Structure of a sanskrit dictionary

Gérard Huet

INRIA-Rocquencourt

September 1st 2000

Introduction

We have been developing over the last 6 years an electronic sanskrit to french dictionary. This dictionary, which uses the tools from TEX[8], extended with the Devnag[12] package of Frans Velthuis, was released as a "Lexique sanscrit-français à l'usage de glossaire indianiste", made freely available on Internet as a Postscript document and advertised to scholars on the Indology forum on June 29th 1996. More recently a PDF[22] version was released as well, for the convenience of Acrobat[17] readers. This dictionay comprises currently 10500 entries in about 300 pages.

We then decided to make a fully indexed hypertext edition of the dictionary for the Web. This involved a complete rehaul of the source document, the writing of a parser which recognized the structure of the dictionary, and the elaboration of two processing chains, one for generating the printed form, the second for generating the Web documents with proper indexing. This new version was released on Internet in january 2000 as

`http://pauillac.inria.fr/~huet/SKT/DICO/`

The present report presents the underlying structure of the sanskrit dictionary, which got formalised in the course of this work.

This document comprises two sections. The first one gives a full example entry under the various formats, to give a concrete instance of the processing chains. The second one discusses the main points about the abstract syntax design. We conclude with a discussion of possible future research.

1 Example

We extract from the dictionary the entry "karman".

1.1 Source

First we show the source of the entry in our lexical database.

```
\word{karman}
\desf{k.r#1}{-man} \sm \sn
[\sing \nom \skt{karmaa}; \acc \skt{karmaanam}; \ins \skt{karmanaa};
 \incomp{karma}]
\sem{acte, action, &uvre; fait, exécution, opération; cérémonie, sacrifice}
\ou \sem{ouvrage; occupation, métier; destin}
\ou \sem{fonction individuelle, devoir de sa caste}
\ou \phil{accumulation de mérites et de fautes au cours des existences passées;
          destinée}
\ou \sem{Acte suprême, @uvre sainte}
```

```
\ou \sem{pratique (opp. théorie); \syn{prayoga}}
\ou \gram{le rôle [\skt{kaaraka}] objet (accusatif ou nominatif en mode passif)}
\ou \phil{[\skt{vai"se.sika}] la catégorie [\skt{padaartha}] des activités;
          la tradition en donne 5:
          \skt{utk.sepa.na} l'élévation,
          \skt{avak.sepa.na} l'abaissement,
          \skt{aaku~ncana} la contraction,
          \skt{prasaara.na} l'expansion
          et \skt{gamana} le déplacement}
\fin
```

This source format is basically a linearized tree structure, in a stream of 8 bits-characters (using the ISO-LATIN-1 encoding, permitting direct transcoding of the french letters with accents). Special identifiers starting with a backslash correspond to the labels of the nodes of this tree structure, while the corresponding substructure is indicated within curly brackets. For lists of items the special identifiers act as separators. Thus a semantic meaning item such as `\sem{A} \ou \sem{B} \ou \sem{C}` has three substructures A, B, and C. An alternate representation, in the data format XML[26], would be written as :

```
<semantics><sem> A </sem><sem> B </sem><sem> C </sem></semantics>
```

The TEX expert will recognize the similarity of our data base input format with the syntax of macro processing of this system. This is no surprise indeed, since the first level of reverse engineering of our dictionary consisted in systematizing the TEX macros, and then lifting them up as data structures. It was only natural to keep the concrete syntax, and so our lexical analysers inherit the notation `\macro{arg1}{arg2}` for calling function 'macro' with 2 arguments. But we allow also infixes, such as the variant construct `\ou` above, lists, and optional arguments.

This notation is in some sense arbitrary. The important point is that it is non-ambiguous, and neither too terse nor too verbose for human reading. In the current version this source is edited in a standard text editor (Emacs) ; it is likely that in some future the source will be an XML document, directly edited as a structure using an XML editing tool.

We refrain voluntarily from giving the full concrete syntax of the present version, although it is a perfectly honorable piece of programming using to the full the excellent concrete syntax facilities of the camlp4[18] preprocessor of Objective Caml[21]. But its sheer size prevents its detailed presentation except to specialists. I currently handle 3 lexical analysers and 3 context-free parsers defined as mutually recursive stream analysers, together with 6 transducers that translate exchange formats between Devnag, Tex, HTML, ISO-LAT-1, UTF-8, Math notation, and a canonical mapping of the romanized sanskrit alphabet into 49 integer codes. The semantics part allows many idiosyncraties having to do with trivia such as writing dates and places and names of plants and animals with proper reference to the scientific classification.

1.2 Abstract Syntax

Let us now visualise the tree structure, or "abstract syntax" of the above entry. We use the syntax for inductively defined types of Objective Caml[21], the programming language used to build all the dictionary tools. In this Ocaml notation, a list of three items A, B and C is printed as `[A; B; C]`. An item `Some(A)` corresponds to an optional value `A`, whereas the absence of an item in an optional position is noted `None`. Those are the two constructors of the optional datatype of the Ocaml standard library :

```
type option 'a = [ Some of 'a | None ];
```

Here is the abstract structure of the entry 'karman' :

```
(Entry (((Word "karman"), [], Some (Desf ("k.r#1", "-man"))),
  (Subst [(Meaning
    ((Roles ((Genderorth (Mas, None)), (Genderorth (Neu, None)))),
      Some
        [(Table (Some Singular, [(Caserole (Nom, ["karmaa"]))]));
        (Table (None, [(Caserole (Acc, ["karmaanam"]))]));
        (Table (None, [(Caserole (Ins, ["karmanaa"]))]));
        (Table (None, [(Incomp ["karma"])]))]),
     [(None, (Sem
        [(Affirm [[(Unit "acte")]; [(Unit "action")]; [(Unit "&uvre")]]);
         (Affirm
          [[(Unit "fait")]; [(Unit "exécution")]; [(Unit "opération")]]);
         (Affirm [[(Unit "cérémonie")]; [(Unit "sacrifice")]])]));
      (None, (Sem
        [(Affirm [[(Unit "ouvrage")]]);
         (Affirm [[(Unit "occupation")]; [(Unit "métier")]]);
         (Affirm [[(Unit "destin")]])]));
      (None, (Sem [(Affirm
          [[(Unit "fonction"); (Unit "individuelle")];
           [(Unit "devoir"); (Unit "de"); (Unit "sa"); (Unit "caste")]])]));
      (None, (Phil [(Affirm
          [[(Unit "accumulation"); (Unit "de"); (Unit "mérites");
            ... more of the same ... ]])]))])])))
```

The precise description of the various operators available in our abstract structure datatype will be given in section 2.

1.3 DEVNAG input

The TEX version of our preprocessor is basically a specialised printer of the above structure, creating the intermediate representation which follows :

```
$karman$~{\sl karman}
[{\nobreak}{\sl k{\d r}\(_{1}\)}{\sl -man}\/\nobreak] m. n.
[{\nobreak}sg. nom. {\sl karm\=a}; acc. {\sl karm\=anam}; i. {\sl karman\=a};
en comp. {\sl karma}\/\nobreak]
acte, action, {\oe}uvre;
fait, exécution, opération;
cérémonie, sacrifice
 \(\mid\:\)ouvrage;
occupation, métier;
destin
 \(\mid\:\)fonction individuelle, devoir de sa caste
 \(\mid\:\)phil. accumulation de mérites et de fautes au cours
des existences passées;
destinée
```

```
 \(\mid\:\)Acte suprême, {\OE}uvre sainte
 \(\mid\:\)pratique (opp. théorie);
syn. {\sl prayoga}\/
 \(\mid\:\)gram. le rôle [{\nobreak}{\sl k\=araka}\/\nobreak] objet
(accusatif ou nominatif en mode passif)
 \(\mid\:\)phil. [{\nobreak}{\sl vai\'se{\d s}ika}\/\nobreak] la catégorie
[{\nobreak}{\sl pad\=artha}\/\nobreak] des activités;
la tradition en donne 5: {\sl utk{\d s}epa{\d n}a}\/ l'élévation,
{\sl avak{\d s}epa{\d n}a}\/ l'abaissement, {\sl \=aku\~ncana}\/ la contraction,
{\sl pras\=ara{\d n}a}\/ l'expansion et {\sl gamana}\/ le déplacement.
```

The next step of processing is effected with the DEVNAG tool originally developped by Franz Velthuis at University of Groningen. We currently use version 1.6 of this processor, maintained by Anshuman Pandey[12]. This tool is basically a ligature analyser, which produces *devanāgarī* representations in a Metafont alphabet of ligature glyphs, fit for further processing by the TEX composition system. In our example, the string `$karman$` expands into a cryptic notation `{\dn km\0\qq{n}}`, the rest of the document being preserved. This produces the TEX input document.

1.4 Final print

Using the TEX processor LATEX 2_ε[9], we then obtain a low-level printable document in the device independent format DVI, which may be itself translated into the Postscript[23] format, fit for direct printing by high-resolution printers.

Alternatively, the DVI document may be read on computer screens with tools such as the XDVI Previewer of the X Window System. The Postscript document itself may be previewed in a variety of platforms with tools such as Ghostscript[20].

Here is the output ultimately produced by the above entry (in one column format, the actual dictionary uses a two-column format) :

> कर्मन् *karman* [*kṛ*$_1$*-man*] m. n. [sg. nom. *karmā* ; acc. *karmānam* ; i. *karmanā* ; en comp. *karma*] acte, action, œuvre ; fait, exécution, opération ; cérémonie, sacrifice | ouvrage ; occupation, métier ; destin | fonction individuelle, devoir de sa caste | phil. accumulation de mérites et de fautes au cours des existences passées ; destinée | Acte suprême, Œuvre sainte | pratique (opp. théorie) ; syn. *prayoga* | gram. le rôle [*kāraka*] objet (accusatif ou nominatif en mode passif) | phil. [*vaiśeṣika*] la catégorie [*padārtha*] des activités ; la tradition en donne 5 : *utkṣepaṇa* l'élévation, *avakṣepaṇa* l'abaissement, *ākuñcana* la contraction, *prasāraṇa* l'expansion et *gamana* le déplacement.

Notice how we obtain typographical perfection through TEX, which gives access to fine controls such as italics correction. In order to capture acceptable hyphenation, notably in the sanskrit romanised portions, we allow fine tuning by recognizing hyphenation hints in our input format. Thus `\skt{bodhi+sattva}` indicates that this word may be hyphenated as *bodhi-sattva*.

1.5 Portable document format

Postscript[23], a registered product of the Adobe company, is a photocomposition programming language, interpreted by high-end printers. It is progressively replaced by the more portable PDF[22] (portable document format). The commercially available Adobe tool Acrobat Distiller may be used to translate from Postscript to PDF. We released on Internet our dictionary as a PDF file, which may be viewed (and printed) using the freely available Acrobat Reader[17].

1.6 HTML document

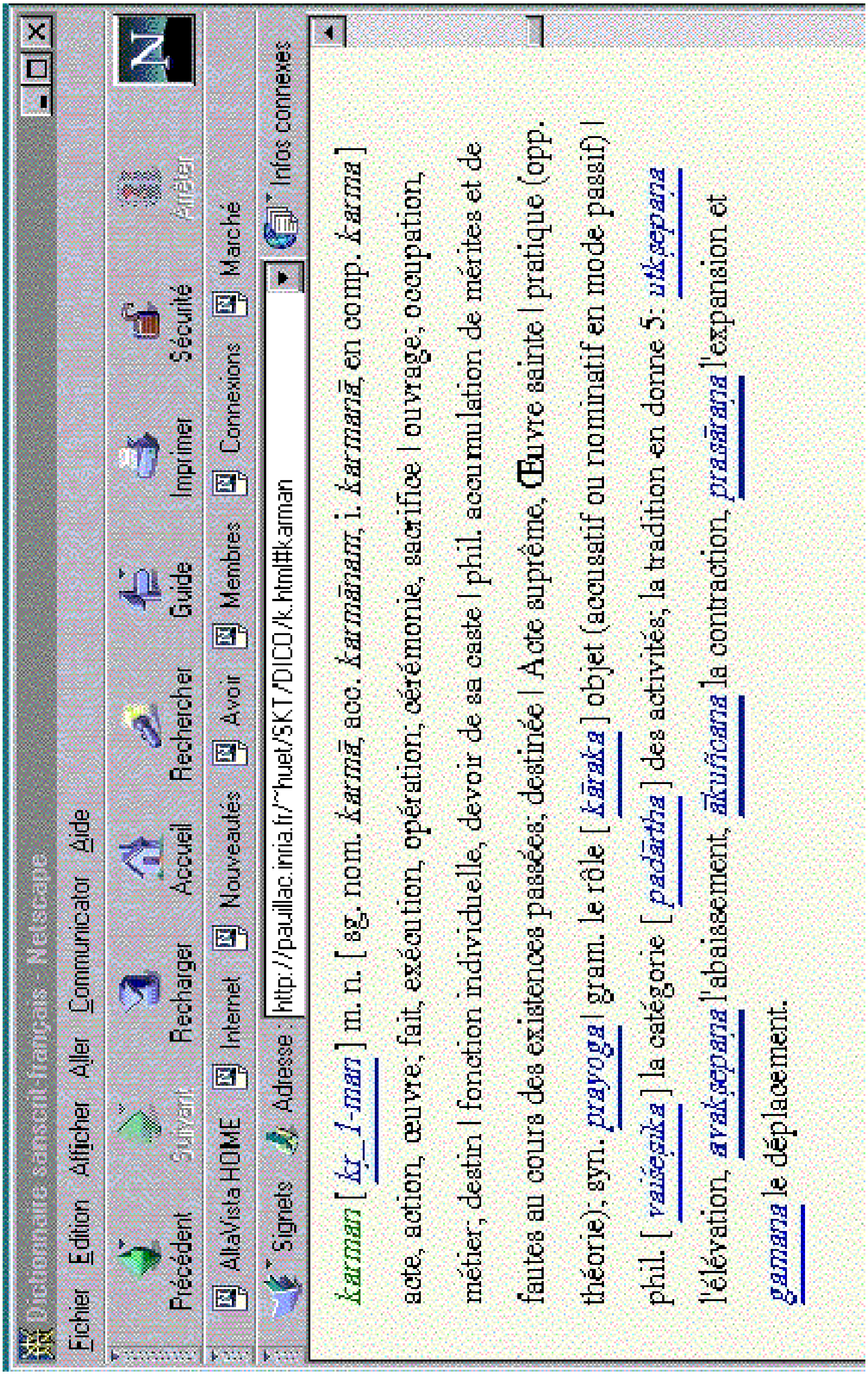
Another chain of processing prints our abstract structure format into an HTML representation. HTML[13] is the language used for the representation of Web pages. We use HTML V4.0, using so-called cascading style sheets (CSS) for the layout instructions. Letters with diacritics used for the romanised sanskrit words are represented in the Unicode[25] alphabet, under the RTF-8 encoding. We use the publicly available Indic Roman[6] font designed by Chris Fynn for their rendering.

Here is the HTML form of the above entry :

```
<P CLASS="dico";><A CLASS="defg"; NAME="karman"><I>karman</I></A>
[ <A CLASS="defb"; HREF="#k.r*1"><I>k&#7771;_1</I></A><A CLASS="defb";
HREF="#-man"><I>-man</I></A> ] m. n. [ sg. nom. <I>karm&#257;</I>;
acc. <I>karm&#257;nam</I>; i. <I>karman&#257;</I>; en comp. <I>karma</I> ]
acte, action, &#339;uvre; fait, ex&#233;cution, op&#233;ration;
c&#233;r&#233;monie, sacrifice
 | ouvrage; occupation, m&#233;tier; destin
 | fonction individuelle, devoir de sa caste
 | phil. accumulation de m&#233;rites et de fautes au cours des existences
pass&#233;es; destin&#233;e
 | Acte supr&#234;me, &#338;uvre sainte
 | pratique (opp. th&#233;orie);
syn. <A CLASS="defb"; HREF="#prayoga"><I>prayoga</I></A>
 | gram. le r&#244;le
[ <A CLASS="defb"; HREF="#k=araka"><I>k&#257;raka</I></A> ]
objet (accusatif ou nominatif en mode passif)
 | phil. [ <A CLASS="defb"; HREF="#vai^se.sika">
<I>vai&#347;e&#7779;ika</I></A> ] la cat&#233;gorie
[ <A CLASS="defb"; HREF="#pad=artha"><I>pad&#257;rtha</I></A> ]
des activit&#233;s; la tradition en donne 5:
<A CLASS="defb"; HREF="#utk.sepa.na"><I>utk&#7779;epa&#7751;a</I></A>
l'&#233;l&#233;vation,
<A CLASS="defb"; HREF="#avak.sepa.na"><I>avak&#7779;epa&#7751;a</I></A>
l'abaissement,
<A CLASS="defb"; HREF="#=aku~ncana"><I>&#257;ku&#241;cana</I></A> la contraction,
<A CLASS="defb"; HREF="#pras=ara.na"><I>pras&#257;ra&#7751;a</I></A> l'expansion
et <A CLASS="defb"; HREF="#gamana"><I>gamana</I></A> le d&#233;placement.
```

1.7 Web rendering

Using the Netscape Navigator 4.5 Browser, on a platform where the Indic Roman font has been installed, this entry is rendered as shown in the next page.


Dictionnaire sanskrit français - Netscape
Fichier Edition Afficher Aller Communicator Aide
Précédent Suivant Recharger Accueil Rechercher Guide Imprimer Sécurité Arrêter
AltaVista HOME Internet Nouveautés Avoir Membres Connexions Marché
Signets Adresse : http://pauillac.inria.fr/~huet/SKT/DICO/k.html#karman
Infos connexes
karman [kṛ_1-man] m. n. [sg. nom. karmā acc. karmāṇam, i. karmaṇā en comp. karma]
acte, action, œuvre; fait, exécution, opération; cérémonie, sacrifice | ouvrage; occupation,
métier; destin | fonction individuelle, devoir de sa caste | phil. accumulation de mérites et de
fautes au cours des existences passées; destinée | Acte suprême, Œuvre sainte | pratique (opp.
théorie); syn. prayoga | gram. le rôle [kāraka] objet (accusatif ou nominatif en mode passif) |
phil. [vaiśeṣika] la catégorie [padārtha] des activités; la tradition en donne 5: utkṣepaṇa
l'élévation, avakṣepaṇa l'abaissement, ākuñcana la contraction, prasāraṇa l'expansion et
gamana le déplacement.

Note that the blue underlined words are links which may be followed by clicking on them, allowing convenient navigation in the dictionary, which is thus usable as a small hypertext encyclopedia for indian culture. Users do not have to be versed in subtleties of sanskrit phonetics and transliteration. For instance, one may click on *liṅga* without worrying about the transliteration convention `"n` for the guttural nasal *ṅ*.

1.8 Further Web tools

Translating the dictionary into one single Web page is the simplest way to obtain a fully indexed hypertext document where one may navigate. However the size of this page (about 3.2 Megabytes) makes it inconvenient to use interactively at a distance, and so it must be downloaded in your computer offline first. Even then, it puts a heavy load on your workstation. For instance, Netscape grows to a 25 Mb process when loading it.

In order to make a Web version which is browsable interactively without lengthy downloads, we built a more modular version, where consultation at a distance is possible. The current version at the time of writing breaks the document into 50 pages, one for each initial letter of the entries. This limits each page to a maximum of 350Kb, with tolerable load times.

We then proceeded in designing a little search engine, allowing consultation of an index through an Ocaml script accessible through the Common Gateway Interface[19] convention for remote procedure calls. For instance, one may search for the entry "yoga" by invoking the URL `http://pauillac.inria.fr/bin/dicindex?yoga`. This indexing facility was then encapsulated as an HTML `<FORM>` construct, and the subsequent index facility was released on February 1st, 2000, as `http://pauillac.inria.fr/~huet/SKT/DICO/index.html`.

This search engine gives an index to the closest entry matching a given keyword, entered as a romanised string. By closest we mean the first entry in lexical order such that the keyword is an initial substring of it, or if there is none the first entry in lexical order following the given keyword. The search algorithm looks up a specific data structure of indexing trie, implemented as a fully applicative zipper[7]. This indexing trie is built and stored on disk when processing the dictionary for HTML output. Various special options exist, in order to check for undefined references, or double definitions.

Finally, the lexical ordering of the entry items is automatically checked, using a code normaliser which substitutes its *savarṇa anunāsika* to non-original *anusvāra*, and similarly replaces a *visarga* by its sibilant substitutive. All these computations are performed on the representation of sanskrit words by numerical codes. The next tool envisioned is a sandhi processor.

1.9 A modular system

Our TEX and HTML processors share actually a lot of the work.

Here is the source code of the program 'dictex' that processes the data base for TEX output :

```
open Print_tex;
open Main;
module Dictex = Main(Print_tex);
```

Similarly, here is the source code of the program 'dichtml' that builds the HTML output :

```
open Print_html;
open Main;
module Dichtml = Main(Print_html);
```

That is, these two programs are actually two instances of one parameterized module `Main`. The two parameters, respectively `Print_tex` and `Print_html`, give the details of the low-level rendering in the respective formats. All the parsing, construction of the abstract structure, and all the recursion structure of the rendering are shared.

This modularity is extremely important, because it allows us to maintain only one source, even though the TEX and HTML versions are completely different. The basic structure of the document is the same, only low-level details on its actual printing differ. For instance, the TEX printout insures fine typesetting, and *devanāgarī* rendition of the main entries, whereas the HTML printout inserts the navigation anchors and links which are necessary for Web browsing, but which are irrelevant for the book edition.

The module Grind is the core of our processor. It is basically a parser of our sanskrit lexicographic database, building on the fly the abstract syntax tree representation. We shall not describe here the parsing algorithm used (we just mention that it uses the parsers and printers formalism developed in the camlp4 preprocessor[18]). We rather concentrate on the abstract syntax representation, since it reveals the information contents of our dictionary.

2 Abstract Syntax

2.1 Sanskrit quotations

The following declarations give the structure of sanskrit words, and text as *pada* (list of sanskrit words) and *ṣloka*. We allow single and double *danda* as punctuation of the *pada* list which makes a multiple *ṣloka*. We use Velthuis transliteration scheme, as used in the DEVNAG software program. Thus `"braahma.na"` represents the roman transliteration with diacritics *brāhmaṇa*. In this module, we use `skt` as the type of strings recognised by the `skt` entry of our sanskrit lexical analyser. It is exported from the sanskrit parser to outside modules as an abstract type, insuring proper lexical verification of values of type `skt`.

```
type skt = string;
type pada = list skt;

type danda = [ Singledanda | Doubledanda ]
and sanscrit = [ Pada of pada | Sloka of list (pada * danda) ];
```

Next is a little notation for verb prefixes with precomputed internal sandhi.

```
type prefix = [ Sandhi of (skt * skt) | Nosandhi of skt ];
```

As an example, I annotate the entry for root *sthā* by an index of all its prefixed entries in the dictionary as :

```
\px{adhi(.s), anu(.s), ava, aa, ut(), upa, pra, prati(.s), sam}
```

thus giving references to verbs *adhiṣṭhā*, *anuṣṭhā*, *avasthā*, *āsthā*, *utthā*, etc. Such notation could be partly synthesised by a sandhi preprocessor (that is, except for the elision of the initial ‘s’ with prefix *ut*).

Here as generally later, I shall not give explicitly the concrete syntax which is currently used, since in many ways it is arbitrary, is subject to change, and may easily be adapted to other input format conventions. We may imagine for instance writing a transducer which generates entries in this structure from some other preexisting dictionary, either in electronic form or digitalised from a book. What matters is the (typed) tree structure of the abstract syntax.

2.2 French sentences

The following algebra describes a rudimentary structure for french sentences. French identifiers are represented in ISO-LATIN-1 encoding, except for the symbol `&` (resp. `@`) used for the ligature œ (resp. Œ), and for the symbol `+` used to give hyphenation hints. Thus 'écœuré' will be represented as `"éc&uré"`, a value of type `french`. Such an encoding may be input directly on a french AZERTY keyboard, and rendered adequately in a text editor such as EMACS in the proper "standard-display-european" mode. Again `french` and `math` are abstract types of the french parser module, insuring proper lexical correctness of their string representation.

N.B. In the following Ocaml printouts, strings `(* ... *)` are comments.

```
type french = string
and  math   = string;

type year = [Exact of string | Approx of string | Conject of string]
and date = (year * year);

type word =
   [ Unit of french              (* french word *)
   | Number of string            (* number written with numerals *)
   | Composite of (word * string * word)      (* composed french word *)
   | Skt of skt                  (* sanskrit reference *)
   | Sktd of skt                 (* sanskrit defining occurrence *)
   | Sktn of skt                 (* sanskrit without reference *)
   | Sktcit of sanscrit          (* sanskrit citation *)
   | Np of skt                   (* sanskrit proper name *)
   | Npd of skt         (* sanskrit proper name defining occurrence *)
   | Npn of skt         (* sanskrit proper name without reference *)
   | Gov of subphrase   (* governance e.g. <acc. g. \skt{prati}-acc.> *)
   | Ref of skt                  (* cf. *)
   | Syn of list skt             (* synonyms *)
   | Opp of list skt             (* antonyms *)
   | Math of math                (* maths *)
   | Expo of (string * string)   (* 2^e *)
   | Date of date                (* dates *)
   | Bot of french               (* botanics *)
   | Zoo of french               (* zoology *)
   | Ital of french              (* italics *)
   | Parenthese of utterance     (* (...) *)
   | Quotation of utterance      (* "..." *)
   | Quote of word               (* 'word' *)
   | Bracket of word             (* [word] *)
   | Special of string           (* special notation for |, om, ... *)
   ]
and subphrase = list word        (* separated by spaces *)
and phrase = list subphrase      (* separated by , *)
and utterance =
   [ Affirm of phrase
```

```
  | Excla of phrase              (* ! *)
  | Inter of phrase              (* ? *)
  | Opening of (phrase * utterance) (* : *)
  ]
and sentence = list utterance; (* separated by ; *)
```

Note that we do not attempt any syntactic analysis of french sentences, our analysis concerns solely morphology, punctuation and markup.

We are specially careful about indexing in the french sentence any reference to a sanskrit noun. Actually we distinguish between defining and non-defining occurrences, and between common nouns and proper names, which start with an upper case letter. The defining occurrences will generate an anchor in the HTML printout, whereas the non-defining occurrences will generate a reference to the corresponding anchor. A typical example of the actual concrete notation in my source reads : `np. du mage [\skt{.r.si}] \npd{Agasti}` which will generate an anchor for references to the proper name Agasti, and a link to the entry *ṛṣi*.

Note that our structure of french sentences is very flat and simple-minded compared to say a parse tree - it is basically a hierarchy of list of tokens. On the other hand, it is a quite precise markup of the sentence compared to the recommendations of the Text Encoding Initiative[24]. Elements of the type sentence are going to represent semantical meanings of our dictionary entries.

From the point of view of the dictionary structure, which we are now going to describe, a `phrase` is just a list of meanings forming a semantical sense. Each meaning, as a `subphrase`, is a list of quasi-synonyms whose union gives an approximation, in terms of concepts enunciated in french, of the sanskrit meaning. This concerns the lexicon entries, where semantical senses are translative definitions which have a substitutive character. For encyclopedic entries, in contrast, semantical senses are descriptions and glosses in french explaining the sanskrit notion in terms of both french and sanskrit notions, the latter corresponding to other entries and marked as such. These glosses may take the form of full french sentences, which are sequences of utterances, or phrases given with a mode. It is not always easy to distinguish these two ways of paraphrasing meanings in french, since substitutive definitions may be accompanied by commentaries, and more work is needed to propose a more precise structure to semantics. We shall come back to this question in the conclusion.

2.3 Grammatical attributes

We now describe the overall dictionary structure. We first give a few simple notions from grammar : gender, number, case, and the structure of verbal constructs.

```
type gender = [ Mas | Neu | Fem | Any ]
and genders = list gender;

type number = [ Singular | Dual | Plural ] ;

type case = [ Nom | Acc | Ins | Dat | Abl | Gen | Loc | Voc ];

(* the verb system *)
type voice = [ Active | Reflexive ]
and mode = [ Indicative | Imperative | Causative | Intensive | Desiderative ]
and tense = [ Present of mode | Perfect | Imperfect | Aorist | Future ]
and nominal = [ Pp | Ppr of voice | Ppft | Ger | Infi | Peri ]
and verbal =
```

```
  [ Conjug of (tense * voice)
  | Passive
  | Optative of voice
  | Nominal of nominal
  | Absolutive | Conditional | Precative
  | Derived of (verbal * verbal)
  ];
```

We add to the traditional genders masculine, feminine and neuter, a fourth category "Any" which is attribute to the words whose gender is inherited from the context (pronouns of first and second person, the reflexive pronoun *ātman*, and numerals greater than four).

Conjugate forms have a tense and a voice, the present form admits five modes, there are five sorts of nominals (verbal adjectives), we have optative at both voices, and finally passive, absolutive, conditional and precative derived forms. What we call reflexive is usually called the middle voice, a rather poor terminology. Note that passive is not listed as a third voice, a possible option. More complicated derived compound forms may be formed. Thus we shall use `Derived(Absolutive,Conjug(Present(Causative),Active))` for the absolutive of the active causative, and `Derived(Nominal(Ppr(Active)),Conjug(Present(Intensive),Active))` for the active present participle of the active intensive mode of the present.

2.4 Etymology

Next we present the structure of etymological origin of a word. Thus `Depf("pra","k.r#1")` gives the verb `prak.r` as obtained form root `k.r#1` by preverb `pra`, whereas `Depf("pra","dak.si.na")` gives the derivation of word `pradak.si.na` from `dak.si.na`. Similarly, the word `anukiirtana` is annotated as having etymology `Desf("anukiirt","-ana")`, meaning that it is derived from verb `anukiirt` by suffixing it with `-ana`. Now `anukiirt` itself has etymology `Depf("anu#1","kiirt")`, `kiirt` is marked as a nominal verb derived from substantive `kiirti`, which derives itself from root `k.r#2`. All these etymology links can be followed by hypertext navigation in the Web version. Note the sharp sign notation for homonyms.

We chose to merge in one structure etymologies of verbal forms and of nouns, in order to avoid duplications, but in some sense this is an arbitrary design decision, subject to review.

```
type etym =
  [ Onom                  (* onomatopea *)
  | Rel of skt            (* related to *)
  | De of skt             (* derived from *)
  | Compd of skt          (* compound *)
  | Plde of skt           (* plural of *)
  | Dude of skt           (* dual of *)
  | Contrde of skt        (* contraction of *)
  | Vnde of skt           (* nominal verb *)
  | Verbal of (verbal * skt)       (* verbal form *)
  | Gende of (gender * skt)        (* gender form *)
  | Decl of (case * number * skt)  (* declined form *)
  | Thde of skt           (* stem of *)
  | Rede of skt           (* reduplicated form of *)
  | Red                   (* reduplication (subword) *)
  | Desfun of skt         (* from suffix (subword) *)
```

```
  | Vrde of skt            (* vriddhi form of *)
  | Supde of skt           (* superlative of *)
  | Compde of skt          (* comparative of *)
  | Supplde of skt         (* suppletive of *)
  | Correlde of skt        (* correlative of *)
  | Iicfor of skt          (* iic. for *)
  | Ifcfor of skt          (* ifc. for *)
  | Enclfor of list skt    (* enclitic for *)
  | Dede of (skt * skt)    (* composite from *)
  | Depf of (skt * skt)    (* from prefix and word *)
  | Desf of (skt * skt)    (* from word and suffix *)
  | Devrde of (skt * skt) (* compositive from vriddhi *)
  | Devrsf of (skt * skt) (* vriddhi of word and suffix *)
  | Dedede of (skt * skt * skt)  (* ternary from *)
  | Depfsf of (skt * skt * skt)  (* from prefix word and suffix *)
  | Prakritde of skt       (* prakrit form of *)
  | Palide of skt          (* pali form of *)
  ];
```

Next we give the various kinds of entries in the dictionary. Main entries are the roots, the words and the suffixes. Subentries Subword and Subsubword give two levels of compound words (or derivatives such as qualities in *tva* and *tā*). Auxiliary entries list idiomatic expressions, which appear as indexable entries, other compound expressions, and occasionally we give an example citation. The current version has the following share of the various kinds : 6562 words, 2077 subwords, 66 subsubwords, 521 roots, 59 suffixes, 1178 expressions, 27 idioms and 40 citations.

```
type entete =
  [ Word of skt
  | Root of skt
  | Suffix of skt
  | Subword of skt
  | Compound of skt
  | Subsubword of skt
  | Declined of skt
  | Idiom of skt
  | Expression of sanscrit
  | Citation of sanscrit
  ];
```

Each entry may be annotated with a list of variants. This ends the syntactic part of our entries :

```
type variants = list skt;

type syntax = (entete * variants * option etym);
```

2.5 Conjugation lists

Verbs are listed with a list of conjugation structures. The following is an abstraction of what we needed for our corpus, but it is not yet aligned on the `verbal` structure above. We must think of this structure as the interim form of querying a database with the complete conjugation tables of the entries. A complete systematic analysis of the sanskrit verbal structure is not yet available at the time of writing it seems, and we can only guess from the forms given in the classical "Roots of Sanskrit" of Whitney[16] that the full story is vastly more complicated than the usual classification of verbs in 10 groups that is told in the standard grammars of sanskrit.

```
type conjug =
  [ Pr of skt           (* present *)
  | Pm of skt           (* present middle *)
  | Nopres              (* verb with no present form *)
  | Pf of skt           (* perfect *)
  | Pfm of skt          (* perfect middle*)
  | Impft of skt        (* imperfect *)
  | Impftm of skt       (* imperfect middle *)
  | Ps of skt           (* passive *)
  | Fu of skt           (* future *)
  | Fum of skt          (* future middle *)
  | Fp of skt           (* periphrastic future *)
  | Ca of skt           (* causative *)
  | Cam of skt          (* causative middle *)
  | Ppl of list skt     (* past participle list *)
  | Ppra of skt         (* present participle active *)
  | Pprm of skt         (* present participle middle *)
  | Ppri of skt         (* present participle intensive *)
  | Ppf of skt          (* perfect participle *)
  | Gerl of list skt    (* gerundive list *)
  | Imp of skt          (* imperative *)
  | Impm of skt         (* imperative middle *)
  | Opt of skt          (* optative *)
  | Optm of skt         (* optative middle *)
  | Inf of skt          (* infinitive *)
  | Absl of list skt    (* absolutive list *)
  | Des of skt          (* desiderative *)
  | Desm of skt         (* desiderative middle *)
  | Int of skt          (* intensive *)
  | Aor of skt          (* aorist *)
  | Aorp of skt         (* aorist passive *) (* hapax \root{j~naa#1} *)
  | Aorm of skt         (* aorist middle *)
  | Cond of skt         (* conditional *)
  | Pre of skt          (* precative *)
  | Px of list prefix   (* used with prefixes list *)
  | Alter of string     (* alternative verb form *)
  ];
```

Verb tenses are given with just the 3rd person singular form (except imperatives, for which the default is 2nd person singular). Past participles, gerundives and absolutives are listed under possibly several forms.

The `Px` annotation permits to reference all verbs derived by prefixes from a given stem, alleviating the rather tedious listing as derived forms in the root entries, far away from their lexicographical rank, as sometimes done in "logical" lexicons such as Bergaigne's[1].

2.6 Roles

Now we are entering the area of the grammatical classification of the entry. We define a notion of `role` for this purpose. Here again a certain pragmatism prevails. We do not go as far as Monier-Williams who lumped substantives and adjectives as a type of 'mf(ā)n' or 'mf(ī)n', and rather prefer to stage the different genders as different roles deserving different meanings. We keep the notion of agent, though, as a sharing of meanings of the masculine (or feminine) form and the adjective in the usual assimilation of a unary predicate with the choice operator which abstracts a representative element from a non-empty set.

Within a given role we allow refinements which are governed optionally by a special orthography. This is how for instance we indicate collective meanings of plural forms, such as people names.

```
(* kinds of pronouns *)
type pers = [ First | Second | Third | Refl ]
and pn_kind = [ Pers of pers | Demons | Relat | Other ];

(* kinds of adjectives *)
type adj_kind = [ Compar | Super | Ordinal | Poss | Usual | Allgenders ];

type ind_kind = [ Adv | Part | Prep | Conj | Nota | Flex ];

type mode_kind =
  [ Interr                    (* interrogative *)
  | Exclam                    (* exclamative *)
  | Adr                       (* address *)
  | Interj                    (* interjection *)
  | Inj                       (* injunction *)
  ];

type case_role = (list case * list skt)
and gender_decl = (genders * option skt);

type gram_role =
  [ Gender of gender_decl                 (* substantive with opt. orth *)
  | Adj of adj_kind                       (* adjective *)
  | Agt                                   (* agent *)
  | Pn of (pn_kind * option genders)      (* pronoun *)
  | Num of option genders                  (* numeral *)
  | Ind of ind_kind                       (* undeclinable *)
  | Pref                                  (* prefix *)
  | Gendered of (gender_decl * case_role) (* gender & cases *)
  | Caserole of case_role                 (* cases & opt orth as role *)
  | Caseroleind of (case_role * ind_kind) (* id. as adv role *)
  ];

type position =
  [ Encl                      (* enclitic *)
  | Enclafter of list skt     (* enclitic after eg ena *)
```

```
  | Enclafternum                 (* enclitic after num. *)
  | Ifpx of list prefix          (* after prefix eg a~nc#2 *)
  | Iic                          (* starts a compound *)
  | Ifc                          (* ends a compound *)
  | Iip                          (* starts a phrase *)
  ];

type role =
  [ Unpositioned of gram_role
  | Positioned of (gram_role * position)
  | Position of position
  ];

type decl =
  [ Caseorth of case_role      (* cases & opt orth as decl *)
  | Varencl of case_role       (* enclitic variant *)
  | Except of case_role        (* unallowed decl *)
  | Only of case_role          (* exclusive decl *)
  | Ploc                       (* plural at other cases *)
  ];

(* irregular declensions tables *)

type decls = list decl (* sep COMMA *)
and table =
   [ Table of (option gender * option number * decls)  (* declensions table *)
   | Gendecl of (genders * skt)  (* substantive with orth *)
   | Oblique of skt             (* oblique stem *)
   | Weak of skt                (* weak stem *)
   | Incomp of list skt         (* iic variant *)
   | Otheras of skt             (* other cases as *)
   | Declas of skt              (* declined as *)
   ]
and tables = list table; (* sep SEMI *)

type orth =
  [ Orth of skt          (* other orthography for case *)
  | For of skt           (* stands for *)
  | Var of skt           (* alternate orthography *)
  | Du of skt            (* dual form *)
  | Pl of skt            (* plural form *)
  | Numb of number       (* alternative number *)
  | Mode of mode_kind    (* mode *)
  | Suppl of skt         (* suppletive *)
  | Correl of skt        (* correlative *)
  | Futperi of skt       (* periphrastic future *) (* eg {dra.s.t.r} *)
  | Declension of tables (* Declension directive *)
```

```
];
```

We are conscious here that our notion of role is a compromise between grammatical notions from sanskrit and from french (and even includes some pragmatics) ; for instance, we shall typically distinguish the roles as substantive and as adjective of a sanskrit word whenever french will distinguish the two notions by different translations. The tension between the two uses of the notion is typical for an absolutive. From the sanskrit grammatical point of view, such words are undeclinables. From the french syntax point of view, they are used as adverbs. This is also why we define a coercion from cases to roles with the constructor Caserole, since e.g. locatives are usually translated as adverbs. It is unclear to what extent we should pursue this attempt at local translation within the lexicon of grammatical roles. For instance, an instrumental may sometimes be translated as an adverb, but often it stands for the logical subject in a passive construction, and this discrimination is the responsability of the syntax analyser, not of the lexicon.

We mention irregular declensions in a table which encodes grammatical information. This part of the structure of an entry is optional, since it describes mostly exceptions.

Our declension tables are a bit ad-hoc, in the same way as our conjugation structures above are ad-hoc. They reflect all the information that the user wants to see, in general a rather minimal part. Progressively, complete declension and conjugation tables will be recorded by mechanical computation in our data-base, and the table structure will be a minimal encoding of parameters and exceptions.

2.7 Meanings

Let us turn now to the structure of the semantics. There are two kinds of semantic entries in our dictionary. The first kind, corresponding to constructor `Sem` below, is the plain definition in a substitutive tradition of lexicography. The other entries are encyclopedic, they classify the knowledge abour indian culture as pertaining to vedic culture, or philosophy or myth, or a buddhist or jain tradition, or arts and sciences of astronomy/logy, grammatical and phonetical terminology, and modern words.

```
type sem =
  [ Sem of sentence  (* Semantics *)
  | Ved of sentence  (* Vedic *)
  | Phil of sentence (* Philosophy *)
  | Myth of sentence (* Mythology *)
  | Bd of sentence   (* Bouddhism *)
  | Jn of sentence   (* Jainism *)
  | Astr of sentence (* Astronomy *)
  | Gram of sentence (* Grammar *)
  | Phon of sentence (* Phonetics *)
  | Mod of sentence  (* Modern *)
  | Cf of skt        (* Reference *)
  | Unused           (* verb used only with px *)
  ];
```

Now the meanings of the various roles of a substantive, each associated with a list of semantic entries optionally qualified with a specific orthography, is described as :

```
type semantics = list (list orth * sem);
```

```
type meaning =
  [ Meaning of (role * semantics)
  | Refgen of (gender * skt)   (* s. v. *)
  ]
and meanings = list meaning;
```

Let us give a concrete exemple. In the following entry, we have two meanings, one associated with the role "substantive masculine", the second with the role "substantive neuter". The first one has two semantics, the second of which having a special orthography indicating a different accent.

> अक्ष *ákṣa* m. essieu de voiture, objet tournant | *akṣá* dé à jouer (cylindre d'ivoire de section carrée, qu'on fait rouler pour choisir un nombre entre 1 et 4) — n. œil || lat. *axis* ; ang. axle ; fr. axe.

Note that the semantics "dé" is annotated by a commentary, since indian dice are cylinders and not cubes. This is typical of a cultural difference which makes translation not quite substitutive.

A similar structure is defined for the meanings of verbs, with conjugation lists instead of roles.

```
type vbsemantics = list ((option skt) * sem) (* opt. orthographic variant *)
and vbmeaning = (list conjug * vbsemantics)
and vbmeanings = list vbmeaning;
```

2.8 Entry structure

We end each entry with an optional reference to cognate etymologies in other languages.

```
type cognate =
    [ Fr of list french
    | Gm of list french
    | Eng of string
    | Greek of math
    | Lat of list skt
    | Hindi of list skt
    | Prakrit of list skt
    | Misc of (french * string)
    ];

type cogs = list cognate;
```

We are now in the position of explaining the two types of entries in our dictionary : the *bona fide* entries and the cross-references to those. A genuine `Entry` possesses a `usage`, as a substantive, a verb, or an idiomatic expression (whose role is implicit). A cross-reference does not carry a meanings component.

```
type usage =
   [ Subst of meanings                (* Substantives            *)
   | Verb of (string * vbmeanings)   (* Verbs given with group   *)
   | Idiomatic of semantics           (* Idiomatic expressions    *)
   ];

type crossref =
   [ Refer of skt                     (* simple cross-reference   *)
```

```
  | Initial of skt                (* used iic. for entry      *)
  | Terminal of skt               (* used ifc. for entry      *)
  | Enclitic of list skt          (* enclitic form of entries *)
  ];

type entry =
  [ Entry of (syntax * usage * option cogs * bool)
  | Crossref of (syntax * crossref)
  ];
```

In Entry, the last bool argument is True if a final dot is to be printed. This is due to a very nagging linguistic problem, arising from the ambiguity between full stops and periods used for abbreviations. We solve this problem here by human specification, as opposed to a systematic treatment which is at present lacking, although heuristics have been experimented with.

3 Discussion and Future developments

3.1 Discussion

The first point we want to emphasise is that the structure of dictionary presented here has not been designed as a theoretical concept around which the dictionary was constructed. On the contrary, the dictionary evolved for many years before we set to cast it along a precise syntax. We were just careful from the start to do things in as much a homogeneous manner as possible. This was inforced by a discipline of systematic use of TEX macros, with very little non-generic syntax. The whole structure was thus reverse-engineered from the total database of more than 10000 entries. Needless to say, a lot of smoothing out and simplification had to be applied before succeeding in parsing the whole document with a not-too-complex syntax. At present, the total number of production rules of the various grammars involved is 321 (14 for the sanskrit structure, 54 for the french, and 253 for the dictionary itself). Most production rules are indeed a rather direct mapping from concrete syntax to one abstract operator construction, with a one-to-one mapping between entry points of the grammar and types.

The issue of completeness is a thorny one. Sanskrit documents span a 4000 years period, from the Vedic hymns to contemporary sanskrit revival magazines, and our dictionary does not claim completeness with respect to any well-defined corpus. We are however confident that it is fairly complete with respect to say the 5000 most frequent stems occurring in classical sanskrit. The goal of the work being to serve as an indological glossary, there is a natural bias towards philosophical, ritual, mythological, iconographical and linguistic themes. There are relatively few compound words (2130 to be compared to 7150 main entries, comprising 521 roots). Of course in some sense there is no complete sanskrit dictionary, since the language permits the generation of an unbounded number of lexical items through arbitrary depth of composition. We may compare roughly our 10000 entries to the 50000 entries of the Stchoupak, Nitti and Renou (the standard sanskrit to french dictionary), and to the 160000 entries of the Monier-Williams (the standard sanskrit to english dictionary, itself an abridgement of the more complete St Petersburg dictionary). At the high end of the spectrum the Pune dictionary is under construction on the basis of 11 million paper reference slips.

Specialised vocabulary used by buddhist and jain communities is mostly absent. Indeed we tried hard to present a unified 'orthodox' point of view, or at least to limit the points of view to the 6 orthodox philosophical systems [*darśana*] of traditional Hinduism, without entering details of regional sects or elaborate theories of individual teachers. Myths are simplified by telling one variant among sometimes many. Thus precise quotations and references are lacking. Actually, we may even look at the material as an attempt at presenting a synchronic account of indian culture, in contrast to the standard diachronic view of scholars. Thus historic progressions are seldom mentioned, except for a few notions qualified as 'vedic'. We feel that we have reached the limit size for such a simple point of view, and the dictionary structure would have to be revised to account for a more detailed treatment - we shall discuss this point below.

On the other hand, a few strong design principles where adhered to. For instance, we were careful about internal consistency in several ways. First of all, no sanskrit word or sanskrit origin proper name is mentioned in the body of an entry without having its own entry. This property turned out to be crucial for turning the document into hypertext, since it insured the absence of dangling references. Whenever a synonym or antonym is indicated, the symmetric reference is made in the corresponding entries. Finally, the dictionary is complete with respect to etymological origin, to the extent that it is known. Thus the occurrence of a compound word guarantees the listing of its constituents as entries of their own, and so on down to the roots.

3.2 Possible future developments

This work may be continued in several directions. First, of course, more entries may be added to increase the lexicographic database. As I mentioned above, aiming at a considerably significantly bigger lexicon would necessitate an adjustment of the format, so that historical and regional variations would be properly labeled. If we want to record citations in a significant way, we would have to design proper notation, first of all to distribute the citations in the semantic fields, as opposed to attaching them globally to entries. Then it would make sense to have these citations refer directly to the electronic sanskrit corpus available on Internet. This would involve adjusting our format to the Text Encoding Initiative[24], and understanding how to collaborate with the scolars maintaining these corpus databases.

One may also envision entering in this format existing digitalised sanskrit dictionaries. This involves several problems. The first one is a legal aspect, for copyright works which may not be in the public domain. The second one is that the digitalised document must be of good enough quality to retrieve all the information, including differences in font, etc. Then proper indexing has to be applied, which may not be fully automatisable. requiring costly intervention of human specialists. These difficulties have been encountered, and partially solved, by Köln indologists working on the Monier-Williams dictionary.

The positive aspect is that our tools are scalable. The total processing of our 10000 entries takes about 1' on the current generation of PCs. We could thus process in about an hour the most extensive existing dictionaries. It is our thesis that our software tools (ocaml, camplp4) are appropriate to professional linguistic engineering at an industrial scale.

As noted above, we have a structure describing links to synonyms and antonyms. We could make this more systematic, in the manner of Wordnet[5], and go in the direction of building a thesaurus, by defining a hierarchical taxonomy of concepts.

A more ambitious endeavour would be to maintain a multi-lingual dictionary. A most useful task would be to incorporate a sanskrit-english version sharing the overall structure of our sanskrit-french dictionary, and consistent with it at the semantic fields level. This poses several problems. First of all, the granularity of the two dictionaries may be fairly different. Thus an entry or some meaning of an entry may exist in one and not the other. Next, we would have to face the issue of variance of granularity of verbalisation of concepts in different languages due to cultural differences. Furthermore, proper synchronisation tools would have to be designed so that the various lexicons may be maintained and updated asynchronously in a distributed fashion by a community of contributors. Further research is needed to solve these methodological issues, which are faced by any distributed multilingual dictionary project.

A first step in this direction would be to analyse how our abstract syntax is consistent with the Text Encoding Initiative[24] for the standardisation of lexicons. Note that the type declaration interface presented in section 2 is similar to a well-typed Document Type Definition (DTD). We could easily build an XML document valid with respect to this DTD by simply transforming the uniform pretty-printer which we used for showing the structure in section 1.2. All we have to do is to print `C(t1, ... ,tn)` as `<C>t1 ... tn</C>` in the SGML style of displaying trees. Of course XML would allow an extra dimension in the sructure design through annotations.

Another interesting endeavour would be to use our lexicographic database for computational linguistics applications, and ultimately text analysis. Our structure has started to organise grammatical information tagged to entries roles. One may imagine going further and generating a full list of all flexed forms. This would involve writing an internal *sandhi* generator first, a rather simple task. Deriving all declined forms of the substantives would be relatively unproblematic, since there is a small (less than 10) number of regular families, easily classified from the termination of the stem, and the combinatorics is limited (3 genders, 3 numbers, 8 cases). Exceptions for irregular stems would have to be listed, but we already provided many of those. We could encompass pronouns and numerals as well.

For verbs, however, we would have a much more formidable task. In existing grammars we may find a classification of verbs in 10 families, but this is a very simplified account of a rather formidable verb system. Many verbs admit several conjugation patterns of the present system, and the exceptions are so many that a much more complex classification scheme has to be devised for avoiding having more exceptions than regular cases. A quick look at Whitney's roots[16] or advanced grammars such as Renou's shows that the full story is indeed very complex, and we do not know of an easily accessible compilation of a reasonably complete classification, similar say to the Bescherelle[2] compilation for french. For instance, there are 7 different ways of forming the aorist tense. Furthermore, the combinatorics of conjugate forms is here tremendous; even ignoring derived forms, the structure of verbal constructions given in section 2.3 above permits the generation of 31 patterns, to be multiplied by three persons and three numbers. Not all these combinations are used, of course, but they are potentially usable, could appear from some not yet analysed text or through a poetic creation, and it is indeed in the tradition of the language back from the days of Pāṇini in the 4th century B.C. to allow all sanskrit constructions which are derivable from a formal grammar seen as a generative device.

If these problems can be overcome, that is if a list of derived forms can be generated and indexed, we would have the first brick in a program of computer analysis of sanskrit texts. That is, we could use this data base to compile an automaton that would recognize the sequence of words in a sentence. This automaton would be more complicated than the standard lexicographic automata that do string pattern-matching, however, since *sandhi* modifications (phonetic alterations at the junctions of words) would have to be precomputed. Since *sandhi* is not in general a reversible operation, this would add one source of non-determinism to the lexical analysis.

At this point we would have the basic tool to start investigating the parsing of sanskrit sentences. Similarly to many other indian languages, the order of words in a sentence is very flexible, and thus we conjecture that the standard framework of parsing by pattern-matching context-free or mildly context-sensitive rules in western computational linguistics (TAG, categorial grammars, etc) would not be well suited. Approaches using saturation of government slots driven by the verbs should be sufficient for simple sentences, although ambiguities arising from compound substantives would be unavoidable. More complicated sentences would have to be broken by heuristics, such as using absolutives as sequential sentinels. It is of course too early to discuss general solutions to these problems, but we remark that our dictionary provides an embryonic structure to indicate the governance annotations, with the `Gov` constructor of type `word` above.

Here we are touching semantics again. We must return to the dichotomy between *encyclopedic* semantic entries, which are glosses in french of the meaning of a given word, and *substitutive* semantic entries, which offer a translation in french of the corresponding notion. In the latter case, we may provide *templates* producing the translation from other parts of the parse tree, namely from the assumed slots governing the semantic entry. For instance, let us look at a few verbal instances of these templates in our dictionary :

```
\cas \ca{abhisa"msmaarayati} \sem{rappeler qqc. <acc.> à qqn. <acc. g.>}
```

In english, we would say similarly, but with different complement order :

```
\sem{remind  someone <acc. g.> of something <acc.>}
```

Here are a few other typical examples :

```
\word{ga.n}  ... \sem{imputer qqc. <acc.> à qqn. <loc.>}
... \ca{chandayati} \sem{être favorable à, gratifier qqn. <acc.> de <i.>}
\word{niyuj} ... \sem{confier qqc. <acc.> à qqn. <loc.>}
\root{krii} ... \sem{acquérir; acheter (qqc. <acc.> à qqn. <g. abl.>)}
```

In these examples of verbal entries, we propose a translation to a proper french VP (verb phrase) given the translations of needed slots (i.e grammatical *valence*) for this in the sanskrit sentence seen as a permutative template (vocative nominative accusative instrumental dative ablative genitive locative verb). This account is a first approximation of course, since e.g. multiple accusatives are frequent and would have to be discriminated by a taxonomy structure, and genitives are more often part of NP structure than VP. We may think of these semantic entries as coding transformational rules implementing sanskrit to french translation as a typed construction from a permutative grammatical framework[14, 3] to a permutative one. We shall not develop further these ideas here, but remark that there is now general agreement on the importance of the lexicon structure in parsing and parsing-based translation. Note that the electronic dictionary Wordnet[5] has a notion of *verb frame* similar to our templates.

One last remark about genericity of this research towards computer lexicography. The sanskrit-naive reader may wonder why in our structure we forgot the phonetic renditions. Indeed sanskrit is so concerned with good pronunciation that *devanāgarī* (or its latin transcription) is a precise phonetic specification language. We thus do not have to provide a code to the international phonetic notation (neglecting controversies or regional variations of the vowel ṛ, which are irrelevant to our concerns). This is the reward of having to pay the price of *sandhi* alliterations built into the written forms.

Acknowledgements

An abstract of this work was presented in April 2000 at the XIth International Sanskrit conference in Torino, Italy.

References

[1] Abel Bergaigne. Manuel pour étudier la langue sanscrite. F. Vieweg, Paris, 1884.

[2] Bescherelle. L'art de conjuguer. Hatier, Paris 1990.

[3] Akshar Bharati, Vineet Chaitanya and Rajeev Sangal. Natural Language Processing - A Paninian Perspective. Prentice-Hall of India, New Delhi, 1995.

[4] Guy Cousineau and Michel Mauny. The Functional Approach to Programming. Cambridge University Press, 1998.

[5] Christiane Fellbaum, ed. Wordnet, an Electronic Lexical Database. MIT Press, 1998. See also : `http://www.cogsci.princeton.edu/~wn`.

[6] Christopher J. Fynn. How to use Diacritics for Romanised Indic Text on the WWW. `http://www.user.dicon.co.uk/~cfynn/sanskrit.htm`.

[7] Gérard Huet. The Zipper. J. Functional Programming 7,5 (Sept. 1997), pp. 549–554.

[8] Donald Knuth. The TEXbook. Addison-Wesley, 1984.

[9] Leslie Lamport. LATEX - A Document Preparation System. Addison-Wesley, 1986.

[10] Leslie Lamport et al. LATEX 2_ε - The macro package for TEX. Edition 1.6, Dec. 1994. Distributed as part of Version 19 of GNU Emacs.

[11] K. Narayan Murthy. Machine Assisted Translation. Private communication, Dec. 1999.

[12] Anshuman Pandey. Devanāgarī for TEX, version 1.6, Aug. 1998. Available from any CTAN archive site, such as `ftp://ftp.tex.ac.uk/`.

[13] Dave Raggett, Arnaud Le Hors and Ian Jacobs, Eds. HTML 4.0 Specification. W3C Recommendation, April 24, 1998. `http://www.w3.org/TR/REC-html40`.

[14] Aarne Ranta. Grammatical Framework Tutorial. Revised version, Feb. 19th, 1999. Xerox Research Center Europe, Grenoble. Available from : `http://www.xrce.xerox.com/research/mltt/gf/home.html`.

[15] Pierre Weis and Xavier Leroy. Le langage Caml. 2ème édition, Dunod, Paris, 1999.

[16] William D. Whitney. Roots, Verb-forms and Primary Derivatives of the Sanskrit Language. 1885. Repr. Motilal Banarsidass, Delhi, 1997.

[17] Acrobat Reader, an Adobe product, freely downloadable from : `http://www.adobe.com/products/acrobat/readstep.html`.

[18] The Camlp4 preprocessor. See `http://caml.inria.fr/camlp4/`.

[19] The Common Gateway Interface. See `http://www.w3.org/CGI/Overview.html`.

[20] Ghostscript. See `http://www.cs.wisc.edu/~ghost/index.html`.

[21] Objective Caml. See `http://caml.inria.fr/ocaml/index.html`.

[22] PDF, the portable document format, an Adobe product. See : `http://www.adobe.com/products/acrobat/adobepdf.html`.

[23] Postscript, an Adobe product. See : `http://www.adobe.com/print/postscript/main.html`.

[24] The Text Encoding Initiative. See : `http://www.uic.edu/orgs/tei/`.

[25] The Unicode standard 2.1. See : `http://charts.unicode.org/`.

[26] The XML norm, version 1.1, W3C consortium. See : `http://www.w3c.org/documents/XML/`.