are many video formats all using the CODEC model to compress and decompress videos [15]. A video CODEC (cf. Fig. 3) encodes a source image or video sequence into a compressed form and decodes this to produce a copy or approximation of the source sequence. If the decoded video sequence is identical to the original, then the coding process is said to be 'lossless'; if the decoded sequence differs from the original, the process is said to be 'lossy'. A video encoder consists of three main functional units: a temporal model, a spatial model and an entropy encoder.

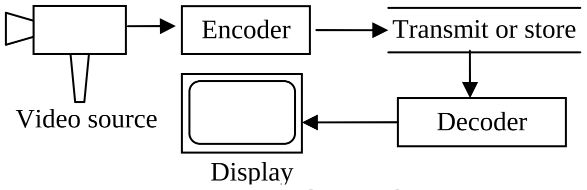


Fig. 3. An enCOder/DECoder.

The goal of the **temporal model** is to reduce redundancy between transmitted frames by forming a predicted frame and subtracting this from the current frame. The output of this process is a residual (difference) frame and the more accurate the prediction process, the less energy is contained in the residual frame. Fig. 4 illustrates the residual form of two adjacent frames. The obvious problem with this simple prediction is that a lot of energy remains in the residual frame (indicated by the light and dark areas) and this means that there is still a significant amount of information to compress after temporal prediction. Much of this residual energy is due to object movements between the two frames and a better prediction may be formed by compensating for motion between the two frames. To reduce this energy, we divide a frame into multiple *NxN* blocks and search for their movement directions called motion vectors. With this approach, the outputs of temporal model are the motion vectors and the residual forms of appropriate blocks belong to two frames. Consider the following example, there is a block (j, k) in the i^{th} frame which moves to the position (m, n) in the $(i+1)^{th}$ frame. If we subtract the whole frame *(i+1)* from frame *i*, the residual form at block (j, k) has high remaining energy because this block already moved to another location. In contrast, this energy is really small if we subtract block (m, n) of the frame *(i+1)* by block (j, k) of the frame *i* because they store the same object.

The function of the **spatial model** is to decorrelate further image or residual data and to convert it into a form that can be efficiently compressed using an entropy coder. The purpose of the transform stage in an image or video CODEC is to convert image or motion-compensated residual data into another domain (the transform domain). The choice of a transform depends on a number of criteria:

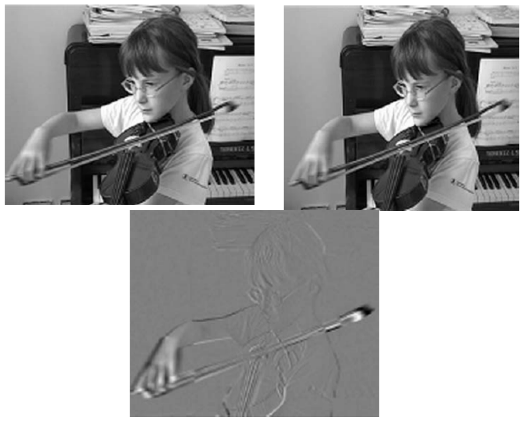


Fig. 4. Residual frame (the third one) of the first two frames.

- Data in the transform domain should be decorrelated (separated into components with minimal inter-dependence) and compacted (most of the energy in the transformed data should be concentrated into a small number of values).
- The transform should be reversible.
- The transform should be computationally tractable (low memory requirement, achievable using limited-precision arithmetic, low number of arithmetic operations, etc.).

The most transform ever-popular is Discrete Cosine Transform (DCT) [15]. The Discrete Cosine Transform (DCT) operates on **X**, a block of *N*×*N* samples (typically image samples or residual values after prediction) and creates **Y**, an *N*×*N* block of coefficients. The action of the DCT (and its inverse, the IDCT) can be described in terms of a transform matrix **A**. The forward DCT (FDCT) of an *N*×*N* sample block is given by:

$$Y = AXA^T \tag{1}$$

and the inverse DCT (IDCT) by:

$$X = A^T YA \tag{2}$$

where **X** is a matrix of samples, **Y** is a matrix of coefficients and **A** is a *N*×*N* transform matrix. The elements of **A** are:

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N}, C_i = \sqrt{\frac{1}{N}}(i=0), C_i = \sqrt{\frac{2}{N}}(i\,!=0) \tag{3}$$

The output of DCT transform will be compressed using the **entropy encoder** which converts a series of symbols representing elements of the video sequence into a compressed bit stream suitable for transmission or storage.

C. Video Shot Boundary Detection (SBD)

The first step in indexing video databases (to facilitate efficient access) is to analyze the stored video streams. Video analysis can be classified into two stages [9]: *shot boundary detection* and *key features extraction*. The purpose of the first stage is to partition a video stream into a set of meaningful and manageable segments, whereas the second stage aims to abstract each shot using representative objects such as frames, salient objects, etc. The problem of shot boundary detection will be addressed at this point while the problem of selecting key features from segmented shots will be addressed within the next section.

Shot boundary detection methods can be categorized into two main groups. The first one works on the uncompressed domain and the second works on compressed videos. Methods in the uncompressed domain can be broadly classified into five categories: *template-matching*, *histogram-based*, *twin-comparison*, *block-based*, and *model-based techniques*.

Within *template-matching* techniques, each pixel at the spatial location (i, j) in frame f_m is compared with the pixel at the same location in frame f_n , and a scene change is declared whenever the difference function exceeds a pre-specified threshold. Using *histogram-based* techniques, the histogram of a video frame and a difference function (S) between f_n and f_m are calculated using equation 4. If S is greater than a threshold, a cut is detected.

$$S(f_{m+1}, f_m) = \sum_{i=1}^{N} H(f_{m+1}, i) - H(f_m, i) \qquad (4)$$

The third method, *twin comparison*, uses two thresholds, one to detect cuts and the other to detect potential starting frames for gradual transitions. A different trend to detect shot boundary is called a *block-based* technique that uses local attributes to reduce the effect of noise and camera flashes. In this trend, each frame f_m is partitioned into a set of r blocks and rather than comparing a pair of frames, every sub-frame in f_m is compared with the corresponding sub-frame in f_n. The similarity between f_n and f_m is then measured. The last shot boundary-detection technique is termed *model based segmentation* where different edit types, such as cuts, translates, wipes, fades, and dissolves are modeled by mathematical functions. The essence here is not only to identify the transition but also the transition type.

On the other hand, methods for detecting shot boundaries that work in the compressed domain can broadly be divided into three categories. The first category uses DCT coefficients of video-compression techniques in the frequency domain. These coefficients relate to the spatial domain, and as such they can be used for scene change detection. The second category makes use of motion vectors. The concept here is that motion vectors exhibit relatively continuous changes within a single camera shot, while this continuity is disrupted between frames across different shots. The final category merges the above two trends and can be termed hybrid Motion/DCT. In these methods, motion information and the DCT coefficients of the luminance component are used to segment the video.

In summary, techniques that work upon uncompressed video data lack the necessary efficiency required for interactive processing. While other techniques that deal directly with compressed data may be more efficient, but they often lack reliability.

D. *Key Features Extraction*

Content based video indexing and retrieval requires that key features be extracted in the processing phase to improve query performance. These features include frame, salient object, audio, text, etc.

Key-frames are the represented images of a video in order that users can roughly understand the video without reviewing through its content. Key-frame extraction is closely related to shot boundary detection because to find out a shot bound, SBD algorithms usually search for the frame that has the largest differences compared to the previous one, while key-frame extraction methods detect the most unchanged frame inside each shot. It is cost saving to extract key frames at the same time as shot boundary detection. The other important feature, the salient object, is the key object displayed on the screen as long as the video shot. These are extracted from video shots and used for many purposes such as video shot classification, video retrieval and video access control.

Audio is one other important aspect of video data along with visual information. Audio can be kept as raw data and will be used to search for the stored video using its tune. However, analyzing audio is a poor performing process and so most systems, especially news video database systems, convert audio into text to reduce processing time when querying the videos. In conjunction with audio, text or captions often appear in videos so they too can be used for video classification and video retrieval efficiently because text processing is relatively faster than audio or video. When words have been collected, they need to be ‘cleaned up’ by some natural language processing algorithms to extract keywords and calculate their weights.

E. *Video Shot Classifying*

After the principal video shots and their visual features are obtained, we focus on generating higher-level visual concepts such as semantic clusters, so that more effective database indexing and access control scheme can be supported. Classification methods have been researched for a long age and there are many methods had been developed such as decision tree, k-nearest neighbor (kNN), Naive Bayes (NB), neural networks (NNet), support vector machines (SVM), etc.

The videos can be classified using its raw visual information such as visual data (color, brightness etc), audio data (tune, frequency etc) or higher level information likes texts or salient objects. To deal with visual and audio data that have a tremendous number of features, we should use methods like neural network or support vector machine who can work smoothly with large number of inputs. For example, we can use the pixels of labeled videos as inputs of a neural network to produce its weight vectors used to classify new unlabeled videos. The most important advantage of these methods is they can work on high dimensional input with acceptable time and quality. However, they are too difficult to understand for human because the result of training step is only a list of numbers.

On the other hand, decision tree or Naive Bayes are suitable for higher level classification because the number of inputs is relatively low in this case. These methods are quite simple and their training results are visual and understandable by human. More details of the above techniques are described in [9], [15], [17], [18], [16], [19], [5], [10], [6], [12].

F. *Video Database Model*

When very large video data sets are regarded, video database models and indexing can no longer be ignored if we want to support effective video retrieval and access control. In this section, we introduce a hierarchical indexing technique and its improvement to support multi-level access control.

1) *Hierarchical Video Database Model*

In order to control access efficiently, most video databases are designed as hierarchical structures such as the *semantic cluster tree* [2]. Within this structure, video contents are first partitioned into a set of semantic clusters; each semantic cluster is then partitioned into a set of sub-clusters and each sub-cluster may consist of a set of sub-regions. Using this indexing method, the system can handle multi-level access control efficiently. The indexing structure includes: a root hash table for keeping track of the information about all the clusters in the database; a leaf hash table for each cluster in order to record the information about all its sub-clusters; a second-leaf hash table for each sub-cluster in order to record the information about all its sub-regions and a hash table for each sub-region for mapping all its data points to the disk pages where the videos reside. To improve input/output efficiency, all semantic clusters are stored into a set of independent disks as shown in Fig. 5.

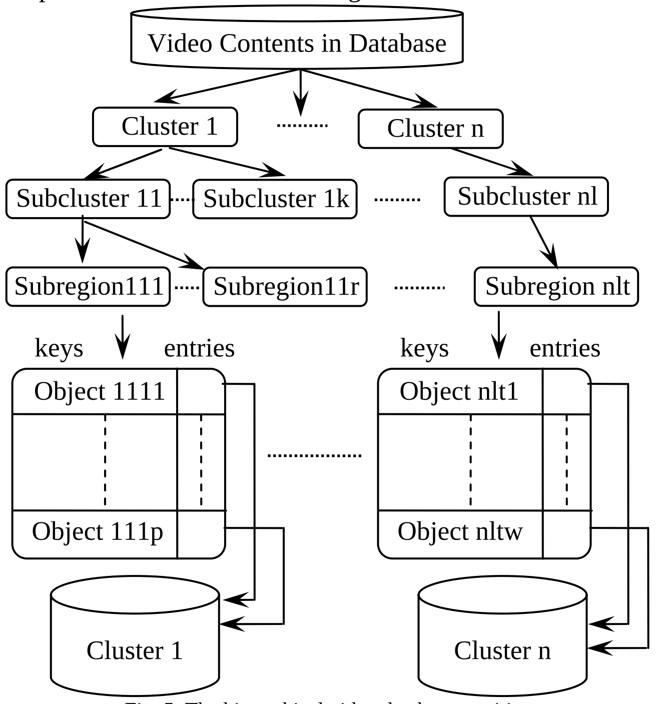


Fig. 5. The hierarchical video database partition and cluster-based indexing structure.

2) *New Finer-Granular Hierarchy Video Database Model*

The model described above has many advantages but it also has some limitations. Firstly, the only video unit supported is video shot while users are often interested in the whole video contains a certain shots. Secondly, the hierarchy tree is inflexible because in the case of extremely large databases, the tree level cannot be increased. Thirdly, this model cannot support access control at a frame and salient object granularity level. Finally, it looses most of the information needed for flexible content-based retrieval. Even though clusters are high semantic level extracted from other information, we still need to remain that information such as captions, audios, images etc. Given the above reasons, this article suggests a new model as illustrated in Fig. 6 to tackle these issues.

To address the first restriction, two new video levels are introduced; *video* and *scene*, meaning that a complete video may contain some scenes and a scene contain some shots. With this enhancement, a video database administrator can specify authorizations at video (most often), scene and shot levels. This article also proposes to modify the original hierarchy of the video tree to use video groups which consist of videos or other groups instead of clusters, sub-clusters and sub-regions. With this amendment, the path from root to the leaves can be controlled with greater flexibility where new groups can be introduced or existing groups removed.

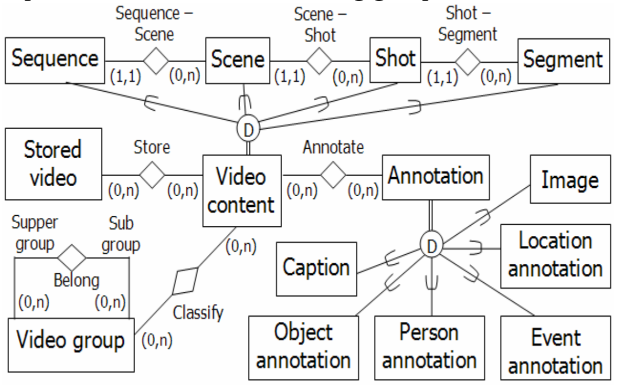


Fig. 6. The extended video database diagram.

Along with the two above amendments, it is suggested that a new video element at the lowest granularity level called *video segment* be introduced. This element would prove very useful when applying access control to a frame or a salient object. Consider the example in Fig. 7, where there are two users *A* and *B* within the system. A policy applies to user *A* that specifies that this user cannot view two frames *(j+1)*th and *(j+2)*th of video shot *V*. In addition, there is another policy that restricts user *B* from seeing object named *XXX* of video shot *V*. The easiest way to handle these two policies is to copy the whole video shot *V* to two appropriate versions. However, this solution will impinge on memory space since video data is often huge. Using segments is another solution which splits the video shot to certain segments and then copies the segments only when needed. In this example, the video shot *V* is split into 5 separate parts: (1) from the beginning to frame

j^{th}, (2) frames $j+1$ and $j+2$, (3) from frame $(j+3)^{th}$ to frame i^{th}, (4) frames $(i+1)$ and $(i+2)$, (5) from frame $(i+3)^{th}$ to the end. With this solution, we only need to copy the segment 4th, which contains only two frames, into two versions: version *#1* with original *XXX* objects, and version *#2* with blurred *XXX* objects. Then, when user A requires this video shot, the system will display the 1st, 3rd, 4th- version #1 and 5th segments while user B sees 1st, 2nd, 3rd, 4th –version #2 and 5th segments.

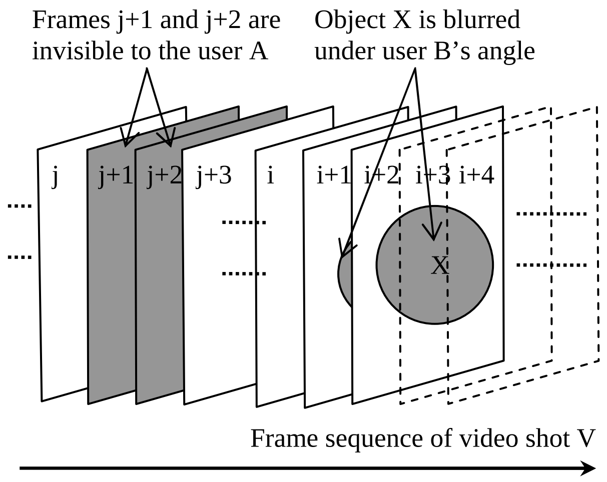

Frames j+1 and j+2 are invisible to the user A
Object X is blurred under user B's angle
j
j+1
j+2
j+3
i
i+1
i+2
i+3
i+4
X
Frame sequence of video shot V

Fig. 7. An example concerning the segments.

The final adjustment is related to annotations. Since information in videos is quite "raw" and dispersed, it is almost impossible to achieve *semantic* content-based access to videos unless some additional information is available. In order to enable flexible and intelligent access to videos, we somehow need to extract "keywords" which describe semantic contents of videos. Typically "keywords" are useful for semantic content-based access to videos include information on:

– what/who appears in the video,
– when the video was broadcast/recorded,
– where the video was recorded,
– what the video is about, etc.

In order to achieve this goal, more annotation is required such as object annotation, person annotation, location annotation, event annotation, caption and image. The first four annotations lend themselves naturally as annotations since they answer four key questions *who*, *what*, *when* and *where* about a video. Caption annotation is broadly used in news video databases where this kind of information exists on almost video news. A video database application rarely uses image annotation because of poor image processing performance. However, it is utilized in some special video databases such as airport and gas station security systems to scan for unusual baggage and terrorist activity.

IV. Flexible Access Control Model

In this paper, a content-based access control model is suggested which is reliant upon high level features extracted during the video processing stage. The goal of the system is to provide a flexible framework that can support different security levels against the video database.

The architecture of the system is illustrated in Fig. 8. There are three main components within this architecture: *authorization*, *query engine* and *authorization management*. The *authorization* component is responsible for filtering authorized videos while the *query engine* searches for interesting videos that a user has requested. The final component, *authorization management*, handles granting permissions and ensures the consistency and integrity of the video database system.

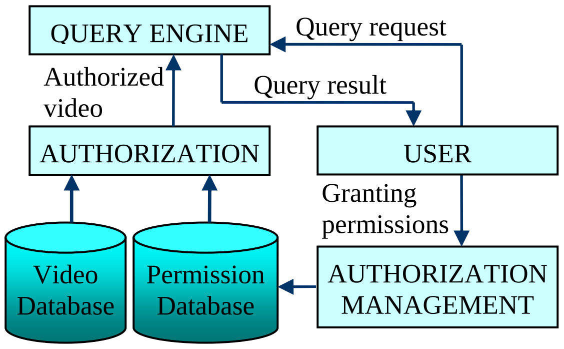


Fig. 8. Video database flexible access control architecture.

A. Authorization Model

In this section, an authorization model based on a flexible authorization model suggested by Bertino *et al.* in [2], [3] is introduced. The proposed model provides both multiple access control polices and a multi-level access control mechanism. This model also allows the administrator to specify multiple authorizations over any users or user groups (named *subject of the authorization*) against any video level such as videos, scenes or video shots.

1) Notation and Definitions

This new model manages access control via authorizations. The subject of each authorization is a user or a user group. A group can contain some users and/or other user groups. The relationship between a subject s and a user group G_k can be either *direct* or *indirect* [2]. If s is a member of G_k, we count this relationship as direct, written $s \in_1 G_k$. In contrast, the relationship is indirect, written $s \in_n G_k$, $n > 1$, if there exists a sequence $<s_1, s_2, \ldots, s_{n+1}>$, such that $s_1 = s$, $s_{n+1} = G_k$ and $s_i \in_1 s_{i+1}$, $1 \le i \le n$. The sequence $<s_1, s_2, \ldots, s_{n+1}>$ is called a membership path of s to G_k, written $mp(s,G_k)$. Let $MP(s, G_k)$ represent a set of memberships of s to G_k, either direct or indirect.

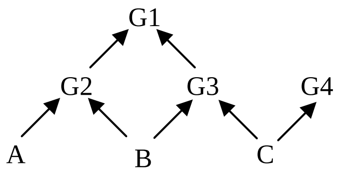


Fig. 9. An extended tree of users and user groups.

In a similar manner for users, the video content of our system is also organized into an extended tree structure. Let *V*, *VG* and *VO* represent the videos, video groups and video objects (frames or salient objects) respectively. A video group can contain videos and other video groups. We use $v \in_k vg$ to denote the relationship where v belongs to vg with a relationship type that is direct ($k = 1$) or indirect ($k > 1$). We also use $mp(v,vg)$ to represent the membership path of v to vg and $MP(v,vg)$ stands for the set of all paths of v to vg.

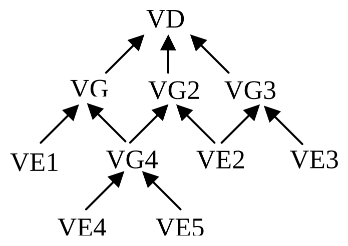


Fig. 10. A sample hierarchical video database.

Authorization handles whether a user or user group can access (*positive authorization*) or cannot access (*negative authorization*) a video element. The authorization model within this article is based upon the multi-level video access control model described in [3]. However, within this new system, the target of each authorization can be *a node on the video content tree* instead of a single table. Also supported are two kinds of authorization called *hard* (authorization that cannot be overridden) and *soft* (authorization that can be overridden). For example, the authorization that an under 18 years of age user not be able to view some specific shots of the videos without any exception should be a hard one.

Let U denote all users, G the set of user groups, $S = U \cup G$ the set of all subjects, V the set of video contents, VG the set of video groups, $VD = V \cup VG \cup VO$ the set of all video elements, AZ the set of all authorizations in our system. Authorizations can be defined as follows.

Definition 1 (Authorizations): An authorization is a 5-tuple of the form (s, v, pt, g, at) where $s \in S$, $v \in VD$, $pt \in (+, -)$, $g \in U$, $at \in \{soft, hard\}$.

The authorization states that s has been granted (if pt = "+") or denied (if pt = "-") access permission on video element v by user g with authorization type is at (*soft* or *hard*). For example, the tub $(A, VG4, +, B, hard)$ means the user B has granted access permission on video group $VG4$ to user A with authorization type is *hard*. Given an authorization a, let $s(a)$, $v(a)$, $pt(a)$, $g(a)$, $at(a)$ denote the subject, target, access type, grantor and authorization type, respectively.

Since a user can belong to a number of different user groups, he or she can be affected by multiple authorizations and some of them have opposite access types over a video element. It is the reason why we need to define the rules to decide which authorization has more priority than the others in case the conflict happens. Our overriding authorization model is a user-driven one means it prioritizes the authorizations based on the relationship between their subjects. The authorization has a more detail subject will have higher priority.

Definition 2 (Overriding authorization): Consider p_i and p_j are two different authorizations, p_i overrides p_j over user s against video element ve, written $p_i >_{s,ve} p_j$, $s \in_m s(p_i)$, $s \in_n s(p_j)$, $m,n \geq 0$, $ve \in_l v(p_i)$, $ve \in_k v(p_j)$, $l, k \geq 0$, iff any of the following conditions is satisfied:

– $at(p_i) > at(p_j)$, means $at(p_i) = hard$ and $at(p_j) = soft$
– $at(p_i) = at(p_j)$ and $s = s(p_i)$, $s \mathrel{!=} s(p_j)$
– $at(p_i) = at(p_j)$ and ($\forall mp \in MP(s, s(p_j))$: $s(p_i) \in mp$ or $\exists s' \in mp$, $\exists p' \in AZ$, $s' \neq s(p_j)$, $s' \notin_k s(p_i)$, $p' >_{s',ve} p_j$)

The above definition can be explained as the followings:

– p_i *override* p_j if the authorization type of p_i is *hard* while p_j's authorization type is *soft*.
– p_i override p_j if p_i and p_j have the same authorization type and p_i is applied over s directly while p_j is not.
– p_i override p_j if p_i and p_j have the same authorization type and for all membership path mp of s to $s(p_j)$, either $s(p_i) \in_k mp$ or exists $s' \in mp$, $p' \in AZ$ and p' override p_j over user s against video element ve.

Example 1: Consider a video database that contain the below set of authorizations:

p1: (G1, VG1, -, C, 1, soft)
p2: (G1, VG4, -, C, 1, hard)
p3: (G2, VG1, +, C, 1, soft)

where *G1*, *G2*, *G3*, *C* are users and user groups in Fig. 9 and *VG1*, *VG4* are video elements in Fig. 10. In this example, user *A* is affected simultaneously by *p1*, *p2* and *p3* authorizations. From *p1*, *A* can access *VG1* whereas *p1* does not allow *A* to access *VG1*. Because *G2*, subject of *p3*, has more detail than *G1*, subject of *G1*, so *p3* overrides *p1* over user *A* and video element *VG1*. In addition, *p2* authorization is a hard one so it will override all other authorization, including *p3*. In this example, user A can only access *VE1*.

Definition 3 (Conflict): Two authorizations p_i and p_j are conflict with respect subject s and video element v, written $p_i <>_{s,v} p_j$, with $s \in_m s(p_i)$, $s \in_n s(p_j)$; $v \in_l v(p_i)$, $v \in_k v(p_j)$; $i, j, l, k \geq 0$, iff $pt(p_i) \neq pt(p_j)$ and neither $p_i >_{s,v} p_j$ *nor* $p_j >_{s,v} p_i$.

In our system, we avoid any conflict by checking any actions that may cause the conflict. The detail of this task will be described in section C.

2) *Authorization Algorithm*

To make sure the users can only view video contents they allowed to access, we suggest an algorithm to retrieve the appropriate videos based on the actor and the set of authorizations in the system. Firstly, we define some more definitions will be used in this section.

Definition 4 (Video database projection): The projection of a video database VD with respect to a video element ve, written $\Pi_{ve}(VD)$, is a set of video ve_i such that $ve_i \in_k ve$, $k \geq 0$. It means $\Pi_{ve}(VD)$ only contains the child nodes of ve in the hierarchical video tree.

$$\Pi_{ve}(VD) = \{v: v \in VD, v \in_k ve, k \geq 0\} \qquad (5)$$

Definition 5 (Video database prune): Consider a set of videos $VS = \{ve_1, ve_2, \ldots, ve_n\}$, the result after VS is pruned, written $\measuredangle(VS)$, is a set contains the elements of *VS* and each one is not a child of any other elements inside VS. It can be described formally as follow.

$$\measuredangle(VS) = VS - \{v_i: v_i \in VS, \exists v_j \in VS, v_i \in_k v_j, k > 0\} \quad (6)$$

The purpose of the prune operator is to filter the nodes and to keep only nodes at highest levels in the tree. We define this operator because if a user can access a video element, he or she will be able to access all its children in the video tree. For instance, $\measuredangle\{VE2, VE4, VE5, VG2, VG3\} = \{VG2, VG3\}$.

Next is the algorithm to filter the list of video contents that a user can access. First of all, it will get all video elements granted to the user by positive authorizations. Then, it collects the video elements that are inaccessible to that user. This list contains all video elements that were granted by negative authorizations except the video contents that the negative authorization is overridden by a positive one.

ALGORITHM 1. FILTER VIDEO CONTENTS THAT A USER CAN ACCESS

```
METHOD authorizeVideo(u)
  initialize AV to be empty
  let pos_permission and neg_permission are lists of
  positive and negative permissions respectively
  let UV is a list of videos that the user cannot access
  let TV is a temporary list of videos
  for each permission $p \in P$ do
    if ($u \in_k s(p)$) then
      if ($pt(p) = +$) then
        add p to pos_permission
      else
        add p to neg_permission
      endif
    endif
  endfor
  for each permission $p+ \in pos\_permission$ do
    $AV = AV \cup v(p+)$
  endfor
  for each permission $p- \in neg\_permission$ do
    $uv = v(p-)$
    for each permission $p+ \in pos\_permission$ do
      $TV = \measuredangle(\Pi v(p+) \cap \Pi v(p-))$
        for each video element $ve \in TV$ do
          if ($p+ >_{u,ve} p-$) then
            $uv = uv - ve$
          endif
      endfor
    endfor
    $UV = UV \cup uv$
  endfor
  $AV = \measuredangle(AV - UV)$
  return AV
END AUTHORIZEVIDEO
```

Fig. 11 illustrates the meaning of this algorithm. In this figure, *AV* represents all video elements the user was granted access permission. *UV* represents the video elements the user was denied access permissions. $TV = AU \cap UV$ represents the video elements belong to both accessible and inaccessible ones. $v_i \in TV$ is a video element that was granted by the positive permission *p+* and negative permission *p-* and *p+* override *p-*. Finally, that user can access all videos belonging to orange parts (left).

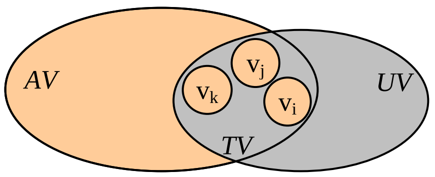


Fig. 11. An illustration of algorithm 1.

B. *Query Engine (Video Retrieval)*

This component collects requests from end users and searches through the authorized videos to retrieve those relevant and returns them to the users.

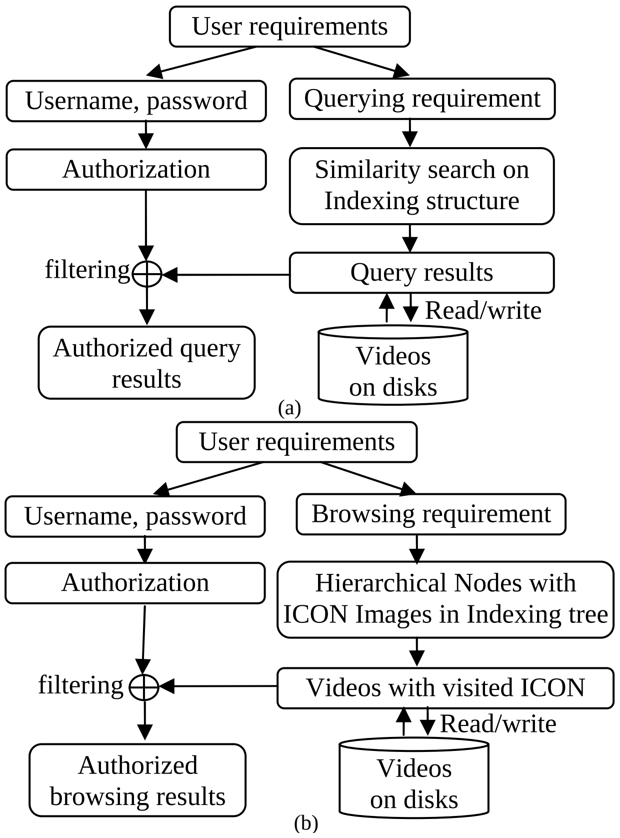


Fig. 12. The access control model under:
(a) querying mode and (b) browsing mode.

The query engine must be reliable, meaning that users can only access those videos to which they have been granted permission. The component must also be flexible, in order to support various means for the users to reach their interesting videos. Bertino *et al*. [2] suggested a system to support two access methods named *querying* and *browsing*. Under the querying method, users request specific video shots based on some criteria. By contrast, under browsing mode, users browse through and navigate the video database through its semantic categories. Based on the previous result, we introduce two adapted algorithms for the same problem with respect to our newly proposed video database schema. Fig. 12a and 12b illustrate the phrases in querying and browsing mode respectively. As can be seen from the diagrams, both algorithms include an authorization phase as described previously in section A. Next, we will present the two access methods described above in more detail.

1) *Querying Mode*

Under the querying mode access control, a user submits a query to require the access to a video element. A query is a n-dimensional tuple $(x_1, x_2, \ldots, x_n)$ of which x_i, $i = 1..$ n is a value of the i^th feature. Below is the algorithm to retrieve video elements based on the features input.

ALGORITHM 2. QUERYING ACCESS CONTROL MODE

```
INPUT:
   User ID
   A query with $(x_1, \ldots, x_n)$ format where $x_i$ is a feature's value
OUTPUT:
      AIV (authorized interesting video) – set of authorized
      filter video elements or
      ACCESS_DENIED if there is no video matches the query
METHOD queryVideo(u, $(x_1, \ldots, x_n)$)
   AV = authorizeVideo(u)
   if (AV is empty)
      return ACCESS_DENIED
   else
      AIV = solve_query(request, AV)
      if (AIV is empty)
         return ACCESS_DENIED
      else
         return AIV
      endif
   endif
END queryVideo
```

This access control procedure consists of two main steps:

(1) Firstly, it narrows the search space by filter a list of videos the user can access;

(2) Secondly, it calculates all matching ranks between each video element in the *AV* list and the input query. Then, the system returns the result which will be videos ordered by their similarities with the input query.

This is a change compare to the original algorithm introduced in [2]. The original algorithm calculates matching ranks first then filters the list based on authorization rules. Here we have reversed this sequence in order to reduce the search space as soon as possible.

'Solve_query' takes $x = (x_1, x_2, \ldots, x_n)$ as an input and calculates the similarity between the x vector and each authorized video. Then, it only keeps N videos which have the highest similarity measures which exceed a predefined threshold. The most popular features used for searching would be the video group, location (the 'where'), produced date (the 'when'), related persons (the 'who'), texts, pictures and audios (the 'what'). These features were extracted from the contents of videos while they were being processed. This is the reason why the access model presented is called a *content-based* video retrieval model. Each feature may be defined a weight representative of its importance level compared to others. Similarity between a video v and a feature vector x is defined as below.

$$rank(v,x) = \sum_{i=1}^{N} match(v_i, x_i) * w_i \tag{7}$$

where v_i represents for the i^{th} feature of video v and w_i is x_i's weight.

Matching video group, location, date, person features are quite simple since they are obviously matched (*match = 1*) or unmatched (*match = 0*). For example, if video v belongs to sport group then *match(v, 'sport')* = 1. In contrast, matching text, image and audio features is difficult since they require text processing, image processing and audio processing knowledge, respectively.

When extracting text, it is split them into words and only the keywords s are stored. That is, words appearing more frequently than a given threshold. To calculate the similarity between an input string containing n keywords $\{k_1, k_2, \ldots, k_n\}$ and a video v, we use the formula below.

$$match(s,v) = \sum_{i=1}^{n} count(k_i, v) * w(k_i) \tag{8}$$

where $count(k_i,v)$ returns number of appearance of k_i word inside video v and $w(k_i)$ is weight value of k_i.

For the produced date, the matching value is bigger when the video is newer and vice versa. Give d is the interesting date that a user wants to query the videos, the distance between a video v and d is calculated as below.

$$match(d,v) = \frac{|d - d_v|}{\max date - \min date} \tag{9}$$

where v_d id the date when v is produced, *max_date* is the produced date of the newest video and *min_date* is the produced date of the oldest video in the search space.

There are some variants of audio retrieval methods such as *query by keywords* and *query by examples*. ***Query by keywords*** applies to audio content and basically follows the same approach used in traditional text based information retrieval. The user provides several keywords as search terms, and the query engine compares them with the textual information attached with audio files in the database to determine the list of returns. ***Query by example*** is a more natural way for retrieving audio content. For example, suppose we are looking for a music masterpiece. We have no clue of the title, but we have a short portion of it, for example, a 10 second clip. We can use this piece of audio sample, normally in the format of a file, as a query object. The search engine analyzes the content of query example, computes acoustic features, compares the audio content with audio files in the database, and then generates search results accordingly. The major issue of this query method is how to speed up the search procedure.

Some modern video databases support image searching, especially faces tracking. For example, camera systems in airports are responsible for detecting unusual objects such as unattended bags or terrorist activity. These systems must discover the above items in quick time in order that security guards will have the ability to react. To improve searching performance, the data needs to be prepared offline using machine learning methods like neural network and support vector machines, etc.

2) *Browsing Mode*

Under the browsing access control mode, a user browses and navigates through video groups without specify searching criteria. Browsing refers to a technique or a process where users skip through information rapidly and decide whether the content is relevant to their needs. Browsing video databases should be like scanning the table of contents and indices of a book, or flipping through the pages, to quickly get a rough idea of the content and gradually focus on particular chapters or sections of interest. We believe the proposed semantic clustering technique and cluster-based hierarchical indexing structure would be very suitable for such fast browsing.

C. *Authorization Management*

The main purpose of authorization management component is to maintain the consistency and integrity of the system. It is responsible for validating all actions that may cause unsolvable conflicts to occur. Consider the example used in definition 3, where two authorizations p1 and p2 are conflict if exist a video element *ve* and a user *u* affected by them and neither $p1 >_{u,ve} p2$ nor $p2 >_{u,ve} p1$.

With two types of authorization–*soft* and *hard*, we may have three kinds of relationship between the authorizations: *hard–hard*, *hard–soft* and *soft–soft*. The second relationship (*hard-soft*) cannot be a conflict because a hard authorization always overrides a soft one. In addition, to prevent the conflicts between hard authorizations, this newly proposed system would only accept negative hard authorization. This means all positive authorizations have a soft property.

$$\forall(p) \in P, pt(p) = + \Rightarrow at(p) = soft \quad (10)$$

This restriction is quite natural because we might often prohibit a user from accessing to some kinds of videos and rarely do we force a user to always access some particular videos.

Finally, there is only the last relationship, *soft – soft*, needed to be verified for conflicts. Below are four actions of an administrator that may cause a conflict to occur:

– Adding a new authorization.
– Adding an existing user subject to a group.
– Adding an existing video element to a group.
– Deleting an existing authorization.

To support checking the consistency of the system, we define a new term named *general conflict* as follows.

Definition 6 (General conflict): Two authorization *p1* and *p2* are generally conflict, written $p1 <> p2$ if exists at least one video *v* and one user *u* such that $p1 <>_{s,v} p2$.

For each kind of action, we suggest a different algorithm to check conflict individually.

1) *Check Conflict when Adding a New Authorization*

When a new authorization *p(s, v, pt, g, at)* is added to the system, a conflict may occur over children nodes of *s* in the user tree. Consequently, the system must verify the conflict between *p* and each authorization *p'* affects any children of *s*.

ALGORITHM 3. CHECK CONFLICT WHEN ADDING A NEW AUTHORIZATION

```
INPUT:
   Authorization p: (s, v, pt, g, soft)
OUTPUT:
   True: if the system still is consistent means there is no
    conflict happens
   False: otherwise
METHOD checkNewPermission(p)
   PP = empty
   for each $s' \in \Pi(s)$
        for each $p' \in P$
            if ($pt(p') \neq pt(p)$) and ($\Pi v(p') \cap \Pi v(p) \neq \phi$) and
             ($s' \in_k s(p')$)
                 $PP = PP \cup p'$
            endif
        endfor
   endfor
   for each $p' \in PP$
        if $p' <> p$
            return False
        endif
   endfor
   return True
END checkNewPermission
```

The first step in the above algorithm is to collect a list of authorizations needed to be verified for conflict against *p*. This list includes all authorizations *p'* which i) has opposite access type compared with *p*, ii) *p* and *p'* affect to at least one video element, iii) subject of *p'* is an ancestor of *s'* or its children. The second step verifies conflict of each authorization in the above list against *p*. This algorithm will return *False* whenever a conflict occurs. Otherwise, it returns *True* meaning the new added authorization is valid.

We will use Fig. 12 to explain the algorithms in this section and the next two sections. The video database in this figure contains five existing authorizations listed {*p1, p2, p3, p4, p5*}. When adding a new authorization *p6* which restricts the permissions of *G5* over *V1*. Based on the algorithm, *PP* contains three authorizations {*p1*, *p2*, *p4*} needed to be verified conflict with *p6*. Authorization *p3* does not belong to this list because it has the same access type as p's. We also don't need to verify *p5* because it and *p6* affect to two disjoin video set. On completion, the algorithm returns *False* because there is one conflict between *p6* and *p4* over user *D* and video group *V1*.

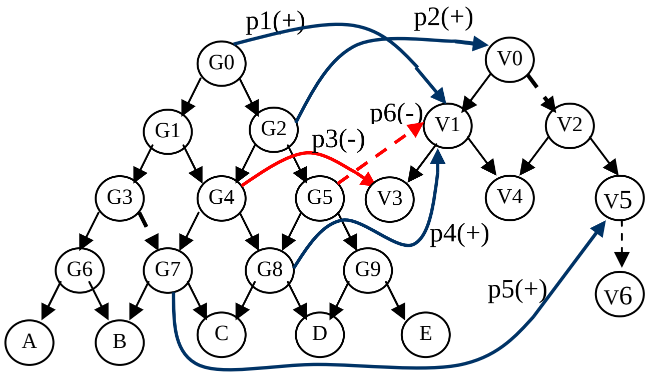


Fig. 13: A video database with some authorizations.

We are now proving that our algorithm is correct and sufficient. Assume that the algorithm is wrong, meaning there exists an authorization named *p'*: $p' \notin GP$ and $p' <> p$. Another assumption is that there is an authorization *p'* conflicts with *p* and the system still is consistent.

With the first assumption, because $p' \notin GP$ we infer that $\prod s(p') \cap \prod s(p) = \varnothing$ or $\prod v(p') \cap \prod v(p) = \varnothing$. This means *p* and *p'* have two separated affected spaces. Therefore, they cannot conflict with each other and hence, this assumption is incorrect.

With the second assumption, let *(u,v)* be a pair of user and video where the conflict happens between *p* and *p'*. The system still is consistent means there is at least one authorization p_1 that satisfies $p_1 >_{u,v} p$ or $p_1 >_{u,v} p'$. If p_1 overrides *p* over *(u,v)*, we can infer that *p'* also overrides *p* over *(u,v)* based on the last item in the authorization definition: $\forall mp \in MP(u,s), \exists u \in mp, p_1 >_{u,v} p$. Similarly, if p_1 override *p'*, we can also infer that *p* overrides *p'*, too. Anyway, *p'* and *p* are not conflict so this assumption is not correct.

2) *Check Conflict when Adding an Existing User Subject to a Group*

When adding an existing user or user group *s* to other user group *g*, *s* will inherit all authorizations affect to *g*. Thus, we need to check conflict between a set contains the authorizations affect to *g*, named *GP*, and another set *SP* contains the authorizations affect to *s* and its children. Naturally, this algorithm collects the authorizations of *GP* and *SP* first and then checks conflict between every each pair in those two sets.

Algorithm 4. Check conflict when adding an existing user subject to a user group

```
INPUT:
    s: user or user group
    g: user group where s will be added to
OUTPUT:
    True: if the system still is consistent means there is no
    conflict happens
    False: otherwise
METHOD checkMoveMember(s, g)
    SP = empty
    GP = empty
    for each $p \in P$ and $g \in_k s(p)$
        $GP = GP \cup p$
    endfor
    for each $p \in P$ and $s \in_k s(p)$
        $SP = SP \cup p$
    endfor
    for each $p \in SP$
        for each $p' \in GP$
            if $pt(p') \neq pt(p)$ and $\prod v(p) \cap \prod v(p') \neq \varnothing$
                return False
            endif
        endfor
    endfor
    return True
END checkMoveMember
```

In Fig. 13, if we add the user group *G7* to *G3*, two possible conflict authorization sets are *GP* = *{p1}* and *SP* = *{p2, p5}*. Since neither *p1* conflict with *p2* nor *p5*, *G7* will be added to *G3* successfully.

3) *Check Conflict when Adding an Existing Video Element to a Group*

Assuming that we are adding an existing video element *v* to a video group *vg*. The fact that two authorizations *p1* and *p2* can only conflict if they affect at least one common video, means we only verify conflict between the authorizations affecting *v* and all its child nodes in the video tree. Below the algorithm is presented in detail.

Algorithm 5. Check conflict when adding an existing video element to a group

```
INPUT:
    v: a video element
    vg: video group where v will be added to
OUTPUT:
    True: if the system still is consistent means there is no
    conflict happens
    False: otherwise
METHOD checkAssignVideo(v, vg)
    PP = empty
    for each $v_i \in \prod v$
        for each $p \in P$ and $v_i \in_k v(p)$
            $PP = PP \cup p$
        endfor
    endfor
    for each $p_i \in PP$
        for each ($p_j \in PP$) and ($pt(p_i) \neq pt(p_j)$) and
        ($\prod v(p_i) \cap \prod v(p_j) \neq \varnothing$).
            if $p_i <> p_j$
                return False
            endif
        endfor
    endfor
    return True
END checkAssignVideo
```

In Fig. 13, if we add the video group *V2* to *V0*, the possible conflict authorization set is *PP = {p1, p2, p4, p5}* and *SP = {p2, p5}*. Since there is no conflict that occurs between any pair of authorizations of *PP* list, *V2* is added to *V0* successfully.

4) Check Conflict when Deleting an Authorization

When an authorization *p(s, v, pt, g, at)* is deleted, *s* and its children will be affected again by the authorizations *p'* which was overridden by *p*. Consequently, we must check the conflict between a set containing the authorizations affecting s, named *SP*, and another set *CP* containing the authorizations affecting *s* and its children.

ALGORITHM 6. CHECK CONFLICT WHEN DELETING AN AUTHORIZATION

```
INPUT:   Authorization p: (s, v, pt, g, soft)
OUTPUT:
    True: if the system still is consistent means there is no
    conflict happens
    False: otherwise
METHOD checkDeletePermission(p)
    SP = empty
    CP = empty
    for each $p' \in P$ and $s \in_k s(p')$
        $SP = SP \cup p'$
    endfor
    for each $s' \in \prod(s)$
        for each $p' \in P$ and $s' \in_k s(p')$
            $CP = CP \cup p'$
        endfor
    endfor
    for each $p_1 \in SP$
        for each $p_2 \in CP$
            if $pt(p_1) \neq pt(p_2)$ and $\prod v(p_1) \cap \prod v(p_2) \neq \varnothing$
                return False
            endif
        endfor
    endfor
    return True
END checkDeletePermission
```

V. SYSTEM PROTOTYPE AND EVALUATION

In order to establish the practical importance of our extended video database model and novel access control mechanism, we implemented a system prototype and carried out empirical evaluations with real-world datasets. The prototype and experimental results are presented below.

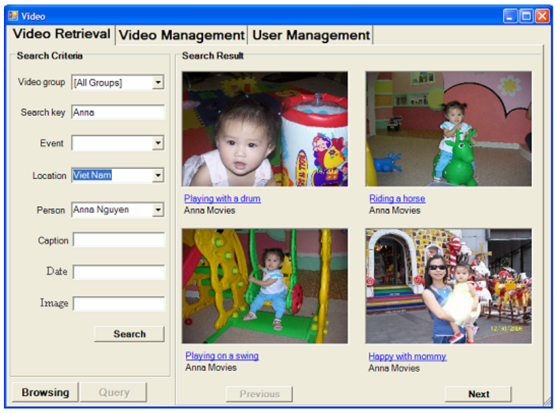
A. Choosing the Database Management System

For supporting digital video, the chosen Database Management System (DBMS) has to provide a multimedia data types such as image and video. In our framework, the video data type will be used to store the *StoredVideo* entities (cf. Fig. 6). Neither the BLOB (Binary Large Object) nor the file solutions are satisfactory because they could not provide a mechanism to identify, retrieve and use a small piece of a stored video segment. The file-based solution brings along the additional solution of managing data that is not fully under control of the DBMS. It will usually be more difficult to maintain the consistency of the system and in some cases impossible to provide necessary access restriction.

Due to these reasons, after considering popular commercial DBMSs, we decided to choose the Oracle interMedia to implement our video database by using its new multimedia data types such as ORDImage and ORDVideo. The first one, ORDImage data type, supports storing and image matching that is ideal for content-based retrieval. While the second one, ORDVideo data type, allows us to retrieve a part of the whole video stream and also to define the video stream's quality via the bit-rate parameter.

B. The Access Control Model

Implementing the *browsing mode* is quite simple because we only need to implement algorithm 1, *authorizeVideo*. In contrast, in addition to authorization problem, the *query mode* require us more efforts to refine the *solve query* algorithm. Fig. 14a and 14b are the screenshots of our system with the query and browsing modes.



Video group: [All Groups]
Search key: Anna
Event:
Location: Viet Nam
Person: Anna Nguyen
Caption:
Date:
Image:

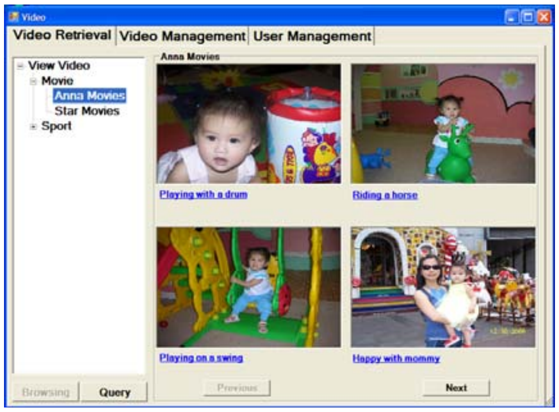
Playing with a drum
Anna Movies

Riding a horse
Anna Movies

Playing on a swing
Anna Movies

Happy with mommy
Anna Movies

(a) Querying mode.



Anna Movies

Playing with a drum

Riding a horse

Playing on a swing

Happy with mommy

(b) Browsing mode.
Fig. 14. Video retrieval pages.

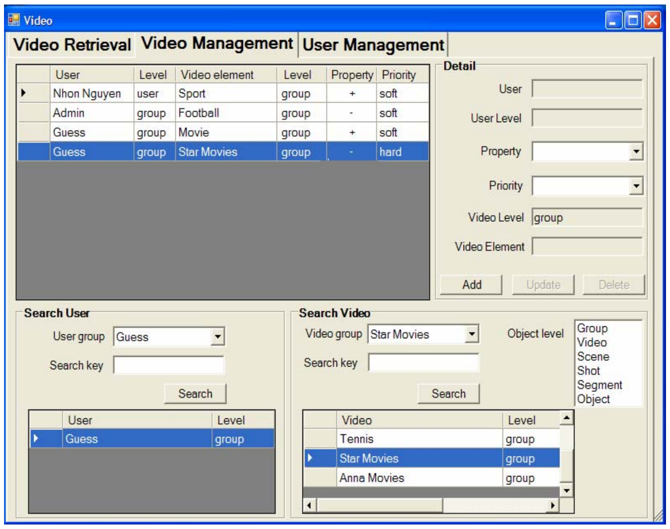
The videos shown in Fig. 14a are the ones that match the criteria entered by a user and sorted by the distances between their contents and the input query which contains keyword, video group, produced date, event, location, object, person, caption and image. Since the expressions for matching video group, location, produced date, person, object and texts (caption and keyword) had been presented in section B, hence, in this section, we will suggest a formula for the last query element, the image.

$$match(i,v) = 1 - \frac{\max(\text{ORDSYS.ORDImageSignature.evaluateScore}(i, I_j, weight))}{100} \quad (11)$$

where *ORDSYS.ORDImageSingnature.evaluateScore* is a built-in function of Oracle interMedia. It returns the distance between two images, 0 if they are identity and 100 if they are totally different. In this formula, I_j stands for the j^{th} key frame of the video *v* and *weight* has a value of *"color=1, texture=0, shape=1, location=0"*, meaning we only focus on the color and the shape while searching. In this case, we compare the input image and every key frame of the video *v* to find out the maximum similar frame. There are a number of previous works that deal with estimating the similarity between two data objects. Interested readers are directed to [8], [13], [7].

C. *The Permission Management Model*

In our system, we allow a user to specify a permission at any user level (user or user group) against multiple video levels (video group, video, scene, shot, segment and object). In addition, we implemented a strict permission management system, meaning there is no conflict accepted. It always checks the conflict occurrence when an actor adds/edits permissions, a user/user group, or a video/video group. When a conflict happens, a message is shown that indicates exactly which user and video generated the conflict. Fig. 15 shows a screenshot of the permission management page.



User	Level	Video element	Level	Property	Priority
Nhon Nguyen	user	Sport	group	+	soft
Admin	group	Football	group	-	soft
Guess	group	Movie	group	+	soft
Guess	group	Star Movies	group	-	hard

User:
User Level:
Property:
Priority:
Video Level: group
Video Element:

User group: Guess
Search key:

User	Level
Guess	group

Video group: Star Movies
Search key:
Object level: Group, Video, Scene, Shot, Segment, Object

Video	Level
Tennis	group
Star Movies	group
Anna Movies	group

Fig. 15: Permission management page.

D. *Preliminary Experimental Results*

The data set has been used to validate the accuracy and performance of the system includes 142 video clips extracted from movies, news and sport clips that fill up 2.8GBs of memory. The Movies are divided into three groups named Action, Children, and Music movies while Sport category contains Football, Tennis and Others groups. Below we present the video list in detail.

TABLE I.
EXPERIMENTAL DATASETS

Video group	Subgroup	Number of Video
Movies	Action movies	35
	Music movies	17
	Children movies	20
News		25
Sport	Football clips	25
	Tennis clips	10
	Other clips	10

There are three user groups access to this video database including: Adult, Children and Disabled groups. To fully control the access of the above groups over the scenes, 11 Action movies have been separated into multiple shots (about 5 shots for each one) and 19 shots are duplicated in order to hide some inappropriate objects. Totally, to efficiently control the access, there are 100 MB of memory added to store the extra shots.

We implemented the prototype using Visual Studio 2005 with Visual Basic/.NET. All the tests were tackled on a laptop with an Intel Pentium M processor 1.73 GHz running Windows XP/SP4, 512 Mbytes of shared memory and some Gigabytes of hard disk capacity. The disk page size was 8Kb for the datasets. With respect to the system performance, we tested and collected the intervals to query the videos, to add new permissions and to retrieve a video. Table 2 shows experimental results for these operations over our datasets.

TABLE II.
EXPERIMENTAL RESULTS

Action	Condition	Time	Description
Add a new permission	No permission in our system	5 ms	
	There are 10 existing permissions	40 ms	
	There are 40 existing permissions	120s	
Query video database	Query using text criteria (title, actor's name, etc.)	43 ms	With 12 videos returned (averagely)
	Query using image field	94 ms	With 3 rows returned (averagely)
Retrieve a video	There are 15 concurrent users are viewing videos	12 ms	

It is obvious from the above results that there are two items that have poor performance and need to be improved. Firstly, time to check conflict when adding a new permission is huge, especially when there are many existing permissions in the system. Secondly, querying using image also consumes too much time. To solve these problems, we need to study an efficient way to check conflict between two permissions and to compare two images. The correctness of our system's access control mechanism is proved by the fact that the system is robust at controlling access to the database since every user can only query the authorized videos and no "false hits" occurred in the tests.

VI. Conclusion and Future Work

In this paper, our main contribution is twofold: (1) Proposing an extended storage model for video data to support semantic visual concepts clustering and flexible content-based retrieval, and (2) Introducing a novel and flexible access control mechanism to support both multi-policy and multi-level access control in the newly proposed video databases. Our access control approach combines video indexing mechanisms with a hierarchical organization of video contents, so that different classes of users can access different video elements or even the same video element with different versions on the basis of their permissions. Besides, robust conflict checking algorithms have also been presented, ensuring conflict-free authorizations in the whole system. Preliminary experimental results with real-world datasets have confirmed the effectiveness of our proposed solutions.

In the future, we plan to investigate the efficiency of the proposed solutions with respect to the large video databases. Also, we will apply results of this research to real-world application domains such as surveillance and satellite video databases.

References

[1] N. Adam, V. Atluri, E. Bertino, E. Ferrari. A Content-based Authorization Model for Digital Libraries. IEEE TKDE, 14(2), 2002, 296-315.

[2] E. Bernito, J. Fan, E. Ferrari, M-S. Hacid, A.K. Elmagarmid, X. Zhu. A Hierarchical Access Control Model for Video Database Systems. ACM TOIS, 21(2), 2003, 157-186.

[3] E. Bernito, S. Jajodia, P. Samarati. Supporting Multiple Access Control Policies in Database Systems. IEEE Symp on Security & Privacy, 1996, pp. 94-107.

[4] A. Baraani-Dastjerdi, J. Pieprzyk, R. Safavi-Naini. A Multi-level View Model for Secure Object-oriented Databases. Data & Knowledge Engineering, 23(2), 1997, 97-117.

[5] J. Calic, E. Izuierdo. Efficient Key-Frame Extraction & Video Analysis. In: Proc. Int. Conf. on Information Technology: Coding & Computing, 2002.

[6] Chang S. F., Chen W., Zhong, D. A Fully Automatic Content-based Video Search Engine Supporting Spatiotemporal Queries. IEEE Trans. Circ. Syst. Video Tech, 1998, 1-4.

[7] Chen J., Taskiran C., Albiol A., Delp E., Bouman C. A Video Indexing and Browsing Environment. In: Proceedings of SPIE/IS&T Conf. Multimedia Storage and Archiving Systems IV, 1999, pp. 1-11.

[8] T. K. Dang. Semantic Based Similarity Searches in Database Systems (Multidimensional Access Methods, Similarity Search Algorithms). PhD thesis, FAW-Institute, Johannes Kepler University of Linz, Austria, 2003.

[9] S. Deb. Video Data Management and Information Retrieval. IRM Press, 2005.

[10] B. Furht, O. Marques. Handbook of Video Databases: Design and Applications. Taylor & Francis Group, 2005.

[11] R. Hjelsvold, R. Midtstraum. Modelling and Querying Video Data. VLDB 1994, pp. 686-694.

[12] K. Hoashi, M. Sugano, M. Naito, K. Matsumoto, F. Sugaya, and Y. Nakajima. Shot Boundary Determination on MPEG Compressed Domain and Story Segmentation Experiments for TRECVID 2004. KDDI R&D Laboratories, 2004, pp. 7-12.

[13] H.-P. Kriegel, P. Kunath, A. Pryakhin, M. Schubert. MUSE: Multi-Represented Similarity Estimation. In: Proc. 24th Int. Conf. on Data Engineering (ICDE'08), Mexico, 2008.

[14] H. Kosch. Distributed Multimedia Database Technologies Supported by MPEG-7 and MPEG-21. CRC Press, 2003.

[15] I.E.G. Richardson. H.264 and MPEG-4 Video Compression. John Wiley & Sons, 2003.

[16] B. L. Yeo, B. Liu. Rapid Scene Analysis on Compressed Video. IEEE Trans Circuits & Systems for Video Technology, 5(6), 1995, 533-544.

[17] J. Y. Zhang. Advances in Image and Video Segmentation. IRM Press, 2006.

[18] H. J. Zhang. Content-based Video Browsing and Retrieval. CRC Press, 1999.

[19] H. J. Zhang, A. Kankanhalli, S. Smoliar, S. Tan. Automatically Partitioning of Full-Motion Video. Multimedia Systems, 1(1), 1993, 10-28.

Multiplicador Electrónico para *Encoder* Incremental

Agustín Cruz Contreras, Edgar A. Portilla Flores y Ramón Silva Ortigoza

***Resumen*—Se presenta el diseño y simulación de un multiplicador electrónico para e*ncoders* incrementales, el sistema consiste en un decodificador que extrae el total de la información contenida en la señal de cuadratura, esta información da referencia para resoluciones en 1x, 2x y 4x. Este multiplicador tiene como propósito elevar la resolución de la señal de retroalimentación, empleando el mismo e*ncoder*. Se diseña totalmente con circuitos digitales para su implementación en lógica reconfigurable.**

***Palabras clave*—multiplicador, *encoder* incremental, resolución de la señal de retroalimentación.**

ELECTRONIC MULTIPLICATOR FOR INCREMENTAL ENCODER

***Abstract*—We present design and experiments on simulation of the electronic multiplicator for incremental encoders. The system consists in a decoder that extracts the total information contained in the quadrature signal. This information is used as reference for 1x, 2x and 4x resolutions. The purpose of the multiplicator is to increment the resolution of the feed back signal using the same encoder. It is designed completely in digital circuits for its implementation in the reconfigurable devices.**

***Index Terms*—Multiplicator, incremental *encoder*, resolution of the feed back signal.**

I. INTRODUCCIÓN

Actualmente, uno de los mayores retos en los sistemas de instrumentación industrial, es el de mantener precisión y robustez en la cuantificación de la posición y velocidad de desplazamiento, esto, en las diferentes partes que componen un sistema en general, o de forma particular, una máquina o mecanismos de la misma [1]. Existen sistemas donde dicha cuantificación es directa, es decir, el movimiento y la posición de las diferentes partes del sistema, se pueden determinar a partir de la elección de un sistema coordenado fijo a un origen; utilizando finales de carrera y sensores intermedios para tal fin. Sin embargo, en sistemas como un brazo robot, esto, no es tan directo, por lo que se utilizan dispositivos que permiten trasladar los desplazamientos angulares de cada una de las articulaciones, en una posición o desplazamiento total. Por otro lado, en máquinas de Control Numérico por Computador (CNC), en sistemas de control industrial o en mecanismos donde el actuador principal es un motor; se debe convertir el desplazamiento rotacional del actuador, en desplazamiento lineal, con respecto a un sistema fijo.

Muchos de los esquemas de instrumentación para los sistemas antes mencionados, están basados en la utilización del transductor rotativo o lineal denominado *encoder*, debido a su facilidad de implementación desde el punto de vista mecánico, y a su relativo bajo costo de adquisición. Existen varios tipos de *encoders*, sin embargo, los denominados *encoders* incrementales, presentan gran demanda en las aplicaciones de robótica y control retroalimentado de sistemas.

No obstante la sencillez de operación del *encoder* incremental, un factor en contra de la utilización del mismo, es la gran demanda de mayor resolución que exigen las aplicaciones de hoy en día. Por lo que en el presente trabajo se diseña un circuito multiplicador para *encoder* incremental, el cual tiene como objetivo; elevar la resolución que ofrece un dispositivo de forma general. Dicho multiplicador permite aumentar la precisión en la determinación de posición y velocidad, sin tener que cambiar el dispositivo actual.

El presente trabajo se organiza de la siguiente forma: en la sección II se explica el funcionamiento básico de un encoder incremental, haciendo énfasis en los aspectos importantes que permiten desarrollar el circuito mutiplicador. La sección III explica el desarrollo del circuito multiplicador por dos 2x para el *encoder* incremental, con su respectiva simulación. El circuito multiplicador por cuatro 4x es desarrollado en la sección IV. En la sección V se discuten los resultados obtenidos. Finalmente, las conclusiones del presente trabajo se exponen en la sección VI.

II. FUNCIONAMIENTO DEL *ENCODER* INCREMENTAL

El *encoder* es un transductor rotativo que transforma un movimiento angular en una serie de impulsos digitales. El *encoder* se basa en la rotación de un disco graduado con un retículo radial formado por espacios opacos, alternados con espacios transparentes. Un sistema óptico de emisor receptor infrarrojo detecta el cambio en la superficie del disco, generando dos señales en cuadratura (defasadas 90°), las señales se identifican como A y B (Fig. 1).

Manuscrito recibido el 19 de septiembre del 2008. Manuscrito aceptado para su publicación el 25 de noviembre del 2008.

A. Cruz Contreras, Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52536; e-mail: acruz@ipn.mx).

E. Portilla Flores, Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52536; e-mail: edport22@gmail.com).

R. Silva Ortigoza, Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52512; e-mail: rsilvao@ipn.mx).

El *encoder*, como su nombre lo indica, es un dispositivo que codifica información del desplazamiento y su dirección, normalmente el mínimo desplazamiento es decodificado a partir de un ciclo completo de la señal A o B. Observando detalladamente la señal en cuadratura se puede apreciar que hay información del desplazamiento en cada flanco de las señales A y B [2], por lo que es posible decodificar la información del desplazamiento y dirección, al doble y cuádruple de la señal originalmente decodificada.
Incrementar la resolución de un *encoder* permite mayor precisión con el mismo costo de dispositivo.

En las señales A y B en cuadratura se encuentra codificada la información correspondiente al avance y su dirección, la cual puede ser en el sentido de las manecillas del reloj (*Clockwise*, CW) o en sentido contrario (*Counterclockwise*, CCW).

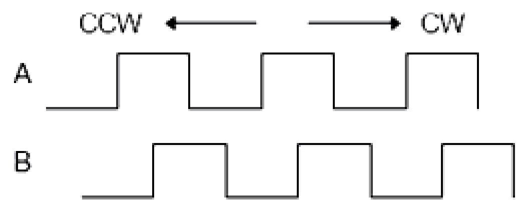


Fig. 1: Señales en cuadratura.

La información correspondiente al desplazamiento se obtiene directamente de A o B, un ciclo de la señal corresponde al mínimo avance, se puede usar como referencia el flanco de subida o bajada; para un *encoder* de 600 pulsos por revolución el mínimo avance corresponde a 360°/600=0.6°. Para determinar la dirección del desplazamiento se requiere de ambas señales; en la Fig. 2 se tiene un circuito a través del cual se determina el sentido del desplazamiento.

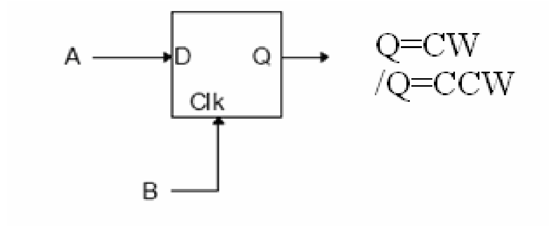


Fig. 2: Determinación del sentido de giro.

Se implementa con un *Flip-Flop* “D”, la señal A se emplea como dato y B como señal de reloj, en el sentido CW (izquierda a derecha) se captura continuamente un nivel alto, esto, porque el flanco de subida de B coincide con el nivel alto de A. Para el sentido CCW (derecha a izquierda) el flanco de subida de B coincide ahora con el nivel bajo de A.

III. Multiplicación por dos (2x)

Las señales A y B presentan en los flancos de subida y bajada puntos intermedios, estos puntos se pueden usar como referencia para obtener valores intermedios, con los cuales se puede incrementar la resolución original del *encoder*. Al tener cuatro flancos intermedios en un ciclo de la señal en cuadratura, la resolución del *encoder* puede ser multiplicada electrónicamente por 2X y 4X. Para la multiplicación por 2X, se deben considerar como puntos intermedios de referencia, los flancos de subida y bajada de A, como se muestra en la Fig. 3. El proceso de decodificación se puede dividir en dos partes; en la primera se determina la frecuencia original multiplicada por dos y en la segunda se determina la dirección del desplazamiento.

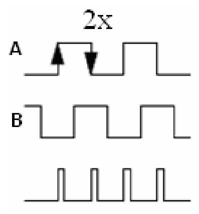
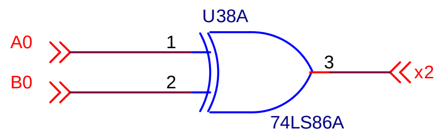
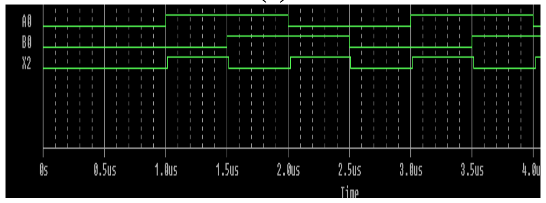


Fig. 3: Flancos a considerar en 2X.

Para el caso 2X resulta muy sencillo determinar la frecuencia final; usando una OR-EX con las señales A y B en sus respectivas entradas, la salida será 2X, la Fig. 4 muestra el circuito y simulación.



(a)



(b)

Fig. 4: circuito para 2X (a), simulación (b).

Para determinar el sentido de giro en 2X será necesario determinar el valor de las señales A y B, poco después que se presentan las transiciones de subida y bajada en la señal A. En el sentido CW (izquierda a derecha) en el flanco de subida A=1 y B=1, en el de bajada A=0 y B=0. En sentido CCW (de derecha a izquierda) en el flanco de subida A=0 y B=1, en el de bajada A=1 y B=0. Lo anterior se muestra a través se muestra el diagrama de estados para el caso 2X en la Fig. 5.

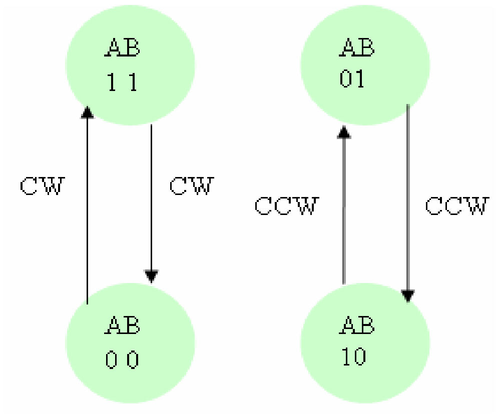
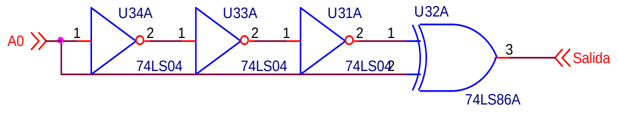


Fig. 5. Diagrama de estados para 2X.

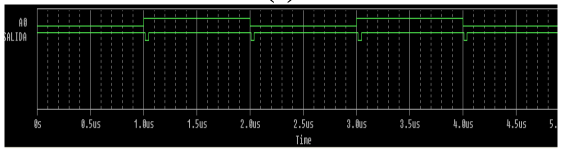
Ahora lo que se requiere es poder determinar el momento en el que se presenta un flanco de subida o bajada; para poder determinar la presencia de los flancos se pueden utilizar circuitos monoestables, con estos se genera un pulso de duración determinada, inmediatamente después de la ocurrencia del flanco. El circuito monoestable es analógico – digital, requiere de componentes externos como resistencias y capacitares, pensando en una implementación con lógica configurable, se requiere un circuito cien por ciento digital, el uso de monoestables impide esta opción. A continuación se presenta una alternativa totalmente digital.

Para la detección de de los flancos se emplea el circuito de la Fig. 6a, en este circuito se aprovecha el tiempo de propagación requerido por cada compuerta. La señal cuadrada A se aplica a la compuerta OR-EX, en una entrada llega sin retardo y en la otra se le aplica inversión y el retardo de tres compuertas. Se puede decir que durante el tiempo de propagación se tienen dos señales iguales; y la salida de la OR-EX es cero, pasado el tiempo de propagación la señal en ambas entradas de la OR-EX es diferente y la salida es uno.

En la Fig. 6b se muestra la simulación del circuito; se aprecia como para cada flanco de la señal A0, se produce un pulso en bajo, la duración de este pulso está determinada por número de compuertas inversoras, en este caso son tres, y su tiempo de propagación es aproximadamente de 40ns.



(a)



(b)

Fig. 6. Detección de flancos (a), simulación (b).

Para determinar el sentido del desplazamiento se utiliza la misma idea que en el caso 1X; “muestrear la señal A con el flanco de subida de la señal B”. Para el caso 2X es similar, a diferencia de que para 2x se de deben muestrear las señales A y B, con los flancos de subida y baja de la señal B.

En la Fig. 7 se muestra el diagrama para 2X, en este A y B son datos, y la señal CK es obtenida a partir de cada flanco de subida o bajada en B, se usa como señal del reloj para muestrear A y B, la OR-EX determina un valor para señales iguales y otro para diferentes, el cual es el criterio acorde con el diagrama de estados para 2x, con lo que se tiene CW=0 y CCW=1.

En la Fig. 8a se tiene la simulación de este circuito para el caso CW, en 8b se tiene para CCW. Se puede ver que el caso CCW presenta en la salida discontinuidades en su nivel alto, esto es debido al tiempo de propagación de las señales, que para este caso resulta indeseable. El problema del retardo se puede eliminar implementado la salida DIR en modo registrado.

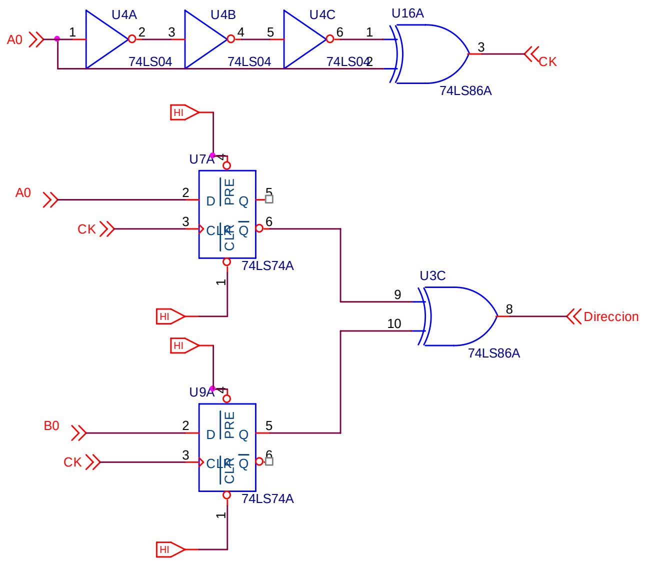
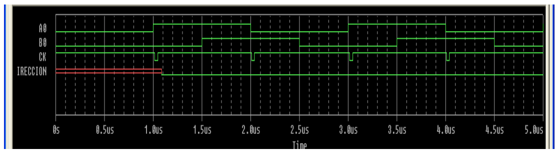
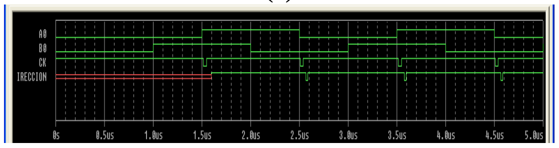


Fig. 7. Circuito para 2X.



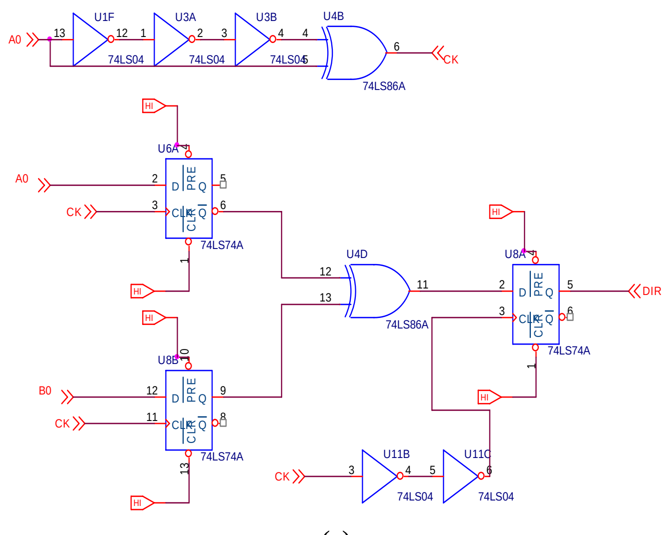
(a)



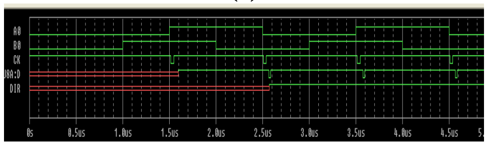
(b)

Fig. 8. Simulación para CW (a), CCW (b).

El circuito de la Fig. 9a se tiene la implementación de la salida DIR en modo registrado, para esto se usa un *Flip-Flop* y la misma señal de reloj con un retardo adicional para dar tiempo a la propagación de las señales. En (b) se presenta la simulación, en ésta se puede ver a la señal de DIR con valor constante.



(a)



(b)

Fig. 9. Salida registrada (a), simulación (b).

IV. Multiplicación por cuatro (4X)

Para el caso 4x de igual manera que en 2x la decodificación se puede dividir en dos partes; determinación de la frecuencia múltiplo de 4, y la dirección del desplazamiento.

Determinar la frecuencia múltiplo de 4 no es tan sencillo cómo para el caso 2x, en 4x se debe generar un pulso en los flancos de subida y bajada de ambas señales A y B. La Fig. 10 muestra que para el caso 4x se debe de tener en consideración el evento de los flancos de ambas señales. En el circuito de la Fig. 11 se obtiene un pulso por cada flanco de A y B, por medio de la AND se unen las dos señales para integrar la señal 4x.

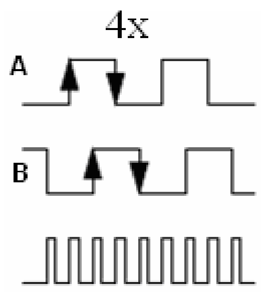


Fig. 10. Flancos para 4x.

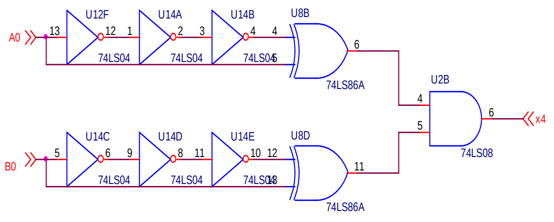


Fig. 11. Detección de flancos en A y B.

La simulación del circuito anterior se puede observar en la Fig. 12; se tienen la señales en cuadratura A y B, la detección de flancos para A y B y la salida x4. En comparación de señales 2x y 4x; 2x tiene un ciclo de trabajo del cincuenta por ciento, 4x no cumple esta condición pero, no es ningún impedimento para su aplicación.

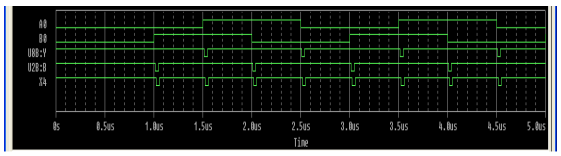


Fig. 12. Simulación, detección de flancos en 4x.

Para obtener la señal de dirección se deben considerar como el caso 2x los valores de A y B en cada flanco, en este caso se consideran los flancos de A y B, y de igual modo se revisa para CW izquierda a derecha y CCW derecha a izquierda.

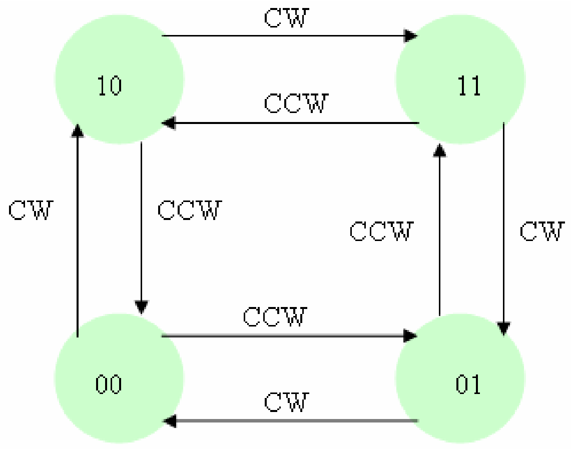


Fig. 13. Diagrama de estados 4x.

Revisando los valores de izquierda a derecha se tienen los siguientes estados: A=1 B=0, A=1 B=1 A=0 B=1, A=0 B=0, en este caso son cuatro en 2x fueron dos. El diagrama de estados de la Fig. 13 representa los estados y transiciones para el caso CW y CCW.

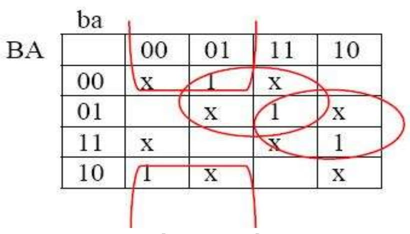
Del diagrama de estados se obtiene la tabla 1, en esta se tienen los estados presentes y siguientes para A y B, de los diez y seis estados posibles únicamente se emplean ocho, los restantes se pueden descartar plenamente dado que por la naturaleza del sistema nunca serán presentes. Considerando CW=0 y CCW=1 y estados no ocupados=X (no importa).

TABLA I.
Caso 4x

Estado presente		Estado siguiente		
B	A	b	a	DIR
0	0	0	0	x
0	0	0	1	CCW
0	0	1	0	CW
0	0	1	1	x
0	1	0	0	CW
0	1	0	1	x
0	1	1	0	x

Estado presente		Estado siguiente		
0	1	1	1	CCW
1	0	0	0	CCW
1	0	0	1	x
1	0	1	0	x
1	0	1	1	CW
1	1	0	0	x
1	1	0	1	CW
1	1	1	0	CCW
1	1	1	1	x

De la tabla se genera el mapa de Karnaugh presentado en la Fig. 14, a partir de este se obtiene la ecuación simplificada: DIR=/Ba+Ab+/A/b



BA \ ba	00	01	11	10
00	x	1	x	
01		x	1	x
11	x		x	1
10	1	x		x

Fig. 14. Mapa de Karnaugh para 4x.

El circuito de la Fig. 15 muestra la implementación de la ecuación para la Dirección, se usa una salida registrada para eliminar valores erróneos debidos al retrazo en las señales, como señal de reloj se utiliza la señal 4x con cierto retardo para captar la salida.

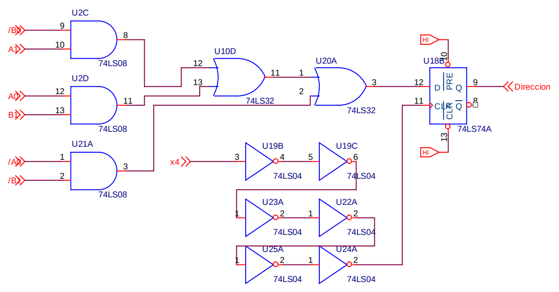
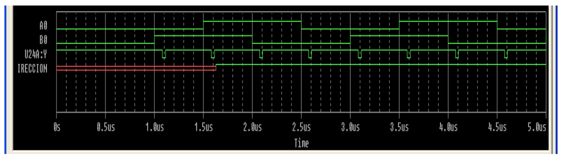
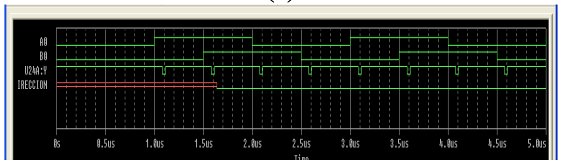


Fig. 15. Circuito para la Dirección en 4x.

El la Fig. 16a y 16b se tiene la simulación para CWW y para CW respectivamente.



(a)



(b)

Fig. 16. Simulación 4x, CWW (a) y CW en (b).

V. RESULTADOS

Se obtuvo un circuito decodificador para la señal en cuadratura de *encoders* incrementales, el decodificador proporciona la información del desplazamiento y su dirección, en las resoluciones 1x, 2x y 4x.

La simulación muestra las señales para el desplazamiento y su dirección en todas las resoluciones de acuerdo a lo esperado, la simulación se realizó en Orcad Pspice, se utilizaron circuitos TTL. En cuanto a velocidad; el sistema no tiene restricción alguna cuando trabaja en baja velocidad, en velocidad alta el sistema se ve limitado por la velocidad de los circuitos empleados (TTL).

VI. CONCLUSIONES

El incremento de la resolución es posible dado que la información existe en la señal de cuadratura; el proceso consiste en una decodificación completa de la señal de cuadratura, con el cual se extrae la información para 1x, 2x y 4x.

Incrementar a otro nivel la resolución, no es posible con *encoders* de salida digital, para mayores resoluciones se pueden emplear dispositivos con salida analógica; *resolvers* y *encoders* con salida seno/coseno [3], [4], se encuentran disponibles en el mercado.

La alta resolución que se puede obtener de estos dispositivos, se basa en el hecho de que una señal analógica tiene una resolución infinita, en estos dispositivos la resolución depende de las características del convertidor, tales como; velocidad y número de bits.

Existen en el mercado circuitos que realizan la multiplicación por cuatro 4x (LS7083/7084), el presente trabajo tiene como propósito un desarrollo con lógica digital que permita llevar el diseño a circuitos de lógica reconfigurable (FPGAs), para integrar en este mismo dispositivo todo el sistema de control.

REFERENCIAS

[1] G. Liu, A. A. Goldenberg, Y. Zhang. Precise slow control of a direct-drive robot arm with velocity estimation and friction compensation. Mechatronics, Volume 14, Issue 7, September 2004, pp. 821-834.

[2] Optical encoders' applications. Technical Articles, Computer Optical Products, Inc., 1995.

[3] Kees van der Pool. High Resolution Optical encoders. Technical Articles, Computer Optical Products, Inc. 1995.

[4] Sine/Cosine encoders, Technical Articles. Computer Optical Products, Inc., 1995.

Distance Online Learning and Evaluation Framework

C. Snae, M. Brueckner, and E. Hirata

Abstract—**In this paper, the authors present the concept of a system for Distance Object Learning and Evaluation (DOLE), which can be used during the teaching-learning process as a virtual learning environment. The term Distance Object Learning is used here for learning over a computer network or the Internet about real world entities that are distinguishable from others. The DOLE system concept uses standards for Learning Object Metadata (LOM) at different levels of abstraction. The objective of the resulting system is not only the correct and retrievable description of the course material covered by the LOM but also the further use of parts of the LOM data set for the generation of learning materials and students' learning development assessment, which can be represented by quizzes and tests. The Distance Object Learning and Evaluation system concept outlined in this paper is based in part on an earlier version of an E-learning Assessment System for Young learners (EASY). DOLE extends the concept of EASY by a learning component and by posing system generated questions with the help of a forward-chaining inference engine to find out about a specific item (object) of the domain of instruction. As the questioning process is bidirectional ("open" questions can be asked by the system as well as by the learner), DOLE is more targeted at advanced and adult learners than at young learners. The ultimate goal is to use the DOLE system concept as a part of a virtual college or university.**

Index Terms—**Distance learning, e-assessment, young learner, rule-based system.**

I. Introduction

LESSON in the teaching-learning process for distance learning [1] are typically designed to help students to find essential information or to carry out desired tasks, e. g. as assignments. As information related instruction conveys information about the domain of instruction there is no specific skill to be learned during that process; for example, in this part of the course the appearance of an object can be described. The lessons for performance-based instruction on the other hand, aim at resulting in improved procedural skills, which the students are expected to train or work out during the teaching-learning process [2].

Manuscript received May 29, 2008. Manuscript accepted for publication October 28, 2008.

C. Snae is with the Naresuan University, Phitsanulok, 65000 THAILAND (phone: +66-81-4755113; fax: +66 55 261025; e-mail: chakkrits@nu.ac.th).

M. Brueckner is with Naresuan University, Phitsanulok, 65000 THAILAND (phone: +66-89-4616110; fax: +66 55 261025; e-mail: michaelb@nu.ac.th).

E. Hirata is with the Department of English, University of Birmingham, Birmingham, (e-mail: exh411@bham.ac.uk).

To find out how much and how well students have learned the material a number of formal and informal tools and methods are used. For example, to formally evaluate student learning, most teachers use quizzes, tests, examinations, and homework, which help to assess the student's level of knowledge and to assign marks and grades.

A number of different techniques are used to assess learning informally, such as teachers listening to students' remarks and questions, teachers posing questions, and observing body language and facial expressions. With these informal assessments the teachers are able to adjust their teaching better to the students' needs. Slowing down the pace of instruction or reviewing specific material of the teaching-learning process as a response to students' demands can increase the learners' motivation and learning outcome.

The distance learning-teaching process is somewhat different to classroom teaching-learning process [3]; there are no:

- traditional, familiar classrooms,
- more or less homogeneous groups of students,
- students' questions, comments, signs of body language, and facial expressions, which the teacher can observe face-to-face,
- ways to control the distance delivery system completely,
- spontaneous ways to talk to students individually.

For these reasons, distance instructors may find it appropriate to extend the formal evaluation process of the students by testing and homework through using a more informal approach in collecting data to determine:

- student comfort with the method used to deliver the distant instruction,
- appropriateness of assignments,
- clarity of course content,
- how well class time is being spent,
- teaching effectiveness,
- the ways to improve a course,
- other types of evaluation.

Evaluation can be either formative, summative, or a combination of both [4]. Relevant data are collected for quantitative and qualitative analysis.

The formative evaluation:

- is an on-going process to be considered at all stages of instruction.
- enables instructors to improve the course as they proceed.

- facilitates the adjustment of course management and materials.
- identifies gaps in the instructional plan or the need for more adjustments.

Among the strategies used by instructors to collect formative data from their distant students are e-mail, online chat, and phone calls.

E-mail (electronic mail) is an asynchronous way of communication and can be very effective for instructors and students to communicate. Instructors can elicit and extend material covered in the online course, and students can ask questions or giving comments.

Online chat is a synchronous communication method and can also be very effective in gathering informal data about students' learning achievement and motivation. As we observe that almost all students use online chat for communication, this method can be seen as a non-interrupting way to communicate with students.

Teachers should call students often and ask them open ended questions to let students voice their concerns. Follow with probes (e.g., "*Then, will you need more information sources*?"). Set phone-in office hours but be sure to welcome calls at other times.

The summative evaluation:

- helps to evaluate the overall effectiveness of the finished course and instructional material,
- can be a basis for developing a revision plan,
- can be a baseline of information for designing a new plan, program, or course,
- does not help current students since can attend only after having completed the course.

Quantitative evaluation uses statistical methods and can evaluate data about large groups of people; under some circumstances, a considerable number of data is needed to come to statistically relevant results. Unfortunately, most of the classes in distance learning courses are small, so that they defy statistical analysis.

By definition and design, forced choice surveys offer respondents a limited number of possible response options. Therefore, new insights and novel perspectives that are not inside the provided response set will not be reported.

The often tedious nature of quantitative data collection can discourage formative evaluation, and leads to an over-reliance on summative evaluation.

Qualitative evaluation uses a wider range of information, which can be very specific und inhomogeneous, so the categorization of the data can be cumbersome. Qualitative evaluation does not depend so much on the size of the classes; small classes are generally not problematic for getting useful results [5].

For qualitative evaluation there are many different methods of data collection available, such as open ended questioning (e. g. respondents are asked to specify strengths and weaknesses of a course and suggest modifications), participant observation and non-participant observation (with the instructor participating or not participating in class and observing group dynamics and behavior), content analysis (the evaluator using predetermined criteria to review course documents including the syllabus and instructional materials as well as student assignments and course-related planning documents), and interviews (with a facilitator or specially trained person gathering data through one-on-one and small-group interviews with students).

Quantitative and qualitative evaluation can be used in various areas of the teaching-learning process and learning environment. Examples are:

- Course content (relevancy, organization of the materials, adequate body of knowledge);
- Class formats (effectiveness of the lectures, appropriateness of the lab assignments);
- Use of technology (attitudes towards technology, familiarity with technology);
- Class atmosphere (conduciveness to student learning)
- Assignments (adequate level of difficulty, usefulness, timeliness of feedback, time required for finishing);
- Tests (frequency, relevancy, sufficient review, difficulty, feedback);
- Support services (facilitator, technology, library services, instructor availability);
- Student achievement (adequacy, appropriateness, timeliness, student involvement);
- Student attitude (attendance, assignments submitted, class participation);
- Instructor (contribution as discussion leader, effectiveness, organization, preparation, enthusiasm, openness to student views).

There are also approaches, which can be seen as mixed methods circumventing some of the drawbacks of the pure quantitative and pure qualitative approach. These approaches are mainly used outside the educational environment, such as evaluating socio-economic programs and universities [6].

Much effort has been spent for the technical reuse of electronically-based distance teaching materials and in particular creating or re-using Learning Objects [7]. Learning objects (LO) are teaching units that are properly indexed (tagged) with keywords, and maybe more metadata. LOs are often stored in as XML files, which enable better indexing and organizing structures. Creating a course requires putting together a sequence of LOs [8].

A common standard format for e-learning content is SCORM [9] whilst other specifications allow for the transporting of "learning objects" (Schools Interoperability Framework) or categorizing meta-data (LOM, [10]).

In this paper, we propose a framework for distance learning and evaluation framework for Thai language. In Sect. 2 the distance learning and distance assessment processes as they are seen in this research are described in more detail. The DOLE framework is presented in Sect. 3 together with some remarks on implementation details. In the final section we draw conclusions of this research and outline further work.

II. Distance Learning and Assessment

While distance learning primarily refers to remote computer-enhanced education it is currently extending to emerging technologies, such as mobile computing (M-learning) and Personal Digital Assistants (PDAs). Distance learning may include the use of web-based technologies, including blogs, polls (electronic voting systems), simulations, games, and wikis. The differentiation to blended learning is floating.

E-learning systems are mostly used together with face-to-face learning, but they may be applied to distance learning after some adaptation. For a differentiation between face-to-face and E-learning, see Fig. 1, which shows the portion of e-learning in each of the teaching-learning models [11].

Distance learning has proven to be useful in tertiary education, e.g. universities, and in environments which need their learners to be lifelong learners. Contents of distance E-learning range from technical and medical knowledge to soft skills, such as social behavior. Even the instruction of hands-on practical work can be assisted by distance learning units.

Distance E-learning has to serve very different learner groups. There are novice learners, intermediate and advanced up to experienced students. Furthermore distance E-learning courses can be attended by dependent or independent learners who study full-time or part-time. Distance E-learning is based on prerequisites, such as management, culture, and IT [12]. Distance E-learning can be presented in many forms (see Table I).

All of these forms can be enhanced by multimedia content, which is designed to suit for various types of learners. Such multimedia materials can consist of:

- e-books,
- e-libraries, where we can borrow books online and check availability of books,
- streaming videos or audio files, where details and information are kept in multimedia files or sound and can be accessed via the Internet.

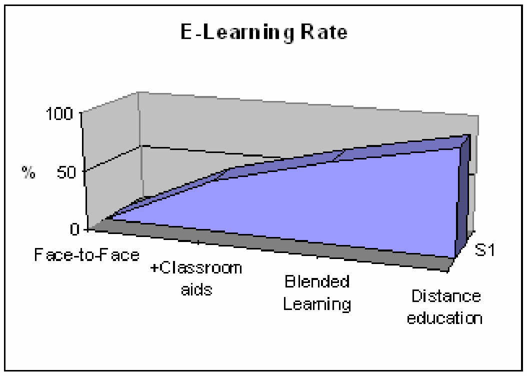


Fig 1. E-Learning rate in different learning environments.

TABLE I
FORMS OF DISTANCE E-LEARNING

Computer Based Training (CBT)	students learn by executing special training programs on a computer
Web Based Training (WBT)	students learn by executing special training programs on a computer via the Internet
Blended Learning	provision or use of resources which combine e-learning (electronic) or m-learning (mobile) with other educational resources
Virtual Classroom, Lab	Students study at home and use Voice over IP and webcams in a virtual class, e.g. performing experiments
Digital Learning Games	Computer games with an educational background

Distance E-learning can meet diverse user needs and requirements and can be consumed just in time at home, on travel or at the working place. It can be designed for the user's context in small parts of the learning content. The content can be made available to a large number of participants. Participants can work out the material self-paced. So, maximum information retention is more likely. As learners study in their natural environments travel costs and associated expenses are considerably reduced.

Application in distance learning is a developing area and can often be seen in first language education and mostly English as a foreign/second language (EFL/ESL) situation. For instance, in the research into computer assisted language learning (CALL), the effectiveness of teaching vocabulary has been reported [13]. The importance of vocabulary knowledge is prominent in understanding any language (e.g. Wilkins, 1972). In the EFL (English as a foreign language) environment in particular, learning is generally more challenging. This is because there is little mental lexicon readily available to the learners at the early stage of learning. For instance, when learners are learning objects (such as fruits, vegetables) it is hard for learners to relate them with their meaning and their existing knowledge if they had not encountered them in their real life. Moreover, in teaching young learners in general, motivation is one of the important factors to consider since they tend to be less intrinsically driven compared to adult learners.

The framework for the learning process is considered by suggesting three steps: presentation, practice and performance. Having the available systems which are targeted to learners, however, they seem to lack the attention to the evaluation stage. In the case of evaluating learners, it is often neglected or the concept of 'assessing' learners being avoided for such reasons as the result could discourage learners, in case of them receiving negative results. However, it is important to consider the way to evaluate and check the understanding of the learners in any learning. It is necessary to have an evaluation stage, after any teachings and techniques used, so that it enables educators to monitor the learners' understanding as well as the effectiveness of the techniques and approaches, which in the end, also serve as a follow-up and feed into a revision phase. The points raised above (i.e. the motivation and learning environment factors) should be fed into the design of the interface for the assessment, which will be discussed in the next section.

III. System Concept

Nowadays, user interfaces can be designed and implemented with a wealth of technical opportunities, which may lead to overshadow the important points. For distance learners, the user interface must be playful and more attractive than that for adult learners, without distracting the users from the intended conversation [14], [15].

The learning design strategy has to take into account the learners' specific behavior and cultural background. In case of Thai students, for instance, there is a great demand of multimedia content, which has to add some fun to the learning materials. From our teaching experience Thai students tend to be social learners studying in groups and comprehend through visual stimuli as well as examples and case studies.

Numerous qualitative and quantitative studies are available that analyze the usefulness of applying computer games for instruction purposes. They are mainly driven by the question as to how acquire and improve the necessary skills that people will face in the 21st century: managing information, being adaptive to changing roles and environments.

Recent interest in games and learning stems from some complex debates about the very role and practices of education in a new century, rather than just from a simple belief that young people find games motivating and fun and, therefore, that they should be exploited in educational contexts. These debates suggest, among other things, that computer games are designed 'to be learned' and therefore provide models of good learning practices, and that by playing games young people are developing practical competencies and learning skills.

Interesting application areas for computer games are the development of strategic and logical thinking as well as language. That means, students are expected to develop their hard skills as well as their soft skills. Even the assessment of students can be made within the gaming environment as long as the boundary conditions are the same for every participant Prensky [16] suggests that today's learners have changed, and that video (and computer) game players are developing skills and competencies that others are not learning, such as decision making, data handling, multi-tasking, and information processing.

Characteristics of good instructional computer games include challenge and adaptability, a more practice-based rather than a didactic approach with authentic tasks, letting the students experience consequences of interactions and choices they make. Games situate players in particular literacy practices associated with the identities being played, immersing them in peculiar vocabularies and social customs; often these literacy practices are associated with real-world professional domains, or are consistent within the fantasy. Games prepare players to deal with complex electronic environments, to negotiate and handle data in multiple formats simultaneously, to interact with images, sounds and actions, and to interact with others through electronic channels [17].

DOLE is a system concept, which can be implemented as a web based online game where users log onto the website and play/learn against an artificial intelligence (A.I.) engine. For example, players think of an animal, vegetable, mineral, or other object and DOLE has to guess which term (word) the player is thinking and vice versa. The resulting system can be used anywhere and anytime. It is fun harnessing with edutainment and game learning style. It can practice the way of learner thinking and can assess skills, knowledge, and thinking of learners. Some advantages of DOLE are described as follows:

DOLE is adaptable (learns and adapts).

- It allows users to customize the game interface.
- It works as a stand-alone application.
- It can be provided via a Web Interface.

DOLE is scalable; the knowledge base can be tailored to fit various game platforms.

- It can be adapted for mobile applications.
- It handles multiple Web servers.
- It is designed to host thousands of simultaneous users.

The software development of the user interface can be carried out using rapid prototyping (for an overview of the various rapid prototyping paradigms for the implementation of user interfaces see [18]).

The system is separated into 2 parts of object learning: 1) the user thinks of an object and the system poses questions, which the user has to answer correctly; 2) the system chooses an object and lets the user ask questions about it, which the system will answer correctly. Framework 1 is described in the following.

A. *Framework 1*

The first framework is designed using Artificial Intelligence and an inference engine with rule base forward and backward chaining. This framework can work as follows (Fig. 2):

- Manage knowledge about things/objects, e.g. fruit, animals.
- Provide yes/no questions for learners.
- Search and ask reasonable questions and lead them to the right direction.
- Display questions that learners have been answered already in the session.
- Guess the answer of a thing/object that the learner is thinking of Suggest answers for each question if learners are in need.
- Assess learners' style of learning, e.g. recognition, understanding, analysis.
- Give marks to student after assessment.
- Add/Edit/Delete data of objects, pictures, multimedia and knowledge related to material of learning.

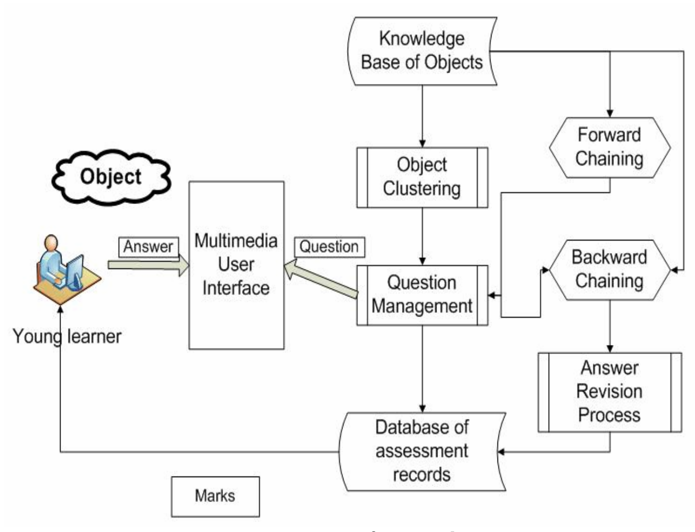


Fig. 2. DOLE framework 1.

To implement this framework the following methodologies can be used:

- Clustering technique is used to group objects or things that have similar or the same characteristics, e.g. shape, taste, smell or color.
- Inference engine tools: forward chaining (data driven) is used for finding the answer of a thing or object that the user is thinking of while using the system. The tool considers/examines all information and data from the answers provided by learners, e.g., after learners choose things/objects that they want to learn and revise such as fruit then the system asks learners about shape, taste, smell, etc. of fruit. The learners have to provide correct answers as yes or no, so that the system can get back with other questions to follow by previous learner answers and can try to guess the answer.
- Backward chaining is used to recheck and trace the answers to the questions, e.g. the answer is elephant (animal section) then the system recheck all questions and answers that learners provide such as answering yes for 4 legs, big size, has trunk and tusk etc. From this, the learners can get full marks if they provide all correct answers to questions that system has generated and less mark if they answered incorrectly. There will be no more than 20 questions in each assessment.

B. *Framework 2*

The second framework is developed using specific name matching algorithms for matching words. This framework can work as follows (Fig. 3). Firstly, a learning and assessment domain is specified by the system, from which the objects are chosen randomly, for example fruit, vegetables, or animals. The system now allocates the properties of this object and prepares for the answering task. After that, the learner poses the questions to the system to find out the correct object. With each question the assessment part of the system evaluates the value of the question for the overall investigation process. This evaluation is based on the time used for posing questions, the number of questions until the correct answer, and so on. Should the learner misspell or mistype a keyword, the system can try to find related words with the help of the allocated properties of the object mentioned above and a word matching algorithm that compares them and their synonyms with the misspelled word. Having found the most similar match the system will reply with a question to confirm the match and provide further yes/no answers.

The user interface can display all interactions (questions and answers by both partners) at any time the learner wants them to review to find the correct answer, i.e. the object that the system is thinking of. From the questioning process and its development of finding the solution the assessment part of the system can mark the learner’s comprehension, skill and knowledge. If the learners need only a few questions to find the correct solution, they get higher marks.

Parts of the concept of DOLE framework 2 are clustering and word matching techniques as outlined in the following.

Clustering is a technique used to classify similar objects into collections using criteria based on similarity, e.g. similar characteristics or similar properties. A partitioning approach can be taken to manage grouping, which results in a division of the object set into groups (classes, clusters) well separated in the feature space. The groups can be regarded as relatively "homogeneous" and labeled with properties (characteristics), so that they can be investigated separately. In DOLE, for example, the clustering technique will be used to group fruits that have similar or the same characteristics, e.g. shape, taste, smell or color. In DOLE (1) properties/characteristics (e.g., kind, shape, color, taste, smell, etc) are sorted and then related by classification; clustering stores any types of objects and some related properties and characteristics, (2) it helps the users to pose useful questions that the system has to answer, (3) it supplies fast access to required information.

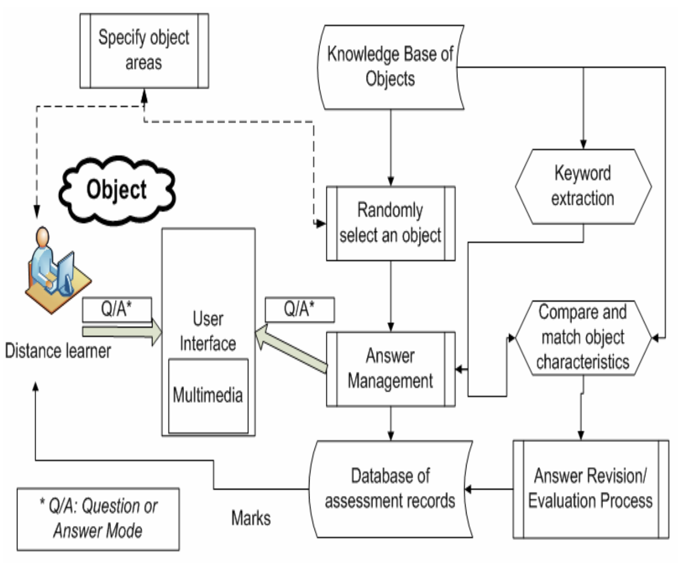

Specify object areas
Knowledge Base of Objects
Object
Keyword extraction
Randomly select an object
Q/A*
User Interface
Q/A*
Distance learner
Multimedia
Answer Management
Compare and match object characteristics
Database of assessment records
Answer Revision/ Evaluation Process
* Q/A: Question or Answer Mode
Marks

Fig. 3. DOLE framework 2.

Word matching is used to match keywords of object characteristics in the knowledge base with words from learner's questions. Word matching can deal with wrong spellings/typing from learners as well as with relating terms/keywords to respond to those spellings correctly. Example: the learner types "grene", the system is not able to find a correct instance of this keyword and will find the nearest match with the help specific name matching algorithms as described in [19], [20], [21].

IV. Conclusions and Further Work

This research has led to a concept for a Distance Object Learning and Evaluation (DOLE) system, which comprises an appropriate user interface, a domain specific ontology, and a neural network for intelligent questioning and answering. A decision tree will be used to automatically arrange and provide the questions and answers.

The framework has its strengths in case of summative assessments, which are related to the learning of concepts shared as common knowledge in the native or in a foreign language. The system can be used as a tool for guided learning, assessment and self-assessment as well.

A more advanced way to enhance DOLE is to incorporate a part for the automatic interpretation of facial expressions during the assessment process. This would add to and give a richer picture of the assessment results.

Another area of further work is the recommendation of further studies and the outline of an individual study plan for the student that has been assessed and incorporating the assessment session into an e-portfolio of learning, which is an "electronically-based portfolio, i.e. a file store and information management system which is modeled on the working method used for paper portfolios, but which takes advantage of the capabilities of ICT, notably allowing earners to store digital artifacts and streamlining the process of review and moderation for learners, tutors, moderators and verifiers" [22].

References

[1] W. Huitt. A transactional model of the teaching/learning process. Educational Psychology Interactive", Valdosta, GA: Valdosta State University, 2003. Available from http://chiron.valdosta.edu/whuitt/materials/tchlrnmd.html.

[2] D. Brethower, K. S. Pfeiffer. Performance-Based Instruction, includes a Microsoft Word diskette: Linking Training to Business Results. 1998.

[3] J. Johnson, R. Fiene, J. Keat, H. Darling, D. Pratt, J. Iutcovich. Mastering course content and learner satisfaction in early childhood education: A comparison of regular classroom instruction with three variations of internet delivery. *Journal of Early Childhood Teacher Education*, vol 22, no 4, pp 267 – 274, 2001.

[4] M. Scriven. The Methodology of Evaluation. In: R. Tyler, R. Gagne, & M. Scriven, *Perspectives of Curriculum Evaluation*, Chicago: Rand McNally, pp.39-83, 1967.

[5] M. Q. Patton. Paradigms and Pragmatism. In: D. M. Fetterman (ed), *Qualitative Approaches to Evaluation in Education: The Silent Scientific Revolution*, New York: Praeger, 1998.

[6] User-Friendly Handbook for Mixed Method Evaluations. Edited by Joy Frechtling, Laure Sharp, Westat August 1997. Available at http://www.ehr.nsf.gov/EHR/REC/pubs/NSF97-153/START.HTM

[7] A. Koohang and K. Harman. Learning Objects and Instructional Design. *Informing Science*, 2007.

[8] D. R. Rehak, R. Mason. Keeping the learning in learning objects, in Littlejohn. *Reusing online resources: a sustainable approach to e-Learning*, Kogan Page, London, pp.22-30, 2003.

[9] B. Caudill and D. Banks. Instructional Designer's Pocket Guide to SCORM (Paperback). *JCA Solutions*, First Edition edition (August 1, 2006).

[10] Learning Technology Standards Committee: IEEE Standard for Learning Object Metadata. IEEE Standard 1484.12.1, Institute of Electrical and Electronics Engineers, New York, 2002. (draft), available online http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

[11] C. Snae and M. Brueckner. Web Based E-Learning Framework for Thai Learning Environment. In: *Proceedings of International Conference e-Learning 2006: Learning Theories vs Technologies*, Bangkok, 14-16 December, 2006.

[12] C. Snae and M. Brueckner. Ontology-Driven E-Learning System Based on Roles and Activities for Thai Learning Environment. *Interdisciplinary Journal of Knowledge and Learning*, Vol. 3, pp 1-17, 2007.

[13] R. P. Donaldson and M. A. Haggstrom. Changing Language Education Through Call (Routledge Studies in Computer Assisted Language Learning). Routledge, 2006.

[14] R. Aust and R. Isaacson. Designing and Evaluating User Interfaces for eLearning.In: *G. Richards (Ed.), Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, Chesapeake, VA: AACE, pp 1195-1202, 2005. Available online from: http://elearndesign.org/papers/eLearn2005_Aust.pdf (viewed Nov. 10, 2007).

[15] S. Harri-Augstein and L. F. Thomas. Learning conversations. London: Routledge, 1991.

[16] M. Prensky, Digital Game-based Learning, New York, 2001.

[17] C. Snae, M. Brueckner, and W. Wongthai. E-Assessment Framework for Individual Learners. In: Proc. of International Conference on E-Learning: Strategies and Edutainment, Bangkok, Thailand, 7-11 March, 2008.

[18] F. Hardtke. Rapid Prototyping for User-Friendly and Useful Human Machine Interfaces. In: *Proceedings of SIMTECT 2001, Simulation Industry Association of Australia*, Canberra, Australia. Available online http://www.siaa.asn.au/get/2395363450.pdf (viewed Nov 3, 2007)

[19] C. Snae. A Comparison and Analysis of Name Matching Algorithms. *International Journal of Applied Science. Engineering and Technology*, Vol 4 no. 1, pp. 252-257, 2007.

[20] C. Snae, K. Namahoot and M. Brueckner. MetaSound: A New Phonetic Based Name Matching Algorithm for Thai Naming System. In: *Proc. of International Conference on Engineering, Applied Science and Technology (ICEAST 2007)*, 21-23 November, Bangkok, Thailand, 2007.

[21] C. Snae and M. Brueckner. A Semantic Web Integrated Searching System for Local Organizations with Thai Soundex. In: *4th International Joint Conference on Computer Science and Engineering (JCSSE 2007)* 2-4 May, Khon Kean, 2007.

[22] JISC, E-assessment (Glossary), 2006. Available from: http://www.jisc.ac.uk/uploaded_documents/eAssess-Glossary-Extended-v1-01.pdf (access Nov. 3, 2007)

Computadoras de Bolsillo como una Alternativa para el Control de Servomotores en Robótica

J. C. Herrera Lozada, I. Rivera Zárate y M. Olguín Carbajal

Resumen—**Este trabajo muestra el diseño de un sistema básico de control para servomotores convencionales, implementado en una computadora de bolsillo. La particularidad de esta realización radica en la interfaz hardware conformada por un microcontrolador que conecta al respectivo servomotor con el PDA a través del puerto serie de éste último. El sistema es de propósito general y se puede adaptar sin cambios drásticos a cualquier aplicación similar.**

Palabras clave—**Servomotores, computadoras de bolsillo, robótica.**

PDA COMPUTERS AS AN ALTERNATIVE FOR SERVO MOTORS CONTROL IN ROBOTICS

Abstract—**This paper presents a system that allows for control of conventional servo motors using PDA computers. The advantage of the proposed implementation is related to hardware interface, namely, to the usage of the specialized microcontroller that connects PDA with the servo motor using serial port of the PDA. The system can be easily adapted to other similar applications.**

Index Terms—**Servo motors, PDA computers, robotics.**

I. INTRODUCCIÓN

Considerando la creciente proliferación de la tecnología de los dispositivos móviles, en específico la tendencia al uso de teléfonos celulares y sobre todo las computadoras de bolsillo o de mano, también conocidas como PDAs (en inglés, *Personal Digital Assistant*, Asistente Digital Personal), es posible afirmar que este tipo de dispositivos son elementos indispensables en la informática contemporánea para el intercambio y proceso de información. Entre muchas otras aplicaciones, en este trabajo se propone su adecuación como lazo primario de control en la supervisión de procesos.

Esta es la continuación de un proyecto que inició con la adquisición de datos para utilizar el PDA como un monitor de señales [1]; ahora, la intención es demostrar de manera sencilla cómo se pueden enviar datos hacia un microcontrolador y que éste pueda controlar la posición del rotor de dos servomotores independientes entre sí, demostrando una alternativa sustentable dirigida hacia el área de la robótica supervisada.

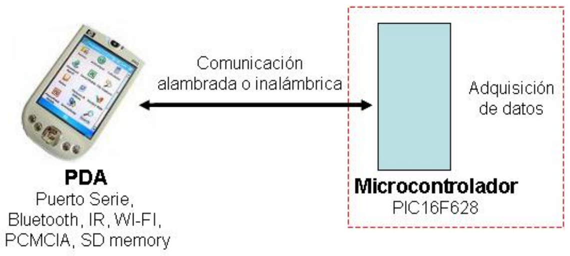


Fig. 1. Prototipo general de adquisición y monitoreo.

En un caso formal de diseño, será importante el análisis detallado de los parámetros y las condiciones del proceso a controlar con la intención de optimizar el sistema como lo demanda la teoría del control [2], [3], [4].

A. Servomotor

Los servos son una derivación de los motores a CD, estos se caracterizan por su capacidad para posicionarse de forma inmediata y exacta dentro de su intervalo de movimiento al estar operando. Para lo anterior, el servomotor espera un tren de pulsos; cada uno de estos pulsos tiene una duración específica para mover el eje de rendimiento del servomotor hacia una posición angular determinada. Se dice que el tren de pulsos es una señal codificada; por lo que cuando ésta cambia en el ancho de sus pulsos, la posición angular del eje también cambia.

Para comprender mejor el funcionamiento de estos dispositivos electromecánicos, obsérvese la Fig. 2, en donde se aprecian las señales que permiten el movimiento de un servomotor estándar.

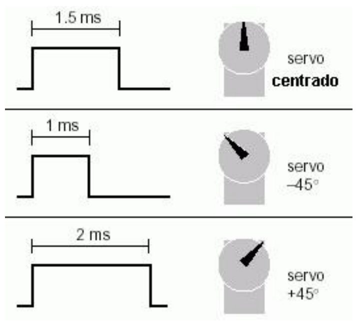


Fig. 2. Movimiento de un servomotor estándar.

Manuscrito recibido el 20 de agosto del 2008. Manuscrito aceptado para su publicación el 3 de noviembre del 2008.

J. C. Herrera Lozada, I. Rivera Zárate, M. Olguín Carbajal, CIDETEC-IPN, México D. F (e-mail:irivera@ipn.mx).

B. Computadora de Bolsillo (PDA)

En la particularidad de esta realización se utilizó una computadora de mano *iPAQ Pocket PC* de *Compaq*, modelo *3950*, con sistema operativo *Windows Pocket 2002* precargado de fábrica. Este sistema operativo es una versión más de la plataforma *Windows CE* (*Compact Edition*) La Tabla I muestra otras prestaciones importantes del equipo.

TABLA I.
CARACTERÍSTICAS DE LA POCKET PC 3950

Procesador @Velocidad	**Conectividad integrada**	**SDRAM @FLASH ROM**	**Resolución pantalla táctil**
Intel PXA250 @400MHz	USB, Serial, IrDA	64MB @32MB	240 x 320 pixeles, Transflective TFT, 65000 colores

En dependencia a la plataforma hardware del PDA (características del procesador y de la memoria) y a su sistema operativo, se eligió *Embedded Visual Tools 3.0* como ambiente de desarrollo. Éste contiene *Microsoft Embedded Visual C++ 3.0* y *Microsoft Embedded Visual Basic 3.0*, con los *kits* de desarrollo (*SDKs*) para *Pocket PC 2002* y *Smartphone 2002*.

Es importante mencionar que los PDAs más recientes, con prestaciones más sofisticadas incorporadas como por ejemplo las conectividades *Bluetooth* y *Wi – Fi*, se programan entre otras herramientas, con *Visual Basic.NET* si fuera el caso del sistema operativo *Windos Mobile* en sus versiones 5.0 y 6.0, o en otro caso, para una independencia del sistema operativo es posible acceder a la programación con *j2me* [6].

En el prototipo planteado, el grueso del procesamiento lo realiza el microcontrolador por lo que también es posible considerar el uso de alguna *hyperterminal* que permita el acceso a los puertos del PDA, para evitar el programar una interfaz de usuario, aunque este aspecto realmente depende de la complejidad de la aplicación misma.

II. DESARROLLO DE LA APLICACIÓN

La interfaz que permite la conexión con el PDA consta de dos módulos principales: un microcontrolador y un programa residente escrito en *Embedded Visual Basic* que controla el puerto serie del PDA para interactuar con el microcontrolador [1], este programa residente es el que podría ser sustituido por la *hyperterminal*, si fuera el caso.

Para la comunicación serial se requirió construir un cable *Null – Modem* de sólo 3 hilos, interconectando las señales sobrantes en el mismo conector DB9, tal y como se aprecia en la Fig. 3.

Este procedimiento emula el protocolo CTS/RTS y DSR/DTR por hardware; para controlar el flujo de datos se recurre al protocolo software XON/XOFF.

Todo el proceso se resume en sincronizar RXD en el PDA con la señal TXD a la salida del microcontrolador; a la vez, TXD del PDA con RXD del microcontrolador.

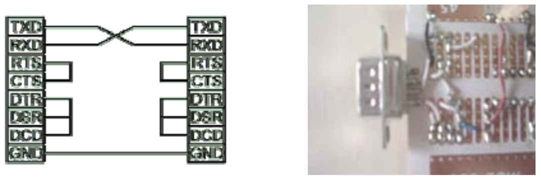


Fig. 3. Cable Null-Modem de 3 hilos.

El prototipo construido incluye un conector DB9 que se une con su contraparte de la cuna de sincronización (*cradle*) del PDA. Obsérvese en la Fig. 4, que en la tablilla que contiene al microcontrolador, las conexiones hacia el DB9 se controlan a través de *jumpers* con la finalidad de concebir otras configuraciones sin realizar cambios significativos en el circuito.

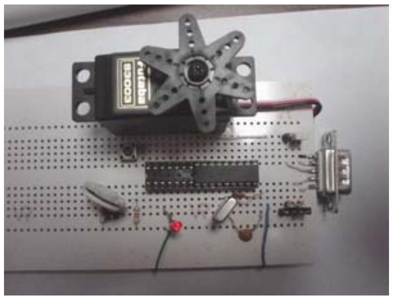


Fig. 4. Prototipo con PIC16F628.

Originalmente, este prototipo se diseñó para soportar un PIC16F73 que contiene internamente 4 canales de conversión A/D en correspondencia a las necesidades del proyecto anteriormente referido [1]; en el caso del PIC16F628 se desconectó el oscilador externo a través de los mismos *jumpers* dado que este dispositivo incorpora un oscilador interno. El prototipo resulta compatible para ambos microcontroladores.

El microcontrolador *PIC16F628* de *Microchip* es un dispositivo CMOS FLASH de 8 bits con arquitectura RISC, capaz de operar con frecuencias de reloj hasta de 20 MHz. Posee internamente un oscilador de 4 MHz y un circuito *Power-On Reset.* Ofrece dos puertos de datos con un total de 16 líneas I/O de propósito general.

Adicionalmente, el PIC16F628 proporciona una memoria de datos EEPROM de 128x8, una memoria de programa FLASH de 2024x14, una memoria de datos RAM de propósito general de 224x8, un módulo de captura/comparación/PWM, un USART, 2 comparadores análogos, una referencia de voltaje programable y tres temporizadores. Se recomienda revisar la hoja de especificaciones de este circuito integrado para complementar la información.

III. INTERFAZ DE USUARIO

Como se mencionó con anterioridad las interfaces para acceso al puerto serie de la iPAQ 3950 se programaron en *Embedded Visual Basic*. Se utilizó el control MSCOMM (control tipo Active X) con la opción *a disparo*, es decir, al depositar tanto para recibir como para enviar datos.

En el caso de recibir datos provenientes del microcontrolador, un byte en el buffer del puerto automáticamente dispara el evento correspondiente.

MSCOMM incorpora todas las funciones para configurar el puerto; sus propiedades más importantes son las siguientes:

ComPort: Activa y regresa el número del puerto serial (Comm1, Comm2).

PortOpen: Activa y regresa el acceso al puerto.

Input: Regresa los caracteres del buffer receptor.

Output: Escribe una cadena sobre el buffer Transmisor.

Settings: Activa y regresa la razón de Baudios, paridad, número de bits, bits de paro. En el caso particular de este trabajo se configuró la cadena *2400, n, 8, 1,* con *Handshaking* puesto a cero, debido a que no se realiza ningún control sobre el flujo de información.

Dentro del programa escrito para este proyecto, se hace referencia al control MSCOMM a través del objeto declarado como *Comm1*. Una aproximación al procedimiento inicial del programa se lista a continuación.

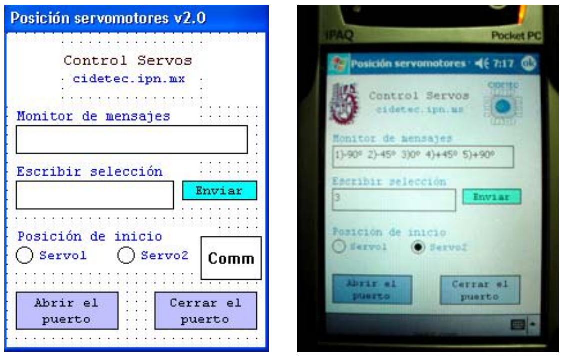
```
Private Sub Adquirir_Click()
  Comm1.PortOpen = True
  Comm1.RThreshold = 1
  Comm1.SThreshold = 0
  Comm1.InputLen = 0
  Comm1.DTREnable = False
End Sub
```

El objeto *Comm1* responde al evento *OnComm*, el cual genera una interrupción, indicando cuando hay comunicación o si algún error ha ocurrido en la transferencia de la información.

Una vez abierto el puerto, se procede a interactuar con el microcontrolador enviando y recibiendo palabras de datos. Las instrucciones básicas se listan a continuación, sólo de manera indicativa.

```
Comm1.Output = DatoEnviar.Text & vbCr
ValorLeido = Comm1.Input
```

La Fig. 5 muestra la interfaz para los servomotores, tanto en tiempo de diseño como en tiempo de ejecución. En esta aplicación se utiliza un monitor de mensajes y una caja de texto para validar una acción. Así mismo, es posible seleccionar el servomotor a operar.



1)-90º 2)-45º 3)0º 4)+45º 5)+90º

3

Fig. 5. Interfaz para los servomotores en tiempo de diseño y en ejecución

Se puede apreciar en ambos casos que es posible abrir y cerrar el puerto en cualquier momento para controlar la comunicación con el microcontrolador.

IV. Servomotores y Programación del Microcontrolador

Los servomotores utilizados son de la marca *Futaba* modelo *s3003* (Fig. 6), con un rango de movimiento de 0º a 180º. Para posicionar el eje de este servomotor se requieren trenes de pulsos con anchos de 0.3 ms. y hasta 2.3 ms. para conseguir el máximo ángulo.

Fig. 6. Servomotor *Futaba s3003*.

El microcontrolador se encarga de recibir serialmente el dato proveniente del PDA en formato estándar binario (también podría enviarse en formato ASCII) con una velocidad predeterminada de 2400 baudios, sin paridad y con un bit de paro. A través de una palabra codificada es posible controlar las condiciones del servomotor, antes mencionadas.

El programa fuente escrito en lenguaje de alto nivel *PicBasic*, se lista parcialmente a continuación [7]. En éste, entiéndase *serin* como la instrucción que permite al microcontrolador recibir datos provenientes del puerto serie del PDA y *serout* como la contraparte que permite enviar datos del microcontrolador hacia el PDA. Para simplificar el código sólo se atañe un servomotor; el código se hace extensivo en el caso de los dos motores indicados en el diagrama de la Fig. 7. En este mismo diagrama, el monitor serial se sustituye por el puerto serie del PDA en cuestión. Las líneas 0 y 1 del puerto B se utilizan para controlar los servomotores.

```
aux = 230       'Con 1 Ms, extrema izq.
servo1: pause 500
PORTB.1 = 0
letra:
pause 1000
serout PORTA.2, N2400, ["Introduce Posición",13,10]
serout PORTA.2, N2400, ["1)-90º  2)-45º  3)0º  4)+45º
                                              5)+90º",13,10]
serin PORTA.0, N2400, tiempo
if (tiempo < $31 OR tiempo > $35) then
serout PORTA.2, N2400, ["Sólo valores entre 1 y 5",13,10]
pause 250
goto letra
else
call deco
aux = dato
pause 250
endif
envia:  if dato = 231 then goto regre
dato=dato + 1
pulsout PORTB.1, dato
pause 20
goto envia
regre:  if dato = aux then goto letra
```

```
dato = dato - 1
pulsout PORTB.1, dato
pause 20
goto regre

deco:
select case tiempo
Case "1":  dato=30  '30 x 10 uS, futaba s3003, 0.3 ms a 2.3 ms; considerando Osc de 4MHz
CASE "2":  dato=80
CASE "3":  dato=130
CASE "4":  dato=180
CASE "5":  dato=230
end select
```

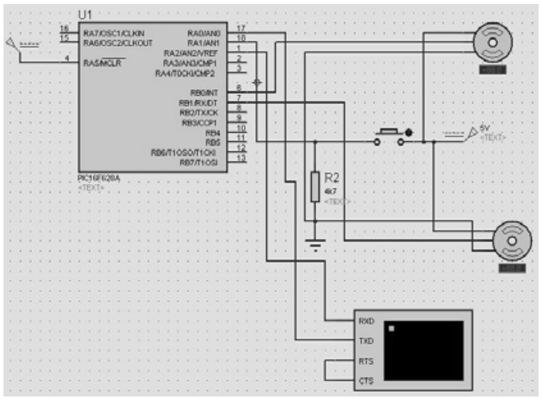


Fig. 7. Diagrama de conexiones.

V. Pruebas y Resultados

El microcontrolador con las rutinas seriales se utilizó primeramente con una PC de escritorio para verificar el funcionamiento correcto de la electrónica de la interfaz para los motores; posteriormente se comprobó el funcionamiento en el PDA ejecutando la aplicación generada en *Embedded Visual Basic* para *Windows Pocket 2002*. El sistema de control básico funcionó correctamente para el caso de un solo servomotor y para la situación extendida hacia dos servomotores.

En la Fig. 8 es posible apreciar al PDA empotrado en su cuna de sincronización y el prototipo en operación. El desarrollo final infiere un PIC16F628.

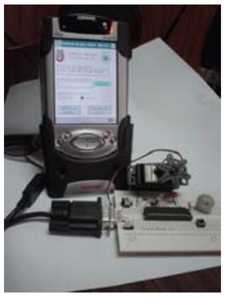


Fig. 8. Prototipo en operación con cuna de sincronización y PIC16F73 (sólo para comprobar compatibilidad).

También se realizaron pruebas con un PDA modelo *HP iPAQ hx2490b*, con sistema operativo *Windows Mobile 5.0*. En este caso, no se programó una interfaz de usuario, sino que se recurrió al uso de una *hyperterminal*. En la Fig. 9, se aprecia la *hyperterminal* en funcionamiento, así como la evolución del prototipo hacia un diseño más robusto con el PIC16F628.

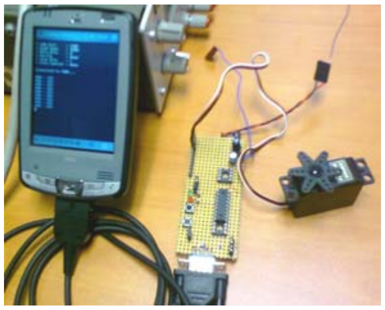


Fig. 9. iPAQ hx2490b con Windows Mobile 5.0 y prototipo modificado (PIC16F628).

Se montó un pequeño laboratorio de pruebas, tal y como se puede apreciar en la Fig. 10, en el cual se utilizó un osciloscopio para medir los anchos de los pulsos generados por el PIC y que posicionan los ejes de los servomotores.

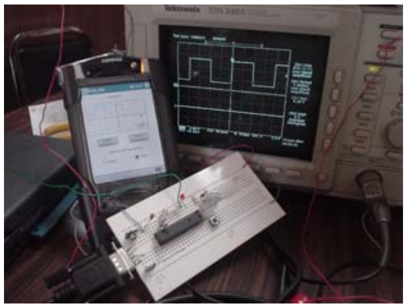


Fig. 10. Aspecto general del laboratorio de pruebas para el prototipo.

VI. Conclusiones

Como se menciona a lo largo de este trabajo, para algunas aplicaciones que no infieren alta complejidad de procesamiento, el PDA es un recurso sustentable para implementar sistemas simples de control, asumiendo que quien realiza el grueso del procesamiento es el microcontrolador, y la computadora de bolsillo se limita a la supervisón y monitoreo de resultados. En el CIDETEC, estas primeras aproximaciones se han aplicado con éxito en el control de aplicaciones robóticas con pocos grados de libertad, en donde cada grado está constituido propiamente por un servomotor.

La transmisión serial es sumamente confiable y sencilla de implementar; además de que actualmente se ha migrado hacia la comunicación inalámbrica utilizando el principio de la transmisión serial convencional, como sucede con el puerto *Bluetooth*, lo que aumenta los alcances futuros del prototipo

Uno de los principales objetivos planteados al inicio de este proyecto, fue el de utilizar el puerto serie de la *iPAQ Pocket PC* para recibir datos de un hardware externo. En el caso del sistema diseñado, el microcontrolador utilizado redujo drásticamente la complejidad de la comunicación entre el PDA y los servomotores; éste se conecta directamente a la iPAQ sin necesidad de un acoplador de nivel de voltaje (por ejemplo, el C.I. MAX232). De manera confiable se puede generar el comando para habilitar el tren de pulsos que posiciona con exactitud el rotor de los motores.

REFERENCIAS

[1] J. C. Herrera Lozada, I. Rivera Zárate, A. Cruz Contreras. Monitor de Señales en un PDA. En: Proceeding XXVII International Congress of Electronic Engineering ELECTRO 2005. División de Estudios de Posgrado, ITCH, pp. 267 – 271.

[2] M. Mazo, J. Ureña, F. J. Rodríguez, J. J. García, F. Espinosa, J. L. Lázaro, J.C. García. Teaching Equipment for Training in the control of DC, Brushless and Stepper Servomotors. IEEE Trans. On Education, vol. 41, nº2, 1998, pp 146-158.

[3] P. Cominos, N. Munro. PID controllers: recent tuning methods and design to specification. In: IEEE Proceedings: Control Theory and Applications, vol. 149, no. 1, 2002, pp. 46–53.

[4] S. N. Vukosavic, M. R. Stojic. Suppression of torsional oscillations in a high-performance speed servo drive. IEEE Transactions on Industrial Electronics, vol. 45, no. 1, 1998, pp. 108 – 117.

[5] Nilas, P.; Sueset, T.; Muguruma, K. A PDA-based high-level human-robot interaction. In: Robotics, Automation and Mechatronics, 2004 IEEE Conference on Volume 2, 1-3 Dec. 2004, pp1158 – 1163.

[6] J. C. Herrera Lozada, J. C. González Robles, A. Cruz Contreras. Programación de Dispositivos de Cómputo Móviles. POLIBITS, Año XV, Vol. 1, Número 31, Enero – Junio de 2005. CIDETEC, México.

[7] J. Iovine. PIC Microcontroller Project Book. Mc. Graw Hill, 2004, 272 p.

Diseño de un Coprocesador Matemático de Precisión Simple usando el Spartan 3E

J. Antonio Álvarez y Michael Lindig B.

***Resumen*—Una Unidad de Punto Flotante (*Floating Point Unit* en inglés) o, más comúnmente conocido como coprocesador matemático, es un componente de la CPU especializado en las operaciones de punto flotante. Las operaciones básicas que toda FPU puede realizar son las aritméticas (suma y multiplicación), si bien algunos sistemas más complejos son capaces también de realizar cálculos trigonométricos o exponenciales. No todas las CPUs tienen una FPU dedicada. En ausencia de FPU, la CPU puede utilizar programas en microcódigo para emular una función en punto flotante a través de la unidad aritmético-lógica (ALU), la cual reduce el costo del hardware a cambio de una sensible pérdida de velocidad. El objetivo de este articulo, es mostrar como puede ser implementado un coprocesador matemático utilizando VHDL, para su implementación en cualquier FPGA.**

***Palabras Clave*—FPU, coprocesador matemático, VHDL, FPGA.**

DESIGN OF MATHEMATICAL COPROCESSOR OF SIMPLE PRECISION USING SPARTAN 3E

***Abstract*—Floating Point Unit (FPU) is also known as mathematical coprocessor and is a specialized component of the CPU dedicated to floating point operations. Basic operations of any FPU are arithmetic (sum and multiplication), though some more complex systems are also able to perform trigonometric or exponential calculations. Not all CPUs have an additional FPU. If there is no FPU present, then CPU can use some programs written in microcode for emulation of floating point operations using arithmetic-logical unit (ALU). This reduces the cost of the hardware but slow down the processing speed. The purpose of this paper is to propose an implementation of the mathematical coprocessor using VHDL, for its further implementation in FPGA.**

***Index Terms*—FPU, mathematical coprocessor, VHDL, FPGA.**

I. INTRODUCCIÓN

La primera computadora personal comercial, fue inventada por IBM en 1981. En su interior había un microprocesador de número 8088, de una empresa llamada Intel. Las características de velocidad de este dispositivo resultan risibles hoy en día: un chip de 8 bits trabajando a 4,77 MHz, aunque bastante razonables para una época en la que el chip de moda era el Z80 de Zilog, como microprocesador común en los sistemas Spectrum que fueron muy populares gracias sobre todo a los juegos increíbles que podían ejecutarse con más gracia y arte que muchos juegos actuales para Pentium.

El 8088 fue una versión de capacidades reducidas del 8086, el cual creo la serie 86 que se integro para los siguientes chips Intel: el 80186 (Control de periféricos), el 80286 (16 bits y hasta 20 MHz) y por fin, en 1987, el primer microprocesador de 32 bits, el 80386, llamado simplemente 386.

Al ser de 32 bits permitía idear software más moderno, con funcionalidades como multitarea real, es decir, disponer de más de un programa trabajando a la vez. A partir de entonces todos los chips compatibles Intel han sido de 32 bits, incluso el flamante Pentium II.

Un día llegó el microprocesador 80486 [1] , que era un microprocesador 80386 con un coprocesador matemático incorporado y una memoria caché integrada, lo que le hacía más rápido; desde entonces todos los chips tienen ambos en su interior, en la Fig. 1 se observa una computadora 486.

Fig. 1. Muestra una Microprocesador Amd modelo 486 (Un módelo 386 con Coprocesador Matemático (FPU)

II. EL COPROCESADOR MATEMÁTICO

El coprocesador matemático es un componente electrónico, que esta diseñado para que funcione en paralelo con el microprocesador. El conjunto de instrucciones incluye muchas operaciones extremadamente potentes para la operación de los datos en punto flotante.

El coprocesador trabaja internamente sólo en formato real, por lo que cualquier carga en los registros de coprocesador provocará que dicho valor sea convertido a punto flotante.

Sus registros están estructurados en forma de pila y se accede a ellos por el número de entrada que ocupan en la pila.

Los registros van desde R(0) hasta R(7), en total ocho registros de 80bits, como deben de manejarse en formato de pila, el coprocesador tiene un puntero de control de pila

Manuscrito recibido el 10 de mayo del 2008. Manuscrito aceptado para su publicación el 2 de septiembre del 2008.

J. A Alvarez Cedillo, Centro de Innovación y Desarrollo Tecnológico en Cómputo del Instituto Politécnico Nacional, México, D. F. (teléfono: 57296000 Ext. 52536; e-mail: jaalvarez@ipn.mx).

Michael Lindig B, Dirección de Cómputo y Telecomunicaciones del Instituto Politécnico Nacional, México, D. F. (e-mail: mlindig@ipn.mx).

llamado **St (state,estado)**, Toda interacción que tengamos que hacer con los registros del coprocesador se realiza a través del puntero de pila **St**, donde el último valor introducido es St o St(0) y si hubiéramos rellenado todos los registros el ultimo seria St(7)

Tipos de datos

El coprocesador puede obtener y escribir datos en memoria de los siguientes tipos.

- Entero
  - Words(16bits)
  - Dword(32 bits)
  - Qwords(64 bits)
- Real
  - Words(16 bits)
  - Dword(32 bits)
  - Qwords(64 bits)
  - Twords(80 bits)

Cada elemento de la pila puede almacenar un valor en formato real temporal de 10 bytes (80bits).

El puntero de la pila (ST) indica en todo momento el elemento en la cima. Puede valer entre 0 y 7.

Las instrucciones del 8087 pueden trabajar con datos en memoria, en el siguiente ejemplo se muestra código en ensamblador que explota esta capacidad:

```
finit
fild multiplicando
fild multiplicador
fmul st,st(1)
fist resultado
```

Las instrucciones del 8087 también pueden trabajar directamente con datos en la pila, en el siguiente ejemplo se muestra código en ensamblador que explota esta capacidad:

```
finit
fild multiplicando
fimul multiplicador
fist resultado
```

Todas las instrucciones del 8087 comienzan con la letra “F”. Si acaban con la letra “P” quiere decir que la pila se cargará con el resultado. Por ejemplo, FISTP VAR saca el resultado de la cima de la pila, lo guarda en la variable en memoria y lo quita de la pila (mueve el puntero).

Tipos de instrucciones:

Existen diferentes tipos de instrucciones, estas se encuentran clasificadas de acuerdo a una función primaria, estas funciones son las siguientes:

- • De transferencia de datos
- • Aritméticas
- • De comparación
- • De cálculo de funciones transcendentes
- • Relativas a constantes
- • De control

Para utilizar el 8087, el 8086 debe ejecutar la instrucción de inicialización FINIT.

El siguiente ejemplo en ensamblador, multiplica dos variables donde el cálculo de resultado es igual a la var1 * var2.

```
pila segment stack 'stack'
dw 100h dup (?)
pila ends
datos segment 'data'
var1 dw 7
var2 dw 3
resultado dw 0
datos ends
codigo segment 'code'
assume cs:codigo, ds:datos, ss:pila
main PROC
mov ax,datos
mov ds,ax
finit
fild var1
fimul var2
fist resultado
mov ax,resultado
call escribir_numero
mov ax,4C00h
int 21h
main ENDP
escribir_numero PROC NEAR
push ax
push dx
mov bx,10
mov dl,al
cmp ax,bx
jb escribir_resto
sub dx,dx
div bx
call escribir_numero
escribir_resto:
add dl,'0'
mov ah,2
int 21h
pop dx
pop ax
ret
escribir_numero ENDP
codigo ends
END main
```

III. Implementación en VHDL

VHDL es el acrónimo que representa la combinación de los conceptos VHSIC y HDL, donde VHSIC es el acrónimo de *Very High Speed Integrated Circuit* y HDL es a su vez el acrónimo de *Hardware Description Language* [2].

Es un lenguaje estándar definido por la IEEE (*Institute of Electrical and Electronics Engineers*), ANSI/IEEE 1076-1993 que se usa para diseñar circuitos digitales. Otros métodos para diseñar circuitos son la captura de esquemas con herramientas CAD y los diagramas de bloques, pero éstos no son prácticos en diseños complejos. Otros lenguajes para el mismo propósito son Verilog y ABEL. Se usa principalmente para programar PLD (*Programable Logic Device* - Dispositivo Lógico Programable) y FPG] (*Field Programmable Gate Array*) [3]. En la Fig. 2 se muestra un *kit* de desarrollo Spartan 3).

El objetivo de este proyecto es diseñar una FPU de precisión simple usando VHDL, simularlo y sintetizarlo. Contara con las operaciones adición de punto sólo flotante, la substracción y la multiplicación. [4]

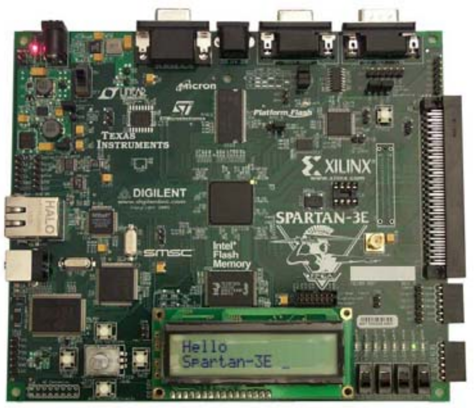


Fig. 2. *Kit* de desarrollo de Spartan 3.

Descripciones de Señal
Entradas:
clk reloj
opa y opb .- Entradas de los operandos A y B
rmode.- Redondeo (00 redondea al más cercano, 01 redondea a cero, 10 redondea a inf, 11 redondea a-inf)
fpu_op .- Operaciones de punto flotante
0 – Suma
1 – Resta
2 -Multiplicación
3 -División
4 -Int.- Convierte en entero,
5 - Flot.- Conversión a int.
Salidas:
fout - salida del resultado
inf.- Es el valor especial INF
ine -Calculo inexacto
overflow .- Salida de *overflow*, por ejemplo el número es mas largo de lo que puede ser representado.
div_by_zero .- División sobre cero.
snan .- SNAN
qnan .- QNAN

La Fig. 3 muestra la entidad del proyecto a construir.

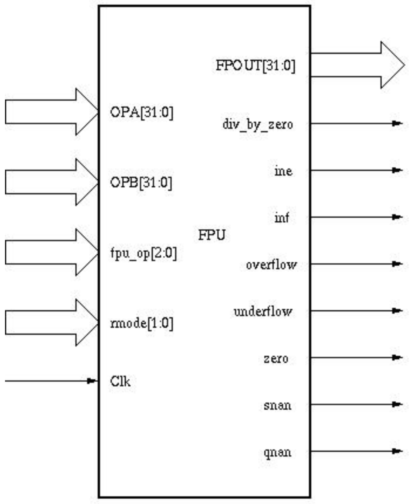


Fig. 3. Muestra una Microprocesador Amd modelo 486 (Un módelo 386 con Coprocesador Matemático (FPU)

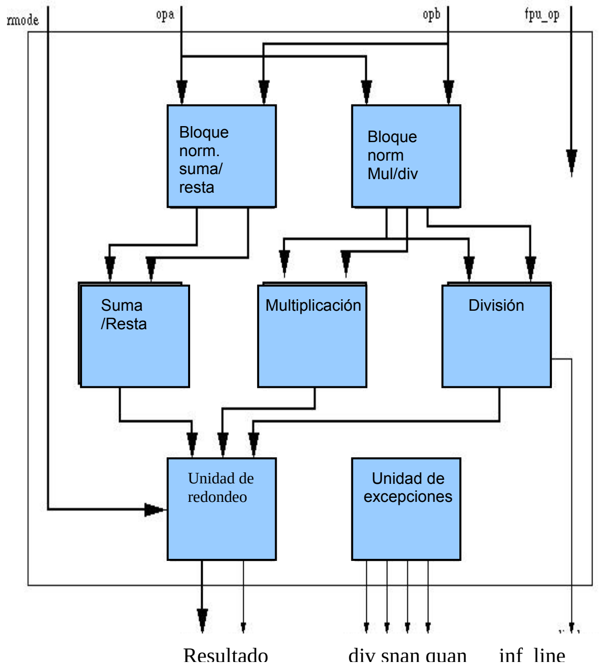


Fig. 4. Muestra el núcleo del proyecto [5].

El núcleo de la FPU (Fig. 4) está formado por las siguientes unidades.

Un bloque normalizado de suma y resta.- Calcula la diferencia entre el exponente mas grande y el pequeño. Ajusta la fracción pequeña por la derecha y determina si la operación es una suma o una resta, después resuelve los bits del signo

Un bloque normalizado de multiplicación y división.- Conmuta la suma o resta de exponentes, checa si existe un exponente en *overflow*, o la condición de *underflow* y el valor INF sobre una entrada.

Unidad de redondeo – Normaliza la fracción y el exponente. Todos los redondeos los hace en paralelo y selecciona la salida correspondiente.

Unidad de Excepciones – Genera y maneja las excepciones.

El diagrama de la Fig. 3 muestra el bloque general de circuitos dentro de la integración del proyecto.

En la Fig. 5 se muestra el *datapath* y el *pipeline*:

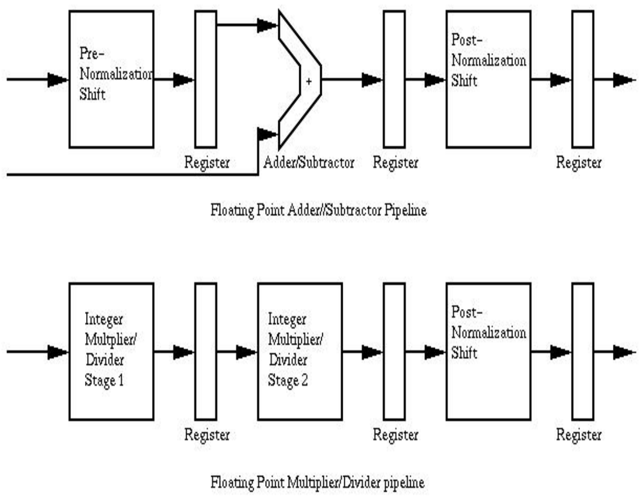


Figura 5.- *Datapath* y *pipeline*

El código en VHDL general es el siguiente:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_misc.ALL;
USE ieee.std_logic_unsigned.ALL;

LIBRARY work;

---------------------------------------------------------------------------
-- FPU Operations (fpu_op):
-- 0 = add
-- 1 = sub
-- 2 = mul
-- 3 = div
-- 4 =
-- 5 =
-- 6 =
-- 7 =
---------------------------------------------------------------------------

---------------------------------------------------------------------------
-- Rounding Modes (rmode):

-- 0 = round_nearest_even
-- 1 = round_to_zero
-- 2 = round_up
-- 3 = round_down
---------------------------------------------------------------------------

ENTITY fpu IS
   PORT(
      clk         : IN     std_logic  ;
      fpu_op      : IN     std_logic_vector (2 downto 0) ;
      opa         : IN     std_logic_vector (31 downto 0) ;
      opb         : IN     std_logic_vector (31 downto 0) ;
      rmode       : IN     std_logic_vector (1 downto 0) ;
      div_by_zero : OUT    std_logic  ;
      fpout       : OUT    std_logic_vector (31 downto 0) ;
      ine         : OUT    std_logic  ;
      inf         : OUT    std_logic  ;
      overflow    : OUT    std_logic  ;
      qnan        : OUT    std_logic  ;
      snan        : OUT    std_logic  ;
      underflow   : OUT    std_logic  ;
      zero        : OUT    std_logic
   );
END fpu ;

ARCHITECTURE arch OF fpu IS
    signal opa_r, opb_r : std_logic_vector (31 downto 0);
    signal signa, signb : std_logic ;
    signal sign_fasu : std_logic ;
    signal fracta, fractb : std_logic_vector (26 downto 0);
    signal exp_fasu : std_logic_vector (7 downto 0);
    signal exp_r : std_logic_vector (7 downto 0);
    signal fract_out_d : std_logic_vector (26 downto 0);
    signal co : std_logic ;
    signal fract_out_q : std_logic_vector (27 downto 0);
    signal out_d : std_logic_vector (30 downto 0);
    signal overflow_d, underflow_d : std_logic ;
    signal mul_inf, div_inf : std_logic ;
    signal mul_00, div_00 : std_logic ;
    signal inf_d, ind_d, qnan_d, snan_d, opa_nan, opb_nan : std_logic ;
    signal opa_00, opb_00 : std_logic ;
    signal opa_inf, opb_inf : std_logic ;
    signal opa_dn, opb_dn : std_logic ;
    signal nan_sign_d, result_zero_sign_d : std_logic ;
    signal sign_fasu_r : std_logic ;
    signal exp_mul : std_logic_vector (7 downto 0);
    signal sign_mul : std_logic ;
    signal sign_mul_r : std_logic ;
    signal fracta_mul, fractb_mul : std_logic_vector (23 downto 0);
    signal inf_mul : std_logic ;
    signal inf_mul_r : std_logic ;
    signal exp_ovf : std_logic_vector (1 downto 0);
    signal exp_ovf_r : std_logic_vector (1 downto 0);
    signal sign_exe : std_logic ;
    signal sign_exe_r : std_logic ;
    signal underflow_fmul1_p1, underflow_fmul1_p2, underflow_fmul1_p3 : std_logic ;
    signal underflow_fmul_d : std_logic_vector (2 downto 0);
    signal prod : std_logic_vector (47 downto 0);
    signal quo : std_logic_vector (49 downto 0);
    signal fdiv_opa : std_logic_vector (49 downto 0);
    signal remainder : std_logic_vector (49 downto 0);
    signal remainder_00 : std_logic ;
    signal div_opa_ldz_d, div_opa_ldz_r1, div_opa_ldz_r2 : std_logic_vector (4 downto
0);
    signal ine_d : std_logic ;
    signal fract_denorm : std_logic_vector (47 downto 0);
    signal fract_div : std_logic_vector (47 downto 0);
    signal sign_d : std_logic ;
    signal sign : std_logic ;
    signal opa_r1 : std_logic_vector (30 downto 0);
    signal fract_i2f : std_logic_vector (47 downto 0);
    signal opas_r1, opas_r2 : std_logic ;
    signal f2i_out_sign : std_logic ;
    signal fasu_op_r1, fasu_op_r2 : std_logic ;
    signal out_fixed : std_logic_vector (30 downto 0);
    signal output_zero_fasu : std_logic ;
    signal output_zero_fdiv : std_logic ;
    signal output_zero_fmul : std_logic ;
    signal inf_mul2 : std_logic ;
    signal overflow_fasu : std_logic ;
    signal overflow_fmul : std_logic ;
    signal overflow_fdiv : std_logic ;
    signal inf_fmul : std_logic ;
    signal sign_mul_final : std_logic ;
    signal out_d_00 : std_logic ;
    signal sign_div_final : std_logic ;
    signal ine_mul, ine_mula, ine_div, ine_fasu : std_logic ;
    signal underflow_fasu, underflow_fmul, underflow_fdiv : std_logic ;
    signal underflow_fmul1 : std_logic ;
    signal underflow_fmul_r : std_logic_vector (2 downto 0);
    signal opa_nan_r : std_logic ;
    signal mul_uf_del : std_logic ;
    signal uf2_del, ufb2_del, ufc2_del, underflow_d_del : std_logic ;
    signal co_del : std_logic ;
    signal out_d_del : std_logic_vector (30 downto 0);
    signal ov_fasu_del, ov_fmul_del : std_logic ;
    signal fop : std_logic_vector (2 downto 0);
    signal ldza_del : std_logic_vector (4 downto 0);
    signal quo_del : std_logic_vector (49 downto 0);
    signal rmode_r1, rmode_r2, rmode_r3 : std_logic_vector (1 downto 0);
    signal fpu_op_r1, fpu_op_r2, fpu_op_r3 : std_logic_vector (2 downto 0);
    signal fpu_op_r1_0_not : std_logic ;
    signal fasu_op, co_d : std_logic ;
    signal post_norm_output_zero : std_logic ;

    CONSTANT INF_VAL : std_logic_vector(31 DOWNTO 0) := X"7f800000";
    CONSTANT QNAN_VAL : std_logic_vector(31 DOWNTO 0) := X"7fc00001";
    CONSTANT SNAN_VAL : std_logic_vector(31 DOWNTO 0) := X"7f800001";

    COMPONENT add_sub27
       PORT(
          add : IN     std_logic  ;
          opa : IN     std_logic_vector (26 downto 0) ;
          opb : IN     std_logic_vector (26 downto 0) ;
          co  : OUT    std_logic  ;
          sum : OUT    std_logic_vector (26 downto 0)
       );
    END COMPONENT;
```

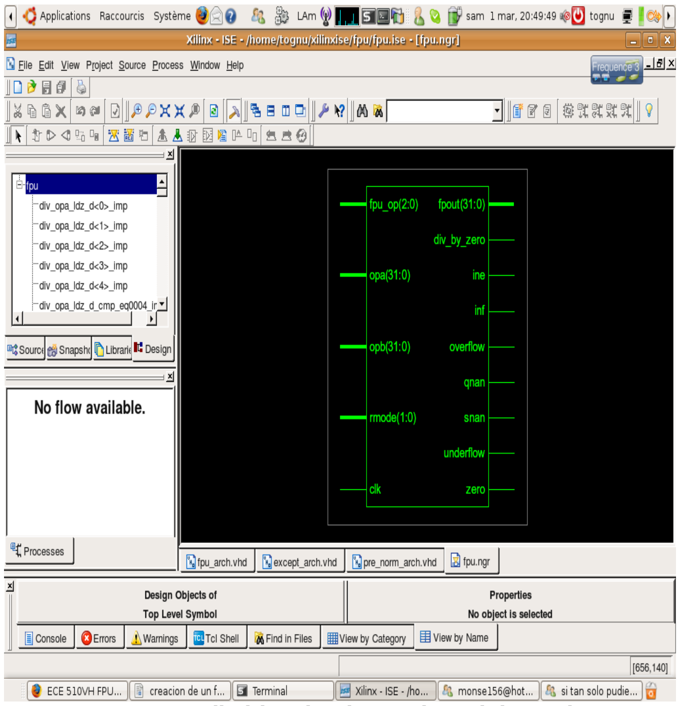
```
COMPONENT div_r2
  PORT(
    clk       : IN     std_logic  ;
    opa       : IN     std_logic_vector (49 downto 0) ;
    opb       : IN     std_logic_vector (23 downto 0) ;
    quo       : OUT    std_logic_vector (49 downto 0) ;
    remainder : OUT    std_logic_vector (49 downto 0)
  );
END COMPONENT;

COMPONENT except IS
  PORT(
    clk     : IN     std_logic  ;
    opa     : IN     std_logic_vector (31 downto 0) ;
    opb     : IN     std_logic_vector (31 downto 0) ;
    ind     : OUT    std_logic  ;
    inf     : OUT    std_logic  ;
    opa_00  : OUT    std_logic  ;
    opa_dn  : OUT    std_logic  ;
    opa_inf : OUT    std_logic  ;
    opa_nan : OUT    std_logic  ;
    opb_00  : OUT    std_logic  ;
    opb_dn  : OUT    std_logic  ;
    opb_inf : OUT    std_logic  ;
    opb_nan : OUT    std_logic  ;
    qnan    : OUT    std_logic  ;
    snan    : OUT    std_logic
  );
END COMPONENT ;

COMPONENT mul_r2 IS
  PORT(
    clk  : IN     std_logic  ;
    opa  : IN     std_logic_vector (23 downto 0) ;
    opb  : IN     std_logic_vector (23 downto 0) ;
    prod : OUT    std_logic_vector (47 downto 0)
  );
END COMPONENT;

COMPONENT post_norm IS
  PORT(
    clk          : IN     std_logic  ;
    div_opa_ldz  : IN     std_logic_vector (4 downto 0) ;
    exp_in       : IN     std_logic_vector (7 downto 0) ;
    exp_ovf      : IN     std_logic_vector (1 downto 0) ;
    fpu_op       : IN     std_logic_vector (2 downto 0) ;
    fract_in     : IN     std_logic_vector (47 downto 0) ;
    opa_dn       : IN     std_logic  ;
    opas         : IN     std_logic  ;
    opb_dn       : IN     std_logic  ;
    output_zero  : IN     std_logic  ;
    rem_00       : IN     std_logic  ;
    rmode        : IN     std_logic_vector (1 downto 0) ;
    sign         : IN     std_logic  ;
    f2i_out_sign : OUT    std_logic  ;
    fpout        : OUT    std_logic_vector (30 downto 0) ;
    ine          : OUT    std_logic  ;
    overflow     : OUT    std_logic  ;
    underflow    : OUT    std_logic
  );
END COMPONENT;

COMPONENT pre_norm IS
  PORT(
    add              : IN     std_logic  ;
    clk              : IN     std_logic  ;
    opa              : IN     std_logic_vector (31 downto 0) ;
    opa_nan          : IN     std_logic  ;
    opb              : IN     std_logic_vector (31 downto 0) ;
    opb_nan          : IN     std_logic  ;
    rmode            : IN     std_logic_vector (1 downto 0) ;
    exp_dn_out       : OUT    std_logic_vector (7 downto 0) ;
    fasu_op          : OUT    std_logic  ;
    fracta_out       : OUT    std_logic_vector (26 downto 0) ;
    fractb_out       : OUT    std_logic_vector (26 downto 0) ;
    nan_sign         : OUT    std_logic  ;
    result_zero_sign : OUT    std_logic  ;
    sign             : OUT    std_logic
  );
END COMPONENT;

COMPONENT pre_norm_fmul IS
  PORT(
    clk       : IN     std_logic  ;
    fpu_op    : IN     std_logic_vector (2 downto 0) ;
    opa       : IN     std_logic_vector (31 downto 0) ;
    opb       : IN     std_logic_vector (31 downto 0) ;
    exp_out   : OUT    std_logic_vector (7 downto 0) ;
    exp_ovf   : OUT    std_logic_vector (1 downto 0) ;
    fracta    : OUT    std_logic_vector (23 downto 0) ;
    fractb    : OUT    std_logic_vector (23 downto 0) ;
    inf       : OUT    std_logic  ;
    sign      : OUT    std_logic  ;
    sign_exe  : OUT    std_logic  ;
    underflow : OUT    std_logic_vector (2 downto 0)
  );
END COMPONENT;

BEGIN

  PROCESS (clk)
  BEGIN
    IF clk'event AND clk = '1' THEN
      opa_r <=  opa;
      opb_r <=  opb;
      rmode_r1 <=  rmode;
      rmode_r2 <=  rmode_r1;
      rmode_r3 <=  rmode_r2;
      fpu_op_r1 <=  fpu_op;
      fpu_op_r2 <=  fpu_op_r1;
      fpu_op_r3 <=  fpu_op_r2;
    END IF;
  END PROCESS;

  ---------------------------------------------------------------------------
  -- Exceptions block
  ---------------------------------------------------------------------------
    u0 : except
      PORT MAP (
        clk => clk,
        opa => opa_r,
        opb => opb_r,
        inf => inf_d,
        ind => ind_d,
        qnan => qnan_d,
        snan => snan_d,
        opa_nan => opa_nan,
        opb_nan => opb_nan,
        opa_00 => opa_00,
        opb_00 => opb_00,
        opa_inf => opa_inf,
        opb_inf => opb_inf,
        opa_dn => opa_dn,
        opb_dn => opb_dn
        );

  ---------------------------------------------------------------------------
  -- Pre-Normalize block
  -- Adjusts the numbers to equal exponents and sorts them
  -- determine result sign
  -- determine actual operation to perform (add or sub)
  ---------------------------------------------------------------------------
  fpu_op_r1_0_not <= NOT fpu_op_r1(0);
  u1 : pre_norm
    PORT MAP (
      clk => clk,                       -- System Clock
      rmode => rmode_r2,                       -- Roundin Mode
      add => fpu_op_r1_0_not,                    -- Add/Sub Input
      opa => opa_r,
      opb => opb_r,                         -- Registered OP Inputs
      opa_nan => opa_nan,                     -- OpA is a NAN indicator
      opb_nan => opb_nan,                     -- OpB is a NAN indicator
      fracta_out => fracta,                  -- Equalized and sorted fraction
      fractb_out => fractb,                  -- outputs (Registered
      exp_dn_out => exp_fasu,                  -- Selected exponent output (registered;
      sign => sign_fasu,                    -- Encoded output Sign (registered)
      nan_sign => nan_sign_d,                 -- Output Sign for NANs (registered)
      result_zero_sign => result_zero_sign_d,  -- Output Sign for zero result
(registered)
      fasu_op => fasu_op                     -- Actual fasu operation output (registered)
      );

  u2 : pre_norm_fmul
    PORT MAP (
      clk => clk,
      fpu_op => fpu_op_r1,
      opa => opa_r,
      opb => opb_r,
         fracta => fracta_mul,
         fractb => fractb_mul,
         exp_out => exp_mul,      -- FMUL exponent output  => registered)
         sign => sign_mul,        -- FMUL sign output (registered)
         sign_exe => sign_exe,    -- FMUL exception sign output (registered)
         inf => inf_mul,          -- FMUL inf output (registered)
         exp_ovf => exp_ovf,      -- FMUL exponnent overflow output (registered)
         underflow => underflow_fmul_d
         );


   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         sign_mul_r <= sign_mul;
         sign_exe_r <= sign_exe;
         inf_mul_r <= inf_mul;
         exp_ovf_r <= exp_ovf;
         sign_fasu_r <= sign_fasu;
      END IF;
   END PROCESS;


------------------------------------------------------------------------
--
-- Add/Sub
--

 u3 : add_sub27
 PORT MAP (
      add => fasu_op,              -- Add/Sub
      opa => fracta,               -- Fraction A input
      opb => fractb,               -- Fraction B Input
      sum => fract_out_d,          -- SUM output
      co => co_d );                -- Carry Output

   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         fract_out_q <= co_d & fract_out_d;
      END IF;
   END PROCESS;


------------------------------------------------------------------------
--
-- Mul
--

   u5 : mul_r2 PORT MAP (clk => clk, opa => fracta_mul, opb => fractb_mul, prod
=> prod);

------------------------------------------------------------------------
--
-- Divide
--
PROCESS (fracta_mul)
BEGIN
      IF fracta_mul(22) = '1' THEN  div_opa_ldz_d <= conv_std_logic_vector(1,5);
      ELSIF fracta_mul(22 DOWNTO 21) = "01" THEN  div_opa_ldz_d <=
conv_std_logic_vector(2,5);
      ELSIF fracta_mul(22 DOWNTO 20) = "001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(3,5);
      ELSIF fracta_mul(22 DOWNTO 19) = "0001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(4,5);
      ELSIF fracta_mul(22 DOWNTO 18) = "00001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(5,5);
      ELSIF fracta_mul(22 DOWNTO 17) = "000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(6,5);
      ELSIF fracta_mul(22 DOWNTO 16) = "0000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(7,5);
      ELSIF fracta_mul(22 DOWNTO 15) = "00000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(8,5);
      ELSIF fracta_mul(22 DOWNTO 14) = "000000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(9,5);
      ELSIF fracta_mul(22 DOWNTO 13) = "0000000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(10,5);
      ELSIF fracta_mul(22 DOWNTO 12) = "00000000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(11,5);
      ELSIF fracta_mul(22 DOWNTO 11) = "000000000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(12,5);
      ELSIF fracta_mul(22 DOWNTO 10) = "0000000000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(13,5);
      ELSIF fracta_mul(22 DOWNTO 9) = "00000000000001" THEN  div_opa_ldz_d <=
conv_std_logic_vector(14,5);
      ELSIF fracta_mul(22 DOWNTO 8) = "000000000000001" THEN  div_opa_ldz_d
<= conv_std_logic_vector(15,5);
      ELSIF fracta_mul(22 DOWNTO 7) = "0000000000000001" THEN  div_opa_ldz_d
<= conv_std_logic_vector(16,5);
      ELSIF fracta_mul(22 DOWNTO 6) = "00000000000000001" THEN  div_opa_ldz_d
<= conv_std_logic_vector(17,5);
      ELSIF fracta_mul(22 DOWNTO 5) = "000000000000000001" THEN
div_opa_ldz_d <= conv_std_logic_vector(18,5);
      ELSIF fracta_mul(22 DOWNTO 4) = "0000000000000000001" THEN
div_opa_ldz_d <= conv_std_logic_vector(19,5);
      ELSIF fracta_mul(22 DOWNTO 3) = "00000000000000000001" THEN
div_opa_ldz_d <= conv_std_logic_vector(20,5);
      ELSIF fracta_mul(22 DOWNTO 2) = "000000000000000000001" THEN
div_opa_ldz_d <= conv_std_logic_vector(21,5);
      ELSIF fracta_mul(22 DOWNTO 1) = "0000000000000000000001" THEN
div_opa_ldz_d <= conv_std_logic_vector(22,5);
      ELSIF fracta_mul(22 DOWNTO 1) = "0000000000000000000000" THEN
div_opa_ldz_d <= conv_std_logic_vector(23,5);
      ELSE div_opa_ldz_d <= (OTHERS => 'X');
      END IF;
END PROCESS;

      fdiv_opa <= ((SHL(fracta_mul,div_opa_ldz_d)) &
                "00" & X"000000") WHEN
               ((or_reduce(opa_r(30 DOWNTO 23)))='0') ELSE
               (fracta_mul & "00" & X"000000");

      u6 : div_r2 PORT MAP (clk => clk, opa => fdiv_opa, opb => fractb_mul,
                     quo => quo,
                     remainder => remainder);

      remainder_00 <= NOT or_reduce(remainder);

      PROCESS (clk)
      BEGIN
         IF clk'event AND clk = '1' THEN
            div_opa_ldz_r1 <= div_opa_ldz_d;
            div_opa_ldz_r2 <= div_opa_ldz_r1;
         END IF;
      END PROCESS;

------------------------------------------------------------------------
--
-- Normalize Result
--

   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         CASE fpu_op_r2 IS
            WHEN "000" => exp_r <= exp_fasu;
            WHEN "001" => exp_r <= exp_fasu;
            WHEN "010" => exp_r <= exp_mul;
            WHEN "011" => exp_r <= exp_mul;
            WHEN "100" => exp_r <= (others => '0');
            WHEN "101" => exp_r <= opa_r1(30 downto 23);
            WHEN OTHERS  => exp_r <= (others => '0');
         END case;
      END IF;
   END PROCESS;


   fract_div <= quo(49 DOWNTO 2) WHEN (opb_dn = '1') ELSE
             (quo(26 DOWNTO 0) & '0' & X"00000");

   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         opa_r1 <= opa_r(30 DOWNTO 0);
         IF fpu_op_r2="101" THEN
            IF sign_d = '1' THEN
               fract_i2f <= conv_std_logic_vector(1,48)-(X"000000" &
                        (or_reduce(opa_r1(30 downto 23))) &
                        opa_r1(22 DOWNTO 0))-conv_std_logic_vector(1,48);
            ELSE
               fract_i2f <= (X"000000" &
                       (or_reduce(opa_r1(30 downto 23))) &
                       opa_r1(22 DOWNTO 0));
            END IF;
         ELSE
            IF sign_d = '1' THEN
               fract_i2f <= conv_std_logic_vector(1,48) - (opa_r1 & X"0000" & '1');
            ELSE
               fract_i2f <= (opa_r1 & '0' & X"0000");
            END IF;
```

```
        END IF;
      END IF;
   END PROCESS;

   PROCESS (fpu_op_r3,fract_out_q,prod,fract_div,fract_i2f)
   BEGIN
      CASE fpu_op_r3 IS
         WHEN "000" => fract_denorm <= (fract_out_q & X"00000");
         WHEN "001" => fract_denorm <= (fract_out_q & X"00000");
         WHEN "010" => fract_denorm <= prod;
         WHEN "011" => fract_denorm <= fract_div;
         WHEN "100" => fract_denorm <= fract_i2f;
         WHEN "101" => fract_denorm <= fract_i2f;
         WHEN OTHERS  => fract_denorm <= (others => '0');
      END case;
   END PROCESS;



   PROCESS (clk, opa_r(31),opas_r1,rmode_r2,sign_d)
   BEGIN
      IF clk'event AND clk = '1' THEN
         opas_r1 <= opa_r(31);
         opas_r2 <= opas_r1;
         IF rmode_r2="11" THEN
            sign <= NOT sign_d;
         ELSE
            sign <= sign_d;
         END IF;
      END if;
   END PROCESS;

   sign_d <= sign_mul WHEN (fpu_op_r2(1) = '1') ELSE sign_fasu;

   post_norm_output_zero <= mul_00 or div_00;
   u4 : post_norm
   PORT MAP (
      clk => clk,                -- System Clock
      fpu_op => fpu_op_r3,            -- Floating Point Operation
      opas => opas_r2,                -- OPA Sign
      sign => sign,                   -- Sign of the result
      rmode => rmode_r3,              -- Rounding mode
      fract_in => fract_denorm,       -- Fraction Input
      exp_ovf => exp_ovf_r,           -- Exponent Overflow
      exp_in => exp_r,                -- Exponent Input
      opa_dn => opa_dn,               -- Operand A Denormalized
      opb_dn => opb_dn,               -- Operand A Denormalized
      rem_00 => remainder_00,         -- Diveide Remainder is zero
      div_opa_ldz => div_opa_ldz_r2,   -- Divide opa leading zeros count
      output_zero => post_norm_output_zero,  -- Force output to Zero
      fpout => out_d,                 -- Normalized output (un-registered)
      ine => ine_d,                   -- Result Inexact output (un-registered)
      overflow => overflow_d,         -- Overflow output (un-registered)
      underflow => underflow_d,       -- Underflow output (un-registered)
      f2i_out_sign => f2i_out_sign    -- F2I Output Sign
      );

------------------------------------------------------------------------
--
-- FPU Outputs
--

   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         fasu_op_r1 <= fasu_op;
         fasu_op_r2 <= fasu_op_r1;
         IF exp_mul = X"ff" THEN
            inf_mul2 <= '1';
         ELSE
            inf_mul2 <= '0';
         END IF;
      END IF;
   END PROCESS;


   -- Force pre-set values for non numerical output
   mul_inf <= '1' WHEN ((fpu_op_r3="010") and ((inf_mul_r or inf_mul2)='1') and
                    (rmode_r3="00")) else '0';

   div_inf <= '1' WHEN ((fpu_op_r3="011") and
                    ((opb_00 or opa_inf)='1')) ELSE '0';

   mul_00 <= '1' WHEN ((fpu_op_r3="010") and ((opa_00 or opb_00)='1')) ELSE '0';
   div_00 <= '1' WHEN ((fpu_op_r3="011") and ((opa_00 or opb_inf)='1')) else '0';
```

```
   out_fixed <= QNAN_VAL(30 DOWNTO 0) WHEN
               (((qnan_d OR snan_d) OR (ind_d AND NOT fasu_op_r2) OR
                ((NOT fpu_op_r3(2) AND fpu_op_r3(1) AND fpu_op_r3(0)) AND opb_00
AND opa_00) OR
                (((opa_inf AND opb_00) OR (opb_inf AND opa_00 )) AND
                (NOT fpu_op_r3(2) AND fpu_op_r3(1) AND NOT fpu_op_r3(0)))
                )='1')
               ELSE INF_VAL(30 DOWNTO 0);


   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         IF ( ((mul_inf='1') or (div_inf='1') or
            ((inf_d='1') and (fpu_op_r3/="011") and (fpu_op_r3/="101")) or
            (snan_d='1') or (qnan_d='1')) and (fpu_op_r3/="100"))  THEN
            fpout(30 DOWNTO 0) <= out_fixed;
         ELSE
            fpout(30 DOWNTO 0) <= out_d;
         END IF;
      END IF;
   END PROCESS;

   out_d_00 <= NOT or_reduce(out_d);

   sign_mul_final <= NOT sign_mul_r WHEN
                 ((sign_exe_r AND  ((opa_00 AND opb_inf) OR
                          (opb_00 AND opa_inf)))='1')
                 ELSE sign_mul_r;
   sign_div_final <= NOT sign_mul_r WHEN
                 ((sign_exe_r and (opa_inf and opb_inf))='1')
                 ELSE (sign_mul_r or (opa_00 and opb_00));

   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         If ((fpu_op_r3="101") and (out_d_00='1')) THEN
            fpout(31) <= (f2i_out_sign and not(qnan_d OR snan_d) );
         ELSIF ((fpu_op_r3="010") and ((snan_d or qnan_d)='0')) THEN
            fpout(31) <= sign_mul_final;
         ELSIF ((fpu_op_r3="011") and ((snan_d or qnan_d)='0')) THEN
            fpout(31) <= sign_div_final;
         ELSIF ((snan_d or qnan_d or ind_d) = '1') THEN
            fpout(31) <= nan_sign_d;
         ELSIF (output_zero_fasu = '1') THEN
            fpout(31) <= result_zero_sign_d;
         ELSE
            fpout(31) <= sign_fasu_r;
         END IF;
      END IF;
   END PROCESS;


-- Exception Outputs
   ine_mula <= ((inf_mul_r OR inf_mul2 OR opa_inf OR opb_inf) AND
            (NOT rmode_r3(1) AND rmode_r3(0)) and
             NOT ((opa_inf AND opb_00) OR (opb_inf AND opa_00 )) AND
fpu_op_r3(1));

   ine_mul <= (ine_mula OR ine_d OR inf_fmul OR out_d_00 OR overflow_d OR
underflow_d) AND
            NOT opa_00 and NOT opb_00 and NOT (snan_d OR qnan_d OR inf_d);
   ine_div <= (ine_d OR  overflow_d OR  underflow_d) AND NOT (opb_00 OR snan_d
OR qnan_d OR inf_d);
   ine_fasu <= (ine_d OR overflow_d OR underflow_d) AND NOT (snan_d OR qnan_d
OR inf_d);

   PROCESS (clk)
   BEGIN
      IF clk'event AND clk = '1' THEN
         IF fpu_op_r3(2) = '1' THEN
            ine <= ine_d;
         ELSIF fpu_op_r3(1) = '0' THEN
            ine <= ine_fasu;
         ELSIF fpu_op_r3(0)='1' THEN
            ine <= ine_div;
         ELSE
           ine <= ine_mul;
         END IF;
      END IF;
   END PROCESS;

   overflow_fasu <= overflow_d AND NOT (snan_d OR qnan_d OR inf_d);
   overflow_fmul <= NOT inf_d AND
```

```
                (inf_mul_r OR inf_mul2 OR overflow_d) AND
                NOT (snan_d OR qnan_d);

    overflow_fdiv <= (overflow_d AND NOT (opb_00 OR inf_d OR snan_d OR qnan_d));

    PROCESS (clk)
    BEGIN
      IF clk'event AND clk = '1' THEN
        underflow_fmul_r <= underflow_fmul_d;
        IF fpu_op_r3(2) ='1' THEN
          overflow <= '0';
        ELSIF fpu_op_r3(1) = '0' THEN
          overflow <= overflow_fasu;
        ELSIF fpu_op_r3(0) = '1' THEN
          overflow <= overflow_fdiv;
        ELSE
          overflow <= overflow_fmul;
        END IF;
      END IF;
    END PROCESS;

    underflow_fmul1_p1 <= '1' WHEN (out_d(30 DOWNTO 23) = X"00") else '0';
    underflow_fmul1_p2 <= '1' WHEN (out_d(22 DOWNTO 0) = ("000" & X"00000"))
else '0';
    underflow_fmul1_p3 <= '1' WHEN (prod/=conv_std_logic_vector(0,48)) else '0';

    underflow_fmul1 <= underflow_fmul_r(0) or
                 (underflow_fmul_r(1) and underflow_d ) or
                 ((opa_dn or opb_dn) and out_d_00 and (underflow_fmul1_p3) and sign)
or
                  (underflow_fmul_r(2) AND
                  ((underflow_fmul1_p1) or (underflow_fmul1_p2)));

    underflow_fasu <= underflow_d AND  NOT (inf_d or snan_d or qnan_d);
    underflow_fmul <= underflow_fmul1 AND NOT (snan_d or qnan_d or inf_mul_r);
    underflow_fdiv <= underflow_fasu AND NOT opb_00;

    PROCESS (clk)
    BEGIN
      IF clk'event AND clk = '1' THEN
        IF fpu_op_r3(2) = '1' THEN
          underflow <= '0';
        ELSIF fpu_op_r3(1) = '0' THEN
          underflow <= underflow_fasu;
        ELSIF fpu_op_r3(0) = '1' THEN
          underflow <= underflow_fdiv;
        ELSE
          underflow <= underflow_fmul;
        END IF;
        snan <= snan_d;
      END IF;
    END PROCESS;


-- Status Outputs
    PROCESS (clk)
    BEGIN
      IF clk'event AND clk = '1' THEN
        IF fpu_op_r3(2)='1' THEN
          qnan <= '0';
        ELSE
          qnan <= snan_d OR qnan_d OR (ind_d AND NOT fasu_op_r2) OR
                  (opa_00 AND  opb_00 AND
                  (NOT fpu_op_r3(2) AND fpu_op_r3(1) AND fpu_op_r3(0))) OR
                  (((opa_inf AND opb_00) OR (opb_inf AND opa_00 )) AND
                   (NOT fpu_op_r3(2) AND fpu_op_r3(1) AND NOT fpu_op_r3(0)));

        END IF;
      END IF;
    END PROCESS;

    inf_fmul <= (((inf_mul_r OR inf_mul2) AND (NOT rmode_r3(1) AND NOT
rmode_r3(0)))
           OR opa_inf OR opb_inf) AND
          NOT ((opa_inf AND opb_00) OR (opb_inf AND opa_00)) AND
          (NOT fpu_op_r3(2) AND fpu_op_r3(1) AND NOT fpu_op_r3(0));

    PROCESS (clk)
    BEGIN
      IF clk'event AND clk = '1' THEN
        IF fpu_op_r3(2) = '1' THEN
          inf <= '0';
        ELSE
          inf <= (NOT (qnan_d OR snan_d) AND
                  (((and_reduce(out_d(30 DOWNTO 23))) AND
                   NOT (or_reduce(out_d(22 downto 0))) AND
                   NOT(opb_00 AND NOT fpu_op_r3(2) AND fpu_op_r3(1) AND
fpu_op_r3(0))) OR
                  (inf_d AND NOT (ind_d AND NOT fasu_op_r2) AND NOT fpu_op_r3(1))
OR
                  inf_fmul OR
                  (NOT opa_00 AND  opb_00 AND
                   NOT fpu_op_r3(2) AND fpu_op_r3(1) AND fpu_op_r3(0)) or
                  (NOT fpu_op_r3(2) AND fpu_op_r3(1) AND fpu_op_r3(0) AND
                   opa_inf AND  NOT opb_inf)
                  )
                 );
        END IF;
      END IF;
    END PROCESS;


    output_zero_fasu <= out_d_00 AND NOT (inf_d OR snan_d OR qnan_d);
    output_zero_fdiv <= (div_00 OR (out_d_00 AND NOT opb_00)) AND NOT (opa_inf
AND opb_inf) AND
                 NOT (opa_00 AND opb_00) AND NOT (qnan_d OR snan_d);
    output_zero_fmul <= (out_d_00 OR opa_00 OR opb_00) AND
                 NOT (inf_mul_r OR inf_mul2 OR opa_inf OR opb_inf OR
                    snan_d OR qnan_d) AND
                 NOT (opa_inf AND opb_00) AND NOT (opb_inf AND opa_00);

    PROCESS (clk)
    BEGIN
      IF clk'event AND clk = '1' THEN
        IF fpu_op_r3="101" THEN
          zero <= out_d_00 and NOT (snan_d or qnan_d);
        ELSIF fpu_op_r3="011" THEN
          zero <= output_zero_fdiv;
        ELSIF fpu_op_r3="010" THEN
          zero <= output_zero_fmul;
        ELSE
          zero <= output_zero_fasu;
        END IF;
        IF (opa_nan = '0') AND (fpu_op_r2="011") THEN
          opa_nan_r <= '1';
        ELSE
          opa_nan_r <= '0';
        END IF;
        div_by_zero <= opa_nan_r AND NOT opa_00 AND NOT opa_inf AND opb_00;
      END IF;
    END PROCESS;

END arch;
```

En la Fig. 6 se muestra la entidad elaborada.


Xilinx - ISE - /home/tognu/xilinxise/fpu/fpu.ise - [fpu.ngr]

No flow available.

Applications Raccourcis Système
sam 1 mar, 20:49:49
File Edit View Project Source Process Window Help
fpu
div_opa_ldz_d<0>_imp
div_opa_ldz_d<1>_imp
div_opa_ldz_d<2>_imp
div_opa_ldz_d<3>_imp
div_opa_ldz_d<4>_imp
div_opa_ldz_d_cmp_eq0004_ir
fpu_op(2:0)
fpout(31:0)
div_by_zero
opa(31:0)
ine
inf
opb(31:0)
overflow
qnan
rmode(1:0)
snan
underflow
clk
zero
Processes
fpu_arch.vhd
except_arch.vhd
pre_norm_arch.vhd
fpu.ngr
Design Objects of Top Level Symbol
Properties No object is selected
Console Errors Warnings Tcl Shell Find in Files View by Category View by Name
[656,140]
ECE 510VH FPU... creacion de un f... Terminal Xilinx - ISE - /ho... monse156@hot... si tan solo pudie...

Fig. 6. Pantalla del ise de Xilinx con la entidad general

IV. Conclusiones

En este proyecto se mostró las características y el poder existente en el uso de VHDL y su simplicidad para usarlos en proyectos mas robustos para aplicarlos en FPGA, En el caso de la unidad de punto flotante construida, se pudo observar el mismo comportamiento y se implementaron en ellas las operaciones comunes como es la suma , la multiplicación y la división, todas las operaciones de simple precisión y de punto flotante, también se muestra el *pipeline*, estrategias de verificación y síntesis.

Referencias

[1] Rudolf Usselman. Documentation for Floating Point Unit, http://www.opencores.org.
[1] John L. Hennessy and David A. Patterson. Computer Architecture: A Quantitative Approach. 2nd Edition, Morgan Kaufmann, 2007, 708 p.
[2] Peter J. Ashenden. The Designer's Guide to VHDL, Morgan Kaufmann, 1995, 688 p.
[3] Donald E. Thomas and Philip R. Moorby. The Verilog Hardware Description Language, Kluwer Academic Publishers, 2002, 408 p.
[4] Stuart Oberman. Design Issues in High Performance Floating-Point Arithmetic Units. Stanford University, Technical report, 1996.
[5] IEEE, IEEE-754-1985 Standard for binary floating-point arithmetic.

Journal Information and Instructions for Authors

I. Journal Information

"*Polibits*" is a half-yearly research journal published since 1989 by the Center for Technological Design and Development in Computer Science (CIDETEC) of the National Polytechnic Institute (IPN) in Mexico City, Mexico. The journal solicits original research papers in all areas of computer science and computer engineering, with emphasis on applied research.

The journal has double-blind review procedure. It publishes papers in English and Spanish.

Publication has no cost for the authors.

A. Main topics of interest

The journal publishes research papers in all areas of computer science and computer engineering, with emphasis on applied research.

More specifically, the main topics of interest include, though are not limited to, the following:

– Artificial Intelligence
– Natural Language Processing
– Fuzzy Logic
– Computer Vision
– Multiagent Systems
– Bioinformatics
– Neural Networks
– Evolutionary algorithms
– Knowledge Representation
– Expert Systems
– Intelligent Interfaces: Multimedia, Virtual Reality
– Machine Learning
– Pattern Recognition
– Intelligent Tutoring Systems
– Semantic Web
– Database Systems
– Data Mining
– Software Engineering
– Web Design
– Compilers
– Formal Languages
– Operating Systems
– Distributed Systems
– Parallelism
– Real Time Systems
– Algorithm Theory
– Scientific Computing
– High-Performance Computing
– Geo-processing
– Networks and Connectivity
– Cryptography
– Informatics Security
– Digital Systems Design
– Digital Signal Processing
– Control Systems
– Robotics
– Virtual Instrumentation
– Computer Architecture
– other.

B. Indexing

The journal indexing is in process.

II. Instructions for Authors

A. Submission

Papers ready to review are received through the Web submission system www.easychair.org/polibits

The papers can be written in English or Spanish.

Since the review procedure is double-blind, the full text of the papers should be submitted without names and affiliations of the authors and without any other data that reveals the authors' identity.

For review, a file in one of the following formats is to be submitted: PDF (preferred), PS, Word. In case of acceptance, you will need to upload your source file in Word (for the moment, we do not accept TeX files, if you are interested to submit a paper in TeX, please, contact the editor). We will send you further instructions on uploading your camera-ready source files upon acceptance notification.

Deadline for the nearest issue (January-June 2009): January 15, 2009. Papers received after this date will be considered for the next issue (July-December 2009).

B. Format

Please, use IEEE format[1], see section "Template for all Transactions (except IEEE Transactions on Magnetics)". The editors keep the right to modify the format of the final version of the paper if necessary.

We do not have any specific page limit: we welcome both short and long papers, provided the quality and novelty of the paper adequately justifies the length.

Submissions in another format can be reviewed, but the use of the recommended format is encouraged.

In case of being written in Spanish, the paper should also contain the title, abstract, and keywords in English.

[1] www.ieee.org/web/publications/authors/transjnl/index.html