UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

How to integrate databases without starting a typology war: the Typological Database System

Dimitriadis, A.; Windhouwer, M.; Saulwick, A.; Goedemans, R.; Bíró, T.

Publication date
2009

Document Version
Author accepted manuscript

Published in
The use of databases in cross-linguistic studies

Link to publication

Citation for published version (APA):
Dimitriadis, A., Windhouwer, M., Saulwick, A., Goedemans, R., & Bíró, T. (2009). How to integrate databases without starting a typology war: the Typological Database System. In M. Everaert, S. Musgrave, & A. Dimitriades (Eds.), *The use of databases in cross-linguistic studies* (pp. 155-208). (Empirical approaches to language typology; No. 41). Mouton de Gruyter. http://www.let.uu.nl/~Alexis.Dimitriadis/personal/papers/TDS-chapter-formatted2b.pdf

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:16 Nov 2026

How to integrate databases without starting a typology war: The Typological Database System

Alexis Dimitriadis, Menzo Windhouwer, Adam Saulwick,
Rob Goedemans, Tamás Bíró

1 Introduction

The Typological Database System (henceforth TDS)[1] is a web-based service that provides integrated access to a collection of independently created typological databases. Thus it is not an original data collection, but an interface to the data contained in its component databases. The TDS web server can be accessed at the URL http://languagelink.let.uu.nl/tds/.

The main challenges in developing the TDS were not, as one might perhaps imagine, due to the technical problem of combining data residing in different software platforms; in fact this only poses minor obstacles. The real difficulties arise from (a) the very large total number of descriptive parameters included in the aggregated databases, and (b) the differences in

---

[1] The TDS Project is being carried out by a research group of the Netherlands Graduate School of Linguistics (LOT), with members from the University of Amsterdam, Leiden University, Radboud University Nijmegen, and Utrecht University: Tamás Bíró (linguistic design and database integration), Alexis Dimitriadis (project manager), Rob Goedemans (database integration and phonology domain expert), Ruth Lind (intern), Adam Saulwick (ontology developer, typologist and database integration), Eugenie Stapert (student assistant), Franca Wesseling (student assistant), Menzo Windhouwer (software system designer and developer). The TDS Project gratefully acknowledges the financial support of the Netherlands Organization for Scientific Research (NWO).

Presentations of various aspects of the TDS have been given at the following conferences: E-MELD 2005 (Dimitriadis et al. 2005), DISWeb 2005 (Saulwick et al. 2005), DGfS 2006 (Saulwick et al. 2006), and elsewhere. We thank the audiences at these conferences for useful feedback and discussions which have contributed to improving the content and presentation of this chapter. We remain responsible for any deficiencies.

structure, terminology, and theoretical assumptions among component databases. The typical typological database contains a very large number of data fields (parameters), often several hundred, about a large number of languages (again in the hundreds). The component databases were created independently of each other, and reflect a focus on diverse aspects of languages, research questions, and theoretical backgrounds.

The TDS Project's approach to data management is central to addressing both challenges. The diversity and sheer quantity of the data make it impossible to deal with differences through some sort of "consensus" representation; indeed, for reasons that will be detailed below, we consider such a goal to be not only unattainable but flawed, with the potential to distort the real content of the data. Instead, the TDS aims to represent the diverse theoretical perspectives faithfully. Purely notational differences are resolved (e.g., if different databases use the values 'yes' and '+' to mean the same thing, these can be converted to the single value 'yes'); but for the rest, the TDS focuses on presenting the contents of the component databases as accurately as practicable. This means that in many cases, divergent classifications or analyses will be presented side by side.

While the total number of database parameters is large, the TDS is not a large-scale data integration project. Approximately a dozen databases are integrated in the initial phase of the Project, and the eventual size of the archive will be in the dozens rather than hundreds or thousands of databases. Hence it was possible to focus on integrating the semantics and encoding of a particular (and progressively extended) set of databases. This is a labour-intensive process, but is justified by the richness of information represented by the data contained in each component database. Such data have been collected one language and one parameter at a time, often following laborious study of the relevant sources. The complex decision-making process involved in creating each component database is reflected in its presentation through the TDS.

In order to be included in the system, the data in each component database are restructured and recoded to the extent that this is possible without loss of information, and organized so as to produce an integrated whole with consistent structuring conventions (as far as this is possible given the diversity of the data). This permits the aggregated data to be effectively navigated. The user interface, data structuring and integration process are supported by an ontology of linguistic concepts developed by the Project for this purpose.

The TDS interface allows users to search for fields of interest, which are then used for querying the data. Searching is thus a two-step process: In the "pre-query" step, the user discovers fields relevant to topics of interest, by using one of several search and browsing options in the TDS interface. Selected fields are accumulated in a sort of shopping basket. In the second step, the collected fields are used to construct and execute a query.

In the next two sections, we introduce the goals of the system and the challenges it confronted. Section 4 presents an overview of the TDS. Sections 5, 6 and 7 describe the system in more detail, focusing first on the knowledge architecture (section 5) and then on the software implementation and user interface. Sections 8 and 9 focus on the practicalities of database integration, as carried out by TDS members. Finally, we close with some discussion and conclusions.

2 Goals of the system

The ubiquity of the internet, which allows data to be shared across large distances efficiently and practically for free, has gradually brought about a development of great importance to typologists: While collections of typological data were once considered a personal tool for the exclusive use of the researchers who compile them, researchers are increasingly coming to view them as resources that can, and therefore should, benefit the academic community. A growing number of typological databases, many originally created for personal or small-group use, are now being made publicly available.

But the growth in the number of such databases, welcome as it is, comes with a cost. As more and more typological databases become accessible to users other than their creators, colleagues, and others already familiar with them, the task of managing and using the information becomes more difficult. We can identify the following kinds of problems facing a linguist who looks for typological information on the web:

1. **Resource discovery**

   This is simply the step of finding a data source with information on some topic of interest.

2. **Correct and effective use**

   As already mentioned, databases use varying terminology, notation, organization of the data, and search commands. Even if these are

documented in detail, they can be quite difficult for a new user to assimilate and employ properly; and proper documentation is not always available.

3. **Efficiency of resource utilization**

   As the amount of online information grows, the time and effort involved in searching databases one by one and collating the results becomes an obstacle to their efficient utilization.

The problem of resource discovery is being addressed by language archives, which collect a large amount of linguistic resources in one place (hopefully well-known or easy to find), and by numerous initiatives that are developing improved methods for resource description and discovery. Generally these include enriched standards for resource description, which can be utilized by existing or new tools.[2] The TDS does not directly address this problem, except by providing a (small) number of databases collected in one place.

The TDS directly addresses the second and third tasks. The goals of the system are (a) to provide an interface that will help users *find* relevant data, and (b) to enable users to *interpret* the data they are presented with. The TDS interface allows speedy combined searches over the data in its collection of typological databases, from a single user interface and using (as much as possible) consistent terminology and encoding conventions. The results are fully aggregated across databases, and can be displayed in a number of formats, including (so far) two export formats, as an XML (Extensible Markup Language) document or as a single table in comma-separated-values (CSV) format. Interpretation of the data is aided by presenting documentation for each database field, supported by documentation on any linked linguistic concepts in the TDS global ontology. Since data in

[2] Metadata-oriented initiatives include the Dublic Core Metadata Initiative (DCMI, at http://dublincore.org/), the Open Archives Initiative (OAI, at http://www.openarchives.org/), and the linguistics-specific initiatives of the Open Language Archives Community (OLAC, at http://www.language-archives.org/) and the International Standards in Language Engineering (ISLE, at http://www.ilc.cnr.it/EAGLES/isle/). A different approach to resource discovery, based not on metadata but on sophisticated pattern matching, is followed by the Online Database of Interlinear Text (ODIN, at http://www.csufresno.edu/odin/; Lewis 2006).

the TDS is often separated from its original context, in all cases the interface must present the provenance of the data along with its database-specific description; this allows users to properly evaluate the information retrieved.

3 The problem

In creating a single data resource from the collection of typological databases, the TDS Project needed to address the various kinds of differences among them. These can be of several kinds:

1. **Different types of content**

   So-called "analytical" typological databases consist of logical variables describing each language as a whole; for example, "language X has/does not have subject-verb agreement." Other databases contain example sentences with detailed annotations ("sentence databases"), or a combination of both types of information. The Project attempts to integrate different types of content so that, for example, a single query can search both examples and logical variables for relevant information.[3]

2. **Different theoretical commitments**

   There exists, of course, no universally accepted and descriptively exhaustive linguistic theory. Thus, the information in each database reflects the analytical and theoretical commitments of its creators. For example, the notions *Subject* and *Object* are used in several different ways by linguists, and there are alternative ways of expressing structural relationships, e.g., the S/A/P/R categorization as recently applied in Haspelmath (2005).[4] Conversions between theories, it turns out, cannot be automated with reliability; and database creators do not want their theoretical commitments to be misrepresented. Hence the TDS places a high priority on preserving and presenting to the user the framework of database-specific assumptions needed to properly interpret the data extracted from a component database. Such information will allow knowledgeable users to recognize both the descriptive con-

[3] The system includes both kinds of data, but such cross-type searches are only partially supported at this time.

[4] This example is discussed in more detail in the following section.

tent and the theoretical commitments of a statement, regardless of whether it matches their own theoretical orientation. It can be beneficial for users to view information even if it is not expressed in terms of their own theoretical framework. For example, information about properties of "subjects" can be useful even to those who do not believe that this is a typologically sound notion.

3. **Constructed for different purposes**

   The focus and detail of coverage of individual databases vary depending on the creators' own research interests, even where there are no theoretical disparities (or no significant ones). Such variability can lead to significant differences in the structure, content, and degree of detail in conceptually similar data.

4. **Different notational conventions**

   In many cases the different databases use equivalent, or near-equivalent, ways of describing data. An obvious example is the use of different glossing labels for broadly accepted linguistic categories, such as "p" or "pl" for *Plural*. It is generally easy to reconcile purely notational differences, but the databases can also differ in the details of how such concepts are defined and applied. It is thus necessary to distinguish notational variation from theoretically important differences, and to resolve differences with respect to the former but not the latter.

5. **Different design choices**

   There are a myriad of ways to organize a body of information into a database, and the component databases therefore differ markedly in their structure. As this source of variation is compensated for, it becomes easier to address the more troublesome types already discussed. Consistent design choices also make it easier for the end user to gain an understanding of the data, compared to dealing with multiple conventions that must each be learned separately.

6. **Different software**

   The TDS component databases were developed with, or for, a variety of database management systems (DBMSs) that currently include Microsoft SQL Server, MySQL, Microsoft Access, Excel, 4th Dimension, and custom-made database software. Their origins in different software environments (operating systems, fonts, storage formats, etc.) introduce additional complications. The TDS uses a plug-in architecture to im-

port data from this plethora of formats; fortunately, it has been possible to find interface modules or exchange formats for each database included. Occasional relatively small problems (such as font-encoding glitches) can require manual intervention in the form of ad hoc fine-tuning scripts, but these are quickly resolved and only need to be addressed once per database. In practice, dealing with this kind of variation does not pose great difficulty.

These sources of variation must be dealt with in different ways. The TDS approach distinguishes between variation in structure or encoding, which is judged to be a design choice of no inherent linguistic significance, and variation in the choice of linguistic terms and (especially) categories and distinctions. Broadly, we can speak of differences in encoding and differences in meaning (semantics). While the metadata initiatives mentioned earlier might one day lead to more uniformity in structure and encoding among databases, they will have no effect on the divergence of theoretical viewpoints and research traditions that constitutes the most intractable source of heterogeneity. These diverse viewpoints are not only dearly held by their practitioners: They are the subject matter and outcome of linguistic analysis, and cannot (indeed, should not) be replaced by any uniform, agreed-upon framework.

During the early stages of the Project, consultation with the community of database creators and prospective users established that preservation of the specific claims made by the creators of the component databases is of the utmost importance. Full harmonization of the collected data into a common form would lead to unacceptable distortion in the accuracy or precision of the data. Conversely, a multiplicity of alternative models and classification schemes for data originating in different databases is acceptable, as long as the model applicable in each case is made explicit.

Accordingly, the TDS approach is as follows: encoding differences are compensated for wherever possible, by transforming the source data to adhere to, or at least be relatable to, a uniform design ("object model"). Semantic divergences are maintained, and are made explicit by suitable documentation and careful construction of relationships between various levels of metadata.

We should point out here that the Project takes a neutral standpoint towards the data in the component databases; that is, we do not consider it the job of the Project to check and correct the data in the component databases, or to

make value judgments on the analyses they express. The developers of the component databases have devoted much time and effort to collecting information in their databases; in database terms, the component databases represent high-value information reservoirs, created through considerable human effort and utilizing extensive domain expertise. In short, each component database represents an extremely valuable resource. The Project must ensure that the contents of the component databases are accurately imported and presented in the TDS, but it is not responsible for the accuracy of those contents themselves. To do otherwise would be impossible, since the TDS could not assume responsibility for the accuracy of the data without undertaking to resolve the empirical and theoretical differences between its component databases. As long as the provenance of information provided by the TDS is explicit, end users will be able to assess its suitability and reliability for themselves.

3.1 The notion Subject*: a mini case study*

To better illustrate the integration process across databases, we will consider a simple case involving a single descriptive parameter. Consider a query such as "which languages have subject-verb agreement." Two of the component databases contain information on "subjects." One database, the Typological Database Nijmegen (henceforth TDN),[5] contains a single boolean variable answering exactly this question; another database, the Person Agreement Database (henceforth PAD), includes a block of variables giving more information about subject-verb agreement, for languages in which this exists. A complicating factor is that the notion *subject* is not a primitive in the PAD. Instead, the PAD relies on a common alternative, the four-way classification of grammatical functions as *sole argument of an intransitive verb* (S), and *agent-like* (A), *patient-like* (P), and *recipient-like* (R) *arguments of a transitive verb.*[6]

How can this classification be used to get information on subject-verb agreement? We must define a query in terms of the available categories.

---

[5] See section 4 for descriptions of the component databases.

[6] The introduction of a category S distinct from A and P is found in Dixon (1972) and Comrie (1978). A recent adaptation of the system can be found in Haspelmath (2005). For discussion of the problems associated with *subject* as a cross-linguistic category, see Comrie (1989: ch. 5).

The common notion *subject* is itself interpreted in different ways within the linguistic community, but in principle it would be expected to be co-extensive with the union of the categories S and A.[7] The most useful strategy, then, is to search for data on S and A controllers. However, there are enough differences in principle and in practice between theories (and between the practice of individual researchers), that an implicit mapping of S and A values to the value *Subject* could lead to erroneous information being reported (for example, if an A by someone's definition is not a *Subject* by someone else's). Therefore this decision is best left to the user.

Let us now consider the reverse situation: A linguist who subscribes to the four-way classification of grammatical roles wants to query for languages which have agreement with A (the agent-like argument). Since one of the component databases does provide this information, the TDS should make it available. But the information in TDN cannot answer this question directly, since TDN does not distinguish between transitive and intransitive subjects. Does that mean that the information in TDN is of no interest in this case? Only the user can answer this; information about the category *Subject* might be useful, although inexact, or it might be irrelevant to the user's needs.

The TDS interface respects these considerations. A user who searches for the term "subject" during the pre-query stage is presented with a list of da-

---

[7] There is considerable disagreement among linguists as to what constitutes a subject, and little of it can be resolved by adopting the S/A/P/R system. For example, consider an experiencer predicate like Spanish *me gustas tu* 'I like you', which assigns dative to the experiencer and nominative to the stimulus. Does this predicate have a "dative subject," as suggested by the thematic relationships and the word order, or should the title "subject" be applied to the nominative-marked stimulus, which also controls "subject agreement" on the verb? Should the answer be the same in all languages with a similar construction? The issue depends on one's underlying principles, definitions, and the desired abstractness of analysis, and is amplified when one considers languages with fewer morphosyntactic clues, or with less well-understood grammars. Adoption of the S/A/P/R system allows ergative languages to be coherently discussed but does not resolve such questions, since one must still determine what is or is not "agent-like," etc.

Fortunately, these issues tend to have little impact on the determination of analytical properties of a language, such as pro-drop or basic word order; these generally depend on the typical or prevailing configurations, for which there is less disagreement on identifying the "subject" (or A).

tabase fields that includes the relevant fields from both databases (this happens even though the documentation directly associated with the PAD fields does not use the word "subject"). The user can assess the relevance of each field for his or her purposes. Presented with the available options, a user interested only in agreement with A might decide to rely exclusively on PAD or to additionally look up information on subject agreement in TDN, later refining it by consulting other information on individual languages.

There is another reason to be conservative in transforming data from one descriptive framework into another: the creators of PAD have made a deliberate choice not to rely on the category *Subject,* which they consider problematic and inadequate; they might not be keen to endorse a statement to the effect that "according to the PAD, language X has subject-verb agreement," since the PAD does not directly make any statements about the category *Subject.* In reporting the contents of its component databases, the TDS must be careful to do so accurately.[8]

Note finally that these issues were illustrated by reference to a very simple example involving a single descriptive parameter. The problem becomes even less tractable when numerous data fields are used to describe aspects of a phenomenon, or when more than one linguistic concept is involved. For example, the Typological Database Amsterdam (henceforth TDA) database encodes subject-predicate order in terms of the following parameters:

1. Basic word order of the clause
2. Whether the order is fixed or variable
3. Whether non-canonical order is morphologically marked

The basic word order is not expressed in the traditional terms of subject, verb and object position (e.g., SVO or SOV), but in terms of the position of the predicate: Predicate-initial, -medial or -final, with the option of having

[8]An accurate report is not necessarily verbatim, however; the issue here is the use of disputed terms or categories. It is sometimes necessary to develop descriptions where none exist, or to expand, in consultation with database developers, on descriptions in the original documentation so that they are interpretable when removed from their original context. We take care that the resulting statements do not violate the theoretical commitments of the database they describe.

multiple orders in a language (cf. Hengeveld et al. 2004). Because multiple orders are allowed, the approach contrasts with a system that identifies *one* basic word order, and perhaps identifies a *secondary* order on the basis of particular criteria. Note that the question of one versus many basic orders is independent of, and more complicated to resolve than, the choice between the traditional "constituent-based" and predicate-based order. Neither system can be converted to the other without loss of information (or the manual addition of information not already in the database).

4 The Typological Database System

The TDS Project began with the goal of unifying a number of typological databases, originally all by Dutch linguists.[9] The Project's early attention to terminological and conceptual differences has evolved into the present focus on developing a software system whose architecture is fundamentally based on the principles of data integration described here. Monachesi et al. (2002) present an early vision of the system.

It has always been the intention to include a moderate number of databases, whose integration into the system would require specialist expertise. At present the following databases are searchable through the TDS interface, and a few others are being prepared for integration.

1. The **Anaphora Typology** database project (Utrecht University) is surveying the binding properties of reflexives and reciprocals, referred to collectively as "local coreference strategies." This database is focused on reflexives, although some information is also provided on pronouns and reciprocals. It contains glossed example sentences (grammatical or ungrammatical) in a variety of syntactic configurations. A limited number of languages are examined in detail. Only a few languages and properties are currently included in the TDS. Developers: Alexis Dimitriadis, Martin Everaert, Eric Reuland, Tanya Reinhart; Utrecht institute of Linguistics OTS.

[9] The early participants in the Project included Paola Monachesi, Anne-Marie Mineur and Manuela Pinto, as well as several of the participants named in footnote 1. The developers of the initial group of component databases played an important role in defining the goals of the system, and identifying problems and areas of concern.

2. The **Person Agreement Database** (PAD) contains analytical (language-level) data for over 400 languages, on person agreement and some related areas such as word order. The information is coded in terms of over 250 variables. For a subset of these, citations to relevant pages in reference grammars are given. Developers: Anna Siewierska, Lancaster University; Dik Bakker, University of Amsterdam.
3. **Smith's Phoneme INventories** (SPIN) is a collection of phoneme inventories and lexical tones in 111 languages based on published works. Developer: Norval Smith, University of Amsterdam. Digitized by the TDS Project.
4. The **Stress Typology Database** (StressTyp) contains information on the metrical systems (stress systems) of 510 languages, based on grammars and theoretical works. Notions covered include rule-based stress, lexical stress, extrametricality, foot types, etc. Developers: Rob Goedemans, Leiden University; Harry van der Hulst, University of Connecticut. See Goedemans and Van der Hulst (this volume).
5. The **Syllable Typology Database** (SylTyp) contains information on syllable structures. Restrictions and rules concerning possible syllabic structures are provided, as well as information pertaining to the content of these structures. Developers: Harry van der Hulst, University of Connecticut; Rob Goedemans, Leiden University.
6. The **Typological Database Amsterdam** (TDA) focuses on basic word order and constituent order systems. Information classifying the parts-of-speech system of the included languages is also provided. Developer: Kees Hengeveld, University of Amsterdam.
7. The **Typological Database Nijmegen** (TDN) contains analytical (language-level) information on a variety of topics, including: basic word order, intransitive predication, case marking, temporal sequencing, relative clause information, comparatives, possessive constructions, verbal morphology, tense/aspect, noun phrase coordination, manner adverb encoding, verbal derivation. The number of languages varies from topic to topic, with a minimum of 140 for all topics, and a maximum of 410 for some. Developer: Leon Stassen, Radboud University Nijmegen.
8. The **Typological Database of Intensifiers and Reflexives** (TDIR) provides information on intensifiers and reflexives as well as on some related domains of grammar such as the middle voice and scalar focus

particles. Its focus is on accurate language description and documentation ("grammar fragments"). The data have been obtained from both native speaker consultation (primary data) and literature on the relevant topics and languages (secondary data). For secondary data, sources and references are generally indicated. The database contains information on more than a hundred languages, including approximately 600 examples. The sample is not balanced genetically or areally. Developers: Volker Gast, Daniel Hole, Ekkehard König, Peter Siemund, Stephen Töpper; Free University of Berlin. See Gast (this volume).

9. The **UCLA Phonological Segment Inventory Database** (UPSID) is a collection of phoneme inventories for 451 languages. Features such as manner, place, length, phonation type and secondary articulation are included. Developer: Ian Maddieson, UCLA.

10. The **Graz Database on Reduplication** provides data on reduplication in the world's languages. The collected examples are described phonologically, morphologically and semantically, together with information on productivity and diachrony. Developer: Bernhard Hurch, University of Graz, Austria. See Hurch and Mattes (this volume).

11. The **World Color Survey** elucidates the relationship between color categories and basic color terms. The summary tables included in the TDS have originally appeared in the World Atlas of Language Structures. Developers: Paul Kay, Luisa Maffi; University of California at Berkeley.

In addition, the TDS includes a number of supporting data resources that are not themselves considered linguistic databases. These include the table of languages and three-letter codes defined by ISO standard 639-3,[10] a table of genetic affiliations for each language as assigned by the Ethnologue directory (used by permission), and two collections of geographic locations (GIS coordinates) for several hundred languages: One originally compiled by Matthew Dryer and updated for use in the World Atlas of Language Structures (see Haspelmath, this volume), and another, limited to African languages, compiled by Guillaume Segerer. Both are used by the kind permission of their creators. The Universal Phoneme Position Chart (described

[10] The ISO 639-3 codes are the successor to the "SIL codes" used in the past by Ethnologue (http://www.ethnologue.org/). SIL is the registration authority for the new ISO codes, and provides an access point at http://www.sil.org/iso639-3/.

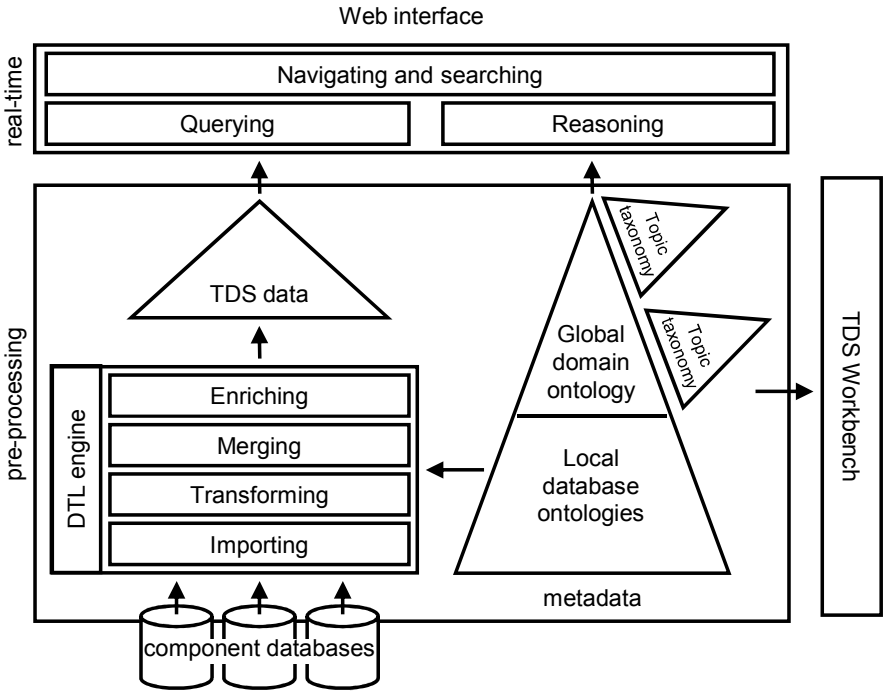

Figure 1 *Overview of the TDS architecture.*

in Section 8.2) is based on data contained in the UPSID database, but involves significant additional processing by the TDS; it can be seen as an additional information source.

Together, the component databases of the TDS contribute more than twelve hundred database fields (attributes), with varying amounts of information on almost one thousand languages.

4.1 System Architecture

Figure 1 gives a global overview of the TDS architecture. Data are imported from the component databases (at the bottom of the diagram), and the user interacts with the system through its web interface (at the top). The core of the system is the metadata, which contain all TDS knowledge about the component databases and the linguistic domain. The knowledge base consists of various specifications which are linked to each other: a set of

database-specific ontologies, one global linguistic ontology and (currently) two topic taxonomies. Maintenance of the knowledge base is supported by the TDS Workbench (right) and other custom-built or general-purpose tools, including an ontology editor.

The functions of the TDS are divided into two core processes: Loading and integration of the data from the component databases, and querying the data in response to user requests.

The first core process, integration of the data from the component databases, is implemented as a pre-processing step that is executed whenever the contents of the TDS change. For example, if a new database is added, or an updated version of an existing component database is received from its developers, or changes are made to existing metadata. Data integration involves the following steps: importing, transforming, merging and (possibly) enriching.[11] The process is completely controlled by database-specific specifications written in the Data Transformation Language (DTL),[12] a declarative description language developed by the TDS Project for this purpose (see section 5.2). The DTL specifications tell the data integration engine about the implementation details of each component database, e.g., which DBMS is used, where the database is located, and which encoding is used. Using this information, the engine selects the proper plug-in to load the data from the component database. Once data are loaded, they undergo a transformation process. The transformation rules are also part of the DTL specification and define the unification of different notations and structures, to the extent that this is possible without compromising the semantics of the component database. Special care is taken in the harmonization of key values, e.g., language and phoneme codes. This is necessary in order to recognize when different databases describe the same object (language, phoneme, etc.) Once the keys are properly harmonized, the next step is to merge all the data about one object, e.g., a specific language, from the various component databases. Next, cross-database data enrichment can take

[11] Although the data integration process is described in this section as sequential, implementation-wise the steps are interleaved, e.g., local database enrichments happen together with the data transformation.

[12] Important TDS terms and acronyms are collected in a glossary at the end of this paper.

place.[13] The end result of interpreting the specifications in the DTL scripts is a unified data collection, the TDS data.

Management of the DTL scripts and other TDS metadata is facilitated by the TDS Workbench, which provides various checks on the consistency of the metadata network; for example, it detects invalid links between the various specifications, and invalid links and structures within the global ontology.

The second core process of the TDS is support for end-user queries over this data collection. The query process is steered by a web interface, which helps the user build and execute queries in two stages. At the first stage, the pre-query, the user interacts mainly with the metadata. By navigating through the network of metadata elements and/or doing full-text searches on their content, the user can identify fields of interest and collect them into the *query basket.* In this pre-query step, the system conducts "smart searches" that exploit the sense relationships encoded in the metadata (especially the ontology and DTL schema). Search terms, for example, use the vocabulary of a user's theoretical framework. By locating these terms in the semantic network, the system can suggest fields to the user which are linked to semantically close terms, even if they are drawn from an alternate theoretical framework. For example, the search term "predicate medial" partially matches the ontology Concept[14] *PredicateMedialWordOrder*, which is part of the Hengeveld et al. (2004) classification of constituent orders (see section 3.1). The ontology provides links to the related Concepts *OVS* and *SVO*, even though these categories only occur in the alternative, three-part classification of constituent order systems. The TDS can now also suggest fields related to the concepts *OVS* and *SVO,* although they will have a lower ranking in the search results than fields directly related to the Concept *PredicateMedialWordOrder*.

Once the user is satisfied with the collection of fields, the query proper is defined by specifying selection and projection criteria, i.e., which records

---

[13] At time of writing, the TDS supports local database enrichment (new computed fields for a database, with possible reference to global TDS data), but no cross-database enrichment (i.e., enrichment that utilizes merged data from multiple component databases).

[14] We use the term *Concept,* with a capital *C*, to refer to an entry in the global linguistic ontology, representing a linguistic concept. (See section 5).

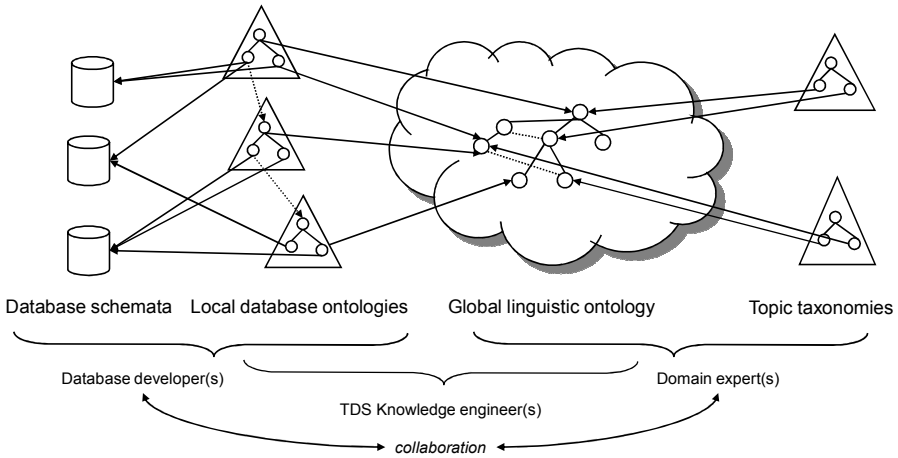

Figure 2 *TDS knowledge architecture with knowledge representation roles.*

to retrieve (by matching the selection criteria) and which fields of these records to display ("project"). The query can now be executed by the system, and the retrieved subset of the TDS data is presented to the user. Once more the metadata assists the user in interpreting the results. The user can then initiate a fresh query, or modify the current query and resubmit it.[15]

5 Knowledge architecture

A key feature of the TDS is its focus on safeguarding the semantic integrity of the integrated data. The TDS Project considers it crucial that the knowledge residing in component databases should be faithfully preserved during the integration process. To support diversity in the theoretical perspectives of component databases, the TDS utilizes a "hybrid," or two-level, model of the semantic domain (Stuckenschmidt and Van Harmelen 2005). In the hybrid TDS model, there are two main levels where knowledge is stored: the global ontology of broad linguistic concepts (henceforth TDS-GO) and the local ontologies of the individual component databases, encapsulated in the DTL schema.

[15] The query interface is described in more detail in section 7.

The global level provides ontology entries, or *Concepts,* which describe general unifying linguistic concepts. The local level includes pointers to these general Concepts, and contains database-specific definitions. The global ontology thus introduces common ground, which enables the possibility of significant cross-database queries, whereas the local ontology is the store of the database-specific knowledge. This model, depicted in Figure 2, forms the core of the TDS knowledge integration.

Each component database is associated with its own local ontology. The local ontology is specified using the DTL, and defines local *Notions,* or entries in the local ontology, by reference to the tables and attributes in the schema of the corresponding component database (typically a relational database). In general, each database attribute (field) is imported and expressed as a Notion; but the DTL also supports powerful means of restructuring, combining, or even splitting up attributes when necessary, and hence the mapping of attributes to Notions is not always one to one. In the DTL specification, a Notion is described through metadata in the form of short labels, more detailed descriptions, and links to the global ontology.

Topic taxonomies provide domain-specific hierarchies as quick entry points into the global linguistic ontology. The system supports multiple alternative taxonomies, so that a linguistic domain can have its own, dedicated search template.[16]

5.1 Database schemata and metadata

The eleven databases currently comprising the TDS exemplify a range of approaches to data modeling. Several of them have properly normalized, or even over-normalized database schemas, while others are completely unnormalized:[17] they consist of a single large table, store multiple pieces of information in one attribute value, etc. Some databases are sparsely filled with data, and could be considered semi-structured, while others are highly

[16] Alternative taxonomies are not extensively used at this time. Only two taxonomies, a general linguistics-oriented one and a taxonomy exposing the organization of the TDS global ontology, are included in the interface. A third, taxonomy-like view ("view by datatype") exposes the native hierarchical organization of the TDS data.

[17] A database is said to be "normalized" if its table structure meets certain technical criteria (see Date 2004).

structured and densely filled. Implementation details of the DBMS used for the component database (either currently or in the past), or the needs of the user interface, are also sometimes apparent in the data. For example, proprietary font encodings are used for IPA characters, and the data structure and values of one database transparently reflect its original status as a dataset for the statistical package SPSS. The DTL transformation rules can address such quirks. To make the imported data useful, however, the intended semantics associated with the encoding needs to be made explicit. As mentioned, most of these databases were created for personal or small-group use and so very limited interpretive metadata is generally available. Even when this exists, it is usually not precise enough for the requirements of the TDS, where a data field may be presented out of its original context of forms or related fields.

In the metadata development process, the interpretive/analytical intentions of the database creator may be discerned by examining the original user interfaces (where these exist), but typically, repeated and sometimes extensive interaction with the database developer(s) is required in order to accurately represent their intention in the local ontology. These analyses form the core of the knowledge component and represent the semantic characterization of the component databases, and by extension the linguistic knowledge in the TDS.

Once a component database has been adequately described in the DTL, the specification is used to drive the importation of data from the component database into the aggregated TDS data. This is a fully automated process that can be repeated whenever the contents of a component database change. Manual intervention is only required if new fields are added to the database (which will need to be described in the DTL schema), or if changes are made to the naming or definition of database fields or *potential* values. Simple addition of more data by the creators of the database, or correction of errors (or other modifications) in existing data, do not require attention by the TDS developers if the database design remains the same.

5.2 Local database ontologies and the Data Transformation Language

The local ontologies should capture theory-specific knowledge. As its name reveals, the Data Transformation Language (DTL) started out as a declarative specification of transformation rules to overcome notational and structural differences. But since the description of the resulting data schema

is part of this specification, the language was extended with constructions that enrich the schema with metadata. Notions are the basic building blocks of a DTL specification. They can express a database-specific theoretical construct, an individual field from a database (or a constructed field), or a group of other Notions. The following kinds of metadata can be attached to each Notion:

1. a short label, preferably about five words long
2. a short description
3. one or more links to Concepts in the global ontology; these links can be labeled with a type, to indicate different kinds of relationships
4. one or more links to other DTL Notions
5. a semantic data type[18]

Notions can be nested and thus form a hierarchy, or rather several hierarchies with distinct roots. Each hierarchy is subdivided into smaller semantically coherent sub-hierarchies. Such a sub-hierarchy forms a local semantic context; Notions should always be shown within their context, or at least the context should be available to the user to allow the proper interpretation (and disambiguation) of displayed data from other Notions in locally nested hierarchies, potentially with the same name. For example, it is clear that the Notion *name* in the context of the Notion *author* represents an author's name (rather than a language name). A powerful aspect of this is that it allows a descendant Notion to inherit some of the sense properties, e.g., the links to Concepts, from its parent and ancestor Notions. This, in turn, prevents local overspecification.

[18] The DTL supports two basic types: enumerations and free text. Subtypes of these types, associated with particular semantics, can be declared and attached to Notions. Their main use is to influence the rendering of Notion values in the user interface. For example, the TDS specification declares a special data type for phonemes, which allows them to be rendered in their proper place in the Universal Phoneme Positioning Chart (UPPC).

```
1.  TOP NOTION tdn:locationalPredicates
2.    LABEL       "Locational predicates"
3.    DESCRIPTION "Information concerning locational predicates, including form of,
4.                 and conditions on, construction, and form of the negation."
5.    LINK TO CONCEPT locationalPredicate
6.    GROUPS {
7.      NOTION tdn:ZeroEncoding
8.        LABEL "Locational predicate is zero“
9.        LINK TO CONCEPT conditionsOnEncoding
10.       GROUPS {
11.         NOTION tdn:v168_Zero_plus_locative_prepositional_phrase
12.           LABEL "Locational predicate is zero + locative prepositional phrase"
13.           DESCRIPTION "The locational predicate is expressed without the use of
14.                        an overt verb, but has a locative prepositional phrase."
15.           VALUE IS FIELD v168
16.           GROUPS WHEN "yes" {
17.             NOTION tdn:v169_Zero_for_present_only
18.               VALUE IS FIELD v169;
19.             NOTION tdn:v170_Zero_in_positive_sentences_only
20.               VALUE IS FIELD v170;
21.           }
22.       }
23.   }
```

Example 1 *An excerpt of the DTL Notion hierarchy for the TDN database.*

Example 1 shows an excerpt from a DTL sub-hierarchy for Locational Predicates. The keywords *TOP NOTION* indicate the start of a sense context. The Notion *tdn:locationalPredicates* has a label, a description and a link to one Concept in the global ontology, *locationalPredicate*. The nested Notion *tdn:ZeroEncoding* also contains a link to one Concept, *conditionsOnEncoding*, but will also inherit the link to the Concept *locationalPredicate* from its parent. The same example also shows the link from the DTL specification to fields in the database schema: the Notion *tdn:v168_Zero_plus_locative_prepositional_phrase* takes its value from the TDN database field labeled 'v168'.

Although each database contains specific, and perhaps unique, Notions, some parts of its DTL hierarchy will overlap with the DTL Notions of other databases. To express such relationships, the DTL supports scopes. Each database has its own scope, where it creates its own Notions. Example 1 shows a part of the *tdn* scope, the scope of the *Typological Database Nijmegen*. Like Notions, scopes form a hierarchy and inherit Notions declared in parent scopes. While the leaves in this hierarchy are formed by database scopes, the root is formed by a "warehouse"[19] scope, the *tds* scope. Com-

[19] The term *warehouse* refers to data warehousing, a computer science discipline

```
1.  WAREHOUSE tds {
2.    DECLARE NAME "Typological Database System";
3.    …
4.    DECLARE ROOT NOTION tds:Language
5.      DESCRIPTION "All linguistic and non-linguistic information about a
6.                   particular language."
7.      GROUPS {
8.        TOP NOTION tds:LangIdentification
9.          LABEL "Language identification"
10.         DESCRIPTION "Information concerning the identification and identity
11.                      of a language: Name, georgaphical area where it is spoke
12.                      etc. Properties that are not part of the synchronous
13.                      description of its system."
14.         LINK TO CONCEPT languageInformation
15.         GROUPS {
16.           NOTION tds:Name
17.             LABEL "Language name"
18.             LINK TO CONCEPT language
19.             TYPE IS TEXT;
20.         … }
21.     … }
22. …}
```

Example 2 *An excerpt from the DTL Notion declarations showing the warehouse scope* tds.

mon Notions and structures such as *tds:Language* and *tds:languageIdentification* are declared in this scope, as shown in Example 2.

Between the warehouse scope and the database scopes there can be domain-specific scopes, which declare domain-specific Notions. For example, there is a scope for the linguistic domain on *phonetics*. Together, the upper scopes define a skeleton hierarchy, or hierarchies, of Notions. The lowest database scopes then localize these declared Notions by associating them with database fields and database-specific transformation rules, and further extend the structure with database-specific Notions. Example 3 shows the reuse of Notion structures from the *tds* warehouse scope in the *stressTyp* database scope, and shows how database-specific Notions are embedded in this structure. The local, database-specific ontologies are thus aligned in their structure, which allows the TDS to integrate the data loaded from the component databases while maintaining (large) data islands of theory-specific information.

---

which focuses on the management of integrated views of multiple (heterogeneous) databases.

```
1.  DATABASE stressTyp {
2.    DECLARE NAME "Stress Typology Database";
3.    …
4.    ROOT NOTION tds:Language
5.      KEY IS LOOKUP(MAP code(FIELD Eth15,FIELD Dialect-name,FIELD Name))
6.      GROUPS {
7.        TOP NOTION tds:LangIdentification
8.          GROUPS {
9.            NOTION tds:Name
10.             VALUE IS FIELD Name;
11.           … }
12.       TOP NOTION tds:Phenomena
13.         GROUPS {
14.           TOP NOTION tds:Phonology
15.             GROUPS {
16.               NOTION tds:MetricalPhonology
17.                 GROUPS {
18.                   TOP NOTION stressTyp:generalStressAssignment
19.                     LABEL "General stress assignment properties and parameters"
20.                     DESCRIPTION "A collection of general parameters and
21.                                  properties describing stress placement
22.                                  patterns."
23.                     LINK TO CONCEPT stressPlacementProperty
24.                     GROUPS {
25.                       TOP NOTION tds:stressRulesDescription
26.                         VALUE IS FIELD Quotation;
27.                     … }
28.                 … }
29.             … }
30.         … }
31.     … }
32. …}
```

Example 3 *DTL Notion hierarchy with Notion localization.*

The relational database model, which underlies the design of most of the component databases, uses a separate database table for each *entity type,* or type of object being described (e.g., language, book, sentence). The TDS equivalent is a so-called *Root Notion*; each Root Notion represents the root of a separate hierarchy of Notions, and includes all properties of the object being described. There is a very limited number of such hierarchies. At present, the TDS contains five: for *languages*, *potential phonemes*, *example sentences*, *glosses* and *bibliographical references*. Instantiations of these Root Notions (for example, a particular language and a particular book) can be related to each other through foreign key relationships, and can be in a many-to-many or one-to-many relation.

The relational database model does not support hierarchical grouping of the database fields (attributes) in a database. While a table's attributes may be conceptually organized into groups and subgroups with related content, this is not expressed in the design of the database.[20] But the data model of the TDS is hierarchically organized, thereby allowing groupings of data fields to be directly expressed as hierarchies of Notions. This model also supports multiple values for a data field without needing a separate table/hierarchy, as would be necessary in a relational database. This has made it possible to replace various component database tables and special data fields, whose only purpose was to accommodate multiple values, with a single TDS Notion that can simply be instantiated multiple times for a single record (e.g., multiple constituent orders for a single language).

5.3 The global linguistic ontology

As stated in section 2, the primary task of the global ontology of linguistic concepts (TDS-GO for short) is to facilitate integration of diverse typological databases containing information on (the analysis of) linguistic facts. To do this, the TDS-GO specifies the domain vocabulary by defining Concepts with descriptions of various linguistic terms and concepts (including information on the logical structure of complex concepts). These descriptions are intended to be descriptive and explanatory of linguistic concepts, but neither exhaustive nor theory-neutral. They are designed to define the Concepts which express and unify local ontology Notions.

The TDS-GO is not meant to include a comprehensive compendium of linguistic concepts; We adopt the bottom-up principle of ontology development (see section 5.3.1), and the TDS-GO only includes information that is relevant, directly or indirectly, to the component databases comprising the TDS.[21] Some parts of the TDS-GO contain highly detailed and hierar-

[20] Some component databases achieve a limited grouping effect by creating a separate table for each group of related attributes; this is unnecessary from the relational perspective.

[21] The TDS-GO is not aligned with GOLD, the General Ontology for Linguistic Description (www.linguistics-ontology.org). GOLD is targeted to concepts related to morphosyntactic annotation, and therefore does not cover many of the topics needed for the TDS global ontology. In addition, GOLD was still evolving at the time the TDS ontology was developed; consequently, the TDS-GO was developed independently of GOLD, rather than as an extension of it. We plan to align the

chically deep Concepts, representing a range of linguistic phenomena in phonology, morphology, syntax, semantics and pragmatics, from a synchronic or diachronic perspective, with particular focus on the following areas: grammatical agreement, parts of speech, word and constituent order, stress placement, predication, phonemic and phonological properties, semantic categories, relational categories (such as case, grammatical alignment, valencies, event types and modification), speech styles, syntactic categories, paradigmatic and systemic groupings, as well as geographic location and genealogical classification. Three such examples of TDS-GO class hierarchies (where "→" represents the is-a relation) are:

1. TDS-GO ontology → linguistic property → phonetic or phonological property → syllable structure property → onset feature → obligatory onset
2. TDS-GO ontology → linguistic property → phonetic or phonological property → suprasegmental property → stress placement → main stress placement → variable stress placement systems → non-lexical stress placement → edge placement → right word edge stress placement → antepenultimate if heavy, else penultimate if heavy, else antepenultimate.
3. TDS-GO ontology → linguistic property → linguistic functions property → marker function → agreement marker function → agreement marker for core arguments → subject agreement marker.

The global ontology organizes the relevant linguistic Concepts into a coherent formal network of classes and relations. Where information in more than one database relates to the same topic, it is the task of the ontology to establish a valid conceptual structure that will enable the integration of diverse conceptualizations. Where multiple senses occur, the TDS-GO also needs to maintain and present coherently the sense differences of component databases. Thus the TDS-GO takes a neutral standpoint towards the analyses represented by the component databases of the TDS. That is, we do not consider it the job of the TDS Project to choose between alternative analyses, but rather to provide access to information through cross-database querying. Therefore the goal of the TDS-GO is not to provide *one* system

---

TDS-GO with GOLD to the extent that this is feasible, by defining correspondences and removing any gratuitous incompatibilities.

of linguistic concepts that is adopted as canonical, but rather to express the systems of all the perspectives represented in the TDS. We therefore describe the TDS-GO as an *inclusive* ontology of linguistic concepts.

The TDS-GO is built using one of the industry-standard languages, Web Ontology Language (OWL). The choice of OWL was motivated by (among other things) the requirement for extensibility, ease of integration with other components of our XML-based system, web-based user interface querying and the availability of development tools.

5.3.1 Conceptual principles underlying ontology development

The TDS-GO is based on a set of underlying principles which govern the process of establishing Concepts and their relationships. As discussed in Saulwick et al. (2005), our methodology follows current recommendations for ontology building (Gruber 1993; Gómez-Pérez et al. 2004), namely: *clarity*, *coherence*, *extendibility* [sic], *minimal encoding bias*, *minimal ontological commitment*, *representation of disjoint and exhaustive knowledge*, *minimization of syntactic difference in encoding* and *standardization in naming conventions*. Important features of ontology-driven integration are the use of shared vocabulary in a coherent and consistent manner (Gruber and Olsen 1994) and where possible the standardization of naming conventions. In the following paragraphs we will discuss the conceptual principles guiding TDS ontology development.

A bottom-up approach

A fundamental design principle of the TDS-GO concerns the basis for the postulation and establishment of Concepts (i.e., classes, properties or individuals). It is a design and methodological principle of the TDS that ontological Concepts are only established on the basis of information *existing* in component databases, thereby constraining the global ontology to a range of relevant areas. This is motivated by the desire to ground the ontology in empirical data-based theory, and thus it acts as a limiting device on otherwise unconstrained ontology growth. However, a Concept *may* be established for which there is no database mapping, if it is syntagmatically or paradigmatically relevant. For instance, at one point the TDS-GO Concept *Transitive Object* (the second argument of a transitive verb) was not linked to any database fields, but it was included in the ontology alongside the Concepts *Intransitive Argument* (the sole argument of an intransitive

verb) and *Monotransitive Argument* (the first argument of a transitive verb), which did relate to data in component databases. In other words, a Concept may be established if it fills a paradigmatic or syntagmatic gap in the network thematic domain. Later, this Concept was linked to new database fields; in this way, its prior addition to the ontology as part of a paradigm facilitated the integration and linking of new data in a globally consistent way.

Prototypes

As is well known from prototype theory (Rosch and Lloyd 1978; Taylor 1989; Varela et al. 1991), an entity included in a category (also a class of entities subsumed by a superordinate category) may have more or fewer of the features/attributes associated with that category, depending on whether it represents a more or less prototypical exponent of the category. The TDS-GO adopts a prototype approach to the classification of linguistic categories. For instance, the class *Free Pronoun* subsumes the classes *Cardinal Pronoun*, *Demonstrative Pronoun*, *Emphatic Pronoun*, *Personal Pronoun*, *Possessive Pronoun, Reflexive Pronoun,* and *Weak Form Of Person Marker*. Subsumption represents the standardly used "is-a (kind of)" relation, where the subordinate entities represent specializations of the category. It is clear that the entities *Cardinal*, *Demonstrative*, *Emphatic*, *Personal* and *Possessive Pronouns* are each a special type of the superordinate class *free pronoun*. That is, each of these classes has at least one additional feature that is the basis for its specialization. We could label each of these features, respectively, as *+value cardinal, +value demonstrative, +value emphatic* and so on. In terms of classification, one could argue that the class *Weak Form Of Person Marker* is an invalid specialization of the class *Free Pronoun* because it is not necessarily *free* or unbound. Its exponents may be free, cliticized or bound depending on the language. Thus in the strictest sense the class *Weak Form Of Person Marker* is not a specialization of the *Free Pronoun*. However, adopting a prototype analysis allows for a subordinate class (in this case *Weak Form Of Person Marker*) to have features in apparent conflict with the superordinate class if certain core features of the specialized class are consonant with the superordinate category. In this case we could describe some of these as: "deictic marker referencing person referents."[22] By permitting the kind of prototype classi-

[22] The degree to which a weak form is able to encode referential specificity is not at issue here; see Siewierska (2004: 9, 124ff).

fication presented here, a richer and thus more fine-grained network of associations between categories is provided. This results in the possibility of more extensive cross-data mappings and thus facilitates more effective resource discovery.

Theory-neutral perspective

Each of the component databases reflects the theoretical stance of its creator in a myriad of choices, both in the way linguistic phenomena are conceptualized and in the terminology used to describe them. When diverse databases provide information about the same topic or use the same term, there is the potential for mismatch. As already mentioned, the TDS-GO is an *inclusive* ontology of linguistic concepts: it provides a common vocabulary that serves as a non-prescriptive basis for the integration of database-specific categories. The TDS-GO is by design maximally compatible with different conceptualizations of linguistic phenomena. It includes crucial concepts but attempts to refrain from incorporating details peculiar to a particular theoretical orientation. This does not mean that the ontology itself consists of Concepts that are "a-theoretic." Indeed we hold that such a pursuit is unattainable for the simple reason that all terms bear the hallmarks of their particular theoretical orientation. Rather, the global ontology is *inclusive* in the sense that it can accommodate the variety and richness of individual theoretical orientations with all their idiosyncrasies. Where appropriate, variant and potentially conflicting orientations or conceptualizations are included in the global ontology and are unified under broader categories.

The decision whether to include a concept in the global ontology is dependent on how widely accepted a linguistic category is, within or across (conflicting) linguistic theories. A linguistic category that is not included in the global ontology is treated as a concept in a local ontology: it is represented as a DTL Notion (Saulwick et al. 2005).

In this way, the ontology strives to achieve a variation on the principle of Gruber's (1993) minimal ontological commitment, namely *minimal orientation commitment*. This is the inclusion of diverse theoretical orientations, without ascribing to any one a favoured status.

An example of "ontological unification" (Saulwick et al. 2005) is the case of the variant Concepts *Basic Word Order* and *Predicate-Based Word Order*. In the TDS-GO these are unified under the supercategory *Core Constituent Word Order*. We call this *semantic unification*; not an ironing

out or watering down of theoretical orientation, but the establishment of an inclusive superconcept for the purposes of information integration. A query over any one of these Concepts allows the end-user access to the others. In adherence to the principle of clarity (Gómez-Pérez et al. 2004), the TDS will ensure that the intention behind each database contributor's use of terminology is faithfully represented in the local ontologies of the DTL.

5.3.2 Linguistic Concepts

The TDS-GO models a variety of linguistic objects, relationships and other linguistics-related ideas. Ontology Concepts are labeled and described with a short explanation, and possibly references to a bibliography. The TDS-GO thus also serves as a guide to interpreting the terminology of component databases.

In this section we only give a high-level overview of the ontology design. The reader is referred to Saulwick et al. (2005) and Dimitriadis et al. (2005) for more details on the structure and implementation of the TDS-GO.

5.3.2.1 Types of linguistic Concepts

We distinguish between the following major types of linguistic Concepts:

1. **Linguistic objects**

   These can be thought of as existing in themselves. They include Concepts such as *Sentence, Morpheme* and *Phonological Segment,* as well as classes representing *Language* and various groups of languages.

2. **Linguistic properties**

   These are (linguistically salient) properties predicated of a linguistic object. For example, *Basic Word Order* is a property of *Languages,* while *Referential* is a property of certain words or syntactic constituents. In this terminology, properties do not relate one linguistic object to another but can be thought of as one-place predicates. They are generally associated with a set of possible values; for some the values are 'True'/'False' or 'Present'/'Absent', while for others it may be one of several possibilities with linguistic meaning such as a paradigm, as with the property *Case* which can have the values *Accusative/ Ergative/Dative*, etc.

3. **Linguistic relations**

   These model a phenomenon involving two or more linguistic objects or properties. For example, following Corbett (1998:191), *Agreement* is modelled as a relationship involving a *controller* ('the element which determines agreement'), a *target* ('[t]he element whose form is determined by agreement'), a *domain* ('[t]he syntactic environment in which agreement occurs'), and *agreement features* ('in what respect there is agreement'). The participants in a relation play distinguished *roles,* whose names may be particular to each relation: for *Agreement*, the roles are *controller, target,* etc. Complex phenomena that are not explicitly relational are also treated in terms of roles: for example, *Stress Assignment* can be described as involving a *Method* (algorithm) that makes reference to types of feet, edge-sensitivity, extrametrical material, etc.

5.3.2.2 Relationships between linguistic Concepts

Entities in the ontology are organized according to the following major relationship types:

1. **Subsumption**

   Some linguistic Concepts are specializations of others. For instance, 'grammatical case' is subsumed by the more general Concept 'case'.

2. **Loose synonymy**

   This designates variant linguistic terminology used to refer to the same phenomenon. When two phrases denote the same conceptualization of a phenomenon, it is useful to link them in order to provide a means of searching using different vocabulary than that used for naming the ontology Concepts. (Loose) synonymy between two phrases is currently implemented as an annotation on the class (essentially, a data property that gets a value for the entire class). The Concept *Agreement Marker* is for example annotated with the alias *Person Inflection*.

3. **Related phenomena**

   This identifies variant linguistic terminology used to refer to similar or related phenomena. For instance, the Concept *Basic Word Order* has this relationship to *Predicate-Based Word Order*. Although the two phrases denote somewhat different conceptualizations of phenomena, it is useful to link them in order to provide a means of unified searching

across both component databases in which the terms occur. As neither of the current standard annotations *owl:sameAs* and *owl:equivalentClass* (Bechhofer et al. 2004) captures our required semantic correspondence, we use our own annotation, *tds:equatesWith,* to equate two related Concepts.

4. Meronymy

This stipulates part–whole relations.[23] Some linguistic concepts are modelled in a strict hierarchical structure. For example, *mora* > *syllable* > *foot* > ... form a *meronymic hierarchy*. Note that their relationship to each other cannot be expressed through subsumption; a syllable, for example, is not *a kind of* foot, but part of one. Certain linguistic hierarchies are organized so that units of one type are a *direct part* of the next higher unit, e.g., in the "prosodic hierarchy" (Nespor and Vogel 1986), which is a hierarchy of utterance constituents from a prosodic perspective in which "[e]very prosodic category in the hierarchy has as its head an element of the next-lower level category" (Kager 1999:146). The direct-part-of relation is a specialization of the general part–whole relation.[24] We encode part-whole relationships via a meronymic predicate, *isDirectPartOf* (and its transitive closure, *isPartOf*). This relation is asserted between pairs of classes, and serves to organize them into meronymic hierarchies.

5. Determination

We use this appellation when a linguistic property is defined in terms of one or more other linguistic properties. For example, if a heavy syllable is defined as a syllable with a long vowel or a coda, then both of these are determinants of the property *Syllable Weight* (even though the

[23] A variety of meronymic relationships may be required. For instance, Story (1993) based on Landis et al. (1987); Winston et al. (1987) and Chaffin et al. (1988) lists seven types of meronymic relations: component–object, member–collection, portion–mass, stuff–object, phase–activity, place–area and feature–event.

[24] Our implementation follows the current W3C recommendation, which calls for expressing meronymic relationships in terms of a direct part relation when "what is needed is not a list of all parts but rather a list of the next level breakdown of parts, the 'direct parts' of a given entity" (Rector and Welty 2005).

presence of only one is enough to make a syllable heavy).[25] The determinant relation is a logical (not linguistic) relationship between linguistic properties or relations.

6. **Form-function relationship**

   Here "function" is used in a specific sense. It refers to the linguistic function served by some linguistic entity. This relationship associates entities of the type *linguistic object* with linguistic properties expressing their possible linguistic function. For example, the Concept *Agreement Marker* is a possible function of *Affix.* A form-function relationship is a (type of) linguistic relation, and it is implemented accordingly (i.e., as an OWL Class with roles expressed as OWL Properties).

5.4 Topic taxonomies

The global ontology as described has a formal structure, which is needed to enable "smart search" facilities. However, this does not make it a natural entry point to the data collection for end-users. The placement of Concepts in the global ontology, while formally correct, will not always be intuitive. To minimize possible confusion, the global ontology is currently largely invisible in the TDS user interface.[26] Its primary function is in the back end of the system, where it facilitates the coherent unification and integration of diverse linguistic concepts.

The process of local and global ontology creation results in fixed Notion and Concept hierarchies, each of which reflects one of many possible organizations of the (meta)data. By introducing topic taxonomies we allow multiple organizations of metadata for different types of uses. A topic taxonomy is a relatively small list of domain-specific terms organized into a loose hierarchy. They are designed to capture the perspective of different

---

[25] Determination only holds when the definition of a concept involves aspects of another. It should not be confused with empirically based implicational relationships. In the latter case we have two concepts which are independently defined, and the implicational relationship is an empirical (contingent) fact rather than part of their meaning.

[26] Concept definitions are shown to the end user when they (partially) match a full text search, or in relationship to a Notion, allowing the user to fine-tune the search or disambiguate the meaning of the Notion.

sub-domains of the linguistic space. The data of component databases is not directly associated to topics in the taxonomies, for the reason that as the collection of component databases grows over time, it would be impractical to maintain links to multiple topic taxonomies. Instead, the global ontology serves as a common frame of reference at the nexus between metadata and topic taxonomy. Topics in the taxonomies and Notions in the component databases are both linked to Concepts in the global ontology, allowing taxonomy topics to be indirectly related to database fields. (The indirectness of the link is not apparent to the users, who only see a list of fields and grouping Notions associated with each taxonomy topic).

The default taxonomy provides a complete overview of topics currently covered by the component databases. Its initial structure was based on the table of contents of Thomas Edward Payne's book *Describing Morphosyntax: a guide for field linguists*, but has been extended with more topics to cover the entire domain of topics in the TDS. In this taxonomy the topic "Complement clauses," a daughter of the topic "Clause combinations, coordination," is linked to the Concept "Syntactic complement" in the global ontology. This Concept has direct relationships with nine Notions in the DTL specifications, e.g., the Notions *tdn:Form_of_the_complements* and *tdn:ComplementOfCopula*. When we take the semantic context of these Notions into account, the topic "Complement clauses" is directly or indirectly related to 63 Notions. The Notions that are directly related will be ranked highest when presented to the user.

No domain-specific taxonomies have been created to date, but it is envisaged that some may be created in the future, e.g., one limited to the domain of phonology.

6 TDS implementation

The TDS architecture shown in Figure 1 is close to a semantic web application, although it is an application-specific web.[27] This made it possible for some core technologies used in the metadata network to follow W3C recommendations or working drafts. Topic taxonomies make use of the

[27] The TDS web interface is of course part of the World Wide Web, and some descriptions of Notions, Concepts and topics refer to web pages on other sites. However, the semantic knowledge base isn't open and thus is not part of the semantic web as envisioned by W3C.

SKOS vocabulary, and the global ontology is defined in OWL. A benefit of using these standards is that externally developed tools are available to aid in managing the corresponding components, e.g., the TDS-GO is edited with the Protégé ontology editor with a plug-in for OWL.

The DTL was developed in-house, as no suitable specification language or toolkit was found. Early versions of the TDS used XSLT as the transformation language, with the result that creating specifications required extensive knowledge of the low level implementation details of the TDS. The DTL was designed with the goal of allowing the knowledge engineer, a linguist, to create such specifications. In addition, it supports the addition of enriched metadata about the database semantics. The DTL is a declarative *domain-specific language* for transformation rules, which can be annotated with metadata that express the semantics of the resulting data (schema). The *DTL Engine,* the interpreter that carries out the database integration according to the DTL specifications, is implemented in Java.

One of the first products of the TDS implementation phase was a tool to import data from various database formats, called *TDS Localize*. This tool is implemented in C, and uses a plug-in architecture to allow the dynamic addition of loaders for new DBMSs. The current set of plug-ins provides access to databases using the following interfaces: ODBC (e.g., MySQL, Microsoft SQL server), ODBTP (e.g., Microsoft Access) and CSV exports (e.g., SPSS, Mircrosoft Excel). The DTL engine relies on this tool to import most of the databases. By now some C plug-ins (e.g., for CSV and XML files) have been replaced by Java equivalents, which are loaded directly by the DTL engine; but TDS Localize is still needed because C libraries are still the only way to access some major DBMSs (e.g., Microsoft Access is only accessible via ODBTP).[28]

The result of the import, transform, merge and enrich process, as implemented in the DTL engine and specified by a set of database specific DTL specifications, is a large XML document containing the instantiated, and interlinked, Notion hierarchies. XML is well suited for this kind of semi-structured data, (semi-structured because the resulting data structures may overlap only partially, allowing very sparsely filled data structures). The

[28] A complete switch to Java would allow the complete TDS framework to be deployed on any platform that Java runs on. TDS Localize is currently the only tool that binds the TDS to the UNIX family of operating systems.

use of XML also enables us to draw on the wealth of technology standards and tools which have been produced since its birth out of SGML in 1998, e.g., XSLT and XQuery.

The TDS web interface uses two additional modules. The front end is powered by Backbase (http://www.backbase.com/), a Rich Internet Application (RIA) library, which uses JavaScript to extend the browser with a set of sophisticated GUI widgets allowing the creation of web applications which resemble desktop applications in functionality. The server back-end uses the 1060 NetKernel application server (http://www.1060.org/), which incorporates a very rich set of tools to handle XML. The most prominent tool is Saxon (http://www.saxonica.com), the leading open source XSLT and XQuery engine.[29] This engine is used to execute the queries and to style the query interface and results.

7 The user interface

The TDS webserver must provide the functionality of a specialized interactive application while running on an ordinary web browser. This is accomplished with the help of Backbase, a sophisticated library of JavaScript routines that extend the web browser's built-in capabilities. JavaScript is executed by the user's web browser, which must therefore be of sufficiently recent vintage, and must have JavaScript enabled. The Backbase library provides user-interface enhancements such as pop-up

[29] Saxon is not the ideal tool for the purposes of the TDS. At the start of the TDS implementation phase, Saxon was the only reasonably up-to-date XQuery processor. However, Saxon is document-oriented, which means that in principle it has to reread the source XML document, the TDS data, for each query. The 1060 NetKernel helps by caching the parsed XML document, but the resource consumption is still quite high as the document has to be fully loaded into memory (which takes on average 5 times the size of the document on disc!)

Since XQuery gained W3C recommendation status, more and more standards-compliant XML database engines are becoming available, and could replace Saxon. An XML database engine would, among other optimizations, take care of loading only XML document "hot spots" into memory; hence it is expected that resource consumption will be more modest, and response times will drop to a fraction of current levels. At the time of writing, however, the TDS implementation still uses Saxon as the XQuery engine, and has not been ported to an XML database engine.

messages, "tabbed" sub-pages, and a rich system of windows and menus, all managed within the browser window. The library makes it possible for many interactive operations to be performed locally on the user's computer, avoiding a time-consuming request to the TDS webserver at each step.[30]

The opening page of the TDS server provides the starting points for using the system, as well as links to the expected information and support pages. This includes documentation about the Project, a tutorial walk-through that introduces new users to the basics of using the system, descriptions of the databases included, and technical features and limitations of the TDS server.

As mentioned in section 4.1, querying the TDS is a two-step process: The *pre-query* involves selecting database fields of interest and adding them to the query basket. The user then opens the query basket and specifies query options and search terms for the actual query. The query can then be submitted to the server, and the results are returned and displayed.

The goal of the pre-query stage is to find database fields whose contents are relevant to the user's goals. The TDS provides ways to *search* or *browse* through the descriptions of fields. In the Search tab, the user types search terms in a form and is shown a list of database fields with matching metadata. Suppose, for example, that a user wants to look up whether Swahili is listed as a null-subject language. They can look for relevant database fields by typing "null subject" in the search field of the Search tab, which will reveal (alongside a large number of other partial matches) a field named *pro-drop*. The documentation displayed with this field indicates that it comes from the TDN database, and that it is indeed relevant to the task at hand. It should be noted that the search terms are not only matched against the text of the field names and descriptions, but also against the possible values of each field and the global ontology Concepts to which the field and values are linked. (Matching Concepts are also shown in the list of

[30] The extra functionality comes at some cost. "Rich internet application" libraries are still relatively unstable technology, due especially to browser inter-compatibility limitations, and place heavy processing demands on the user's browser. The TDS server is only compatible with relatively recent, full-featured versions of Internet Explorer and Firefox; the TDS interface takes 10-20 seconds to start up, or even more on slow computers, because of the complexity of the JavaScript library; and the TDS interface is poorly integrated with the browser's Back button (it is recommended that its use should be avoided).

results). In this way the global ontology of linguistic Concepts, and the links between them, serve as a system of structured keywords that guide searches. For example, the database field *pro-drop* is linked to the Concept *NullSubject.* Searching for "null subject" will match the field *pro-drop* via this indirect link, even though the term does not appear in the field's description. Once a field of interest is found, a user may add it to the "query basket," a sort of shopping cart for database fields.

As an alternative to typing in search terms, a user can browse the hierarchy of topics and data fields in the TDS (i.e., the taxonomy topics and local ontology Notions). The TDS interface provides several alternative hierarchies for navigation. The "view by datatype" tab matches the hierarchy of entities and thematic groupings in the TDS data. At the top of the hierarchy are the objects described: a language treated as a whole, a unit of text (usually a glossed sentence with additional annotations), or an external entity that exists independently of any particular language, such as bibliographic sources or the universal phoneme inventory.[31]

Most component databases of the TDS consist of "analytical" parameters, which describe a language as a whole. Hence, most information is found under the heading *Language.* This contains language identification information (name, ISO code, location and genetic affiliation), and a large group of properties labeled *Linguistic Phenomena*, which is where information about linguistic systems and properties is concentrated. Users of the TDS will most often search under this node.

The second view through which the user can browse for data fields is organized by topic. Here the linguistic topics covered by the component databases of the TDS have been organized into a shallow hierarchy, or *taxonomy*. It is the "table of contents" of the TDS, much as the table of contents of a linguistics textbook provides an overview of the topics it addresses.[32] Because the description of a database field often involves reference to several linguistic Concepts, a database field may appear under several topics in the hierarchy. The TDS currently provides two different topic taxonomies; it is planned that additional taxonomies may be added,

---

[31] In a relational database, each of these would correspond to a separate table. The TDS equivalent is a so-called *Root Notion.*

[32] The topic hierarchy is in fact modelled on the table of contents of Payne (1997), with the necessary adaptations.

customized to particular kinds of uses or subfields of linguistics (e.g., for phonology-related topics).

Once the user has added some fields to the query basket, they can proceed to the second stage of the search procedure, the formulation of the query itself. The query basket is viewed in a separate window, which lists each collected field or group along with its description and controls for defining the search query. To this end, the user must decide on (i) what selection criteria to specify (e.g., "the field *pro-drop* must have the value *True*"), and (ii) which fields to display in the results.[33] If one wants to know the names and language families of languages that have pro-drop, they would specify "pro-drop = True" as the selection criterion and include *Language name* and *Genetic affiliation* for the fields to display.

Only fields in the query basket can be used for this process. If the user wants additional fields, they must resume the pre-query process and add the required fields to the basket.

The method of specifying selection criteria depends on the type of the data field. For fields with "enumerated" values, which come from a fixed set of choices, the query window displays a control that allows for the selection of one or more desired values. For text fields, users can enter text to search for, with a choice between exact and partial (substring) match. For example, one can search for all phonetic segments whose description includes the words *unrounded vowel*. Text fields, and searches, may use any Unicode character.[34]

---

[33] It is possible to specify selection criteria for a field but suppress its display; for example, if one only selects languages that *have* pro-drop, there is no need to see the value "pro-drop = True" for each language shown. If multiple values can match the selection criteria (e.g., "Basic Word Order = SVO *or* VSO"), it is of course necessary to display the relevant field if its value is of interest.

[34] A small icon provides a shortcut to the TDS IPA Console, an external application that can be used to paste special characters (especially IPA characters) into form fields. The TDS IPA Console is a java application developed as a stand-alone tool by the TDS Project. It was developed to simplify the entry of IPA characters in phonological query fields, but it can be used with any other program that can import (paste) text from the system clipboard: text editors, spreadsheets, web browsers, etc. The Console comes with a large number of predefined buttons for entering IPA characters, arranged on several tabs in the familiar IPA table layout;

Once the query has been defined in this fashion, it can be submitted to the system. The result appears in a new window. (Multiple query views can be open simultaneously, and each is automatically assigned a number. They remain open when the query is submitted and can be resumed, modified and resubmitted.[35])

The results are displayed in table format by default, but the TDS supports several alternatives. At the top of the results window is a link (labeled *result settings*) that allows switching between display modes. The "report" format displays each field and value on a separate row, and is useful when very many data fields have been selected for display, or with long text fields. The "summary" style presents counts for the different values (e.g., number of languages in the TDS listed as having each word order), and constitutes the only kind of data statistics currently provided by the TDS. There are also two export formats, XML and comma-separated values (CSV); these are useful for exporting the results to another database application or spreadsheet.

The TDS system, as can be seen from the presentation of its features, is oriented towards discovering and viewing data in the component databases. Because of the mixed nature of its content, the set of languages found in the TDS do not constitute an areally or genetically balanced sample. While balanced sub-samples can be defined among the great number of languages present in the aggregated TDS data, such collections will not have values for all possible typological parameters (database fields); thus, a balanced sample can only be defined on a per-query basis, or for one component database at a time. This feature is not currently supported by the TDS; because of the additional uncertainties involved in aggregating diverse databases, we advise against attributing statistical significance to the results retrieved from the system.

---

users can define additional keys bound to any Unicode character (which can be selected from a list or specified by its "Unicode number").

[35] At the time of writing, support for modifying a query window is limited. Selection and projection criteria can be changed, but new fields cannot be added to an open query view window. Instead, new fields must be added to the query basket and a new query window must then be opened.

8 Working with metadata

The TDS framework for data integration allows a lot of flexibility in carrying out the integration process. Just how this should be done depends on the particular database being integrated, on the general principles that the TDS team has adopted, and not infrequently on the subtleties of relevant linguistic theory.

For each database included in the TDS system, a DTL schema is written that defines the transformation of the source data into the unified data schema of the TDS. After a careful examination of the original database, and usually following some discussion with the database creators, members of the TDS team create an initial schema file that imports the data from the database into "field Notions," i.e., nodes in the DTL hierarchy representing a database field, essentially with no changes. (As already described, the DTL schema includes a reference to a plug-in that establishes access to the database). The TDS knowledge engineer then embarks on reorganizing the data fields of the component database so that they match the hierarchy and encoding conventions of the TDS system, and on entering documentation for all Notions in the resulting schema.

The simplest kind of alteration (conceptually at least) is the recoding of values for common types. For example, Boolean (true/false) fields are represented with the standard values "True" and "False" in the TDS. If a field uses the values "+" and "-", "yes" and "no", or "1" and "0", these will be mapped to the TDS standard.

Another kind of recoding involves various kinds of cleaning up the values found in a database. Component databases often use text fields for parameters that logically allow only a restricted set of values (true/false parameters, word orders such as "SVO", "SOV", etc.). This opens the door to inadvertent misspellings, typos or other irregularities in the content of such fields, which must be resolved if they are to be correctly matched to query selection conditions. DTL specifications often include ad hoc rules to address such problems.

Sometimes the database creators append comments to the logical value of the field, most frequently question marks or other annotations indicating uncertainty about the correctness of the value. Such annotations constitute important information for the end user; they are separated from the value itself and recorded as an "uncertainty marker," which is automatically displayed whenever the associated value is displayed in the results of some

query. (Cf. section 9 for further discussion of uncertainty annotations).

Other fields are combined or split up to yield more consistently organized Notions. A database will sometimes use a different data field for each possible value of a linguistic property; e.g., one field for the basic word order SVO, another field for SOV, etc. Such decomposition into "characters" has its uses, but it does not meet the design guidelines of the TDS; logical fields are reconstituted, so to speak, by mapping groups of such database fields into a single Notion.

In some cases, a new Notion is computed in more elaborate ways from one or more database fields. The field Notion "pro-drop" (whether a language allows main clauses lacking an overt subject) is computed from its logical complement, a field in the TDN database that records whether a language always requires overt subjects in main clauses. A more complicated example, involving the computation of the property *Trochaic language* from a combination of other properties, is described in the next section.

A major part of harmonization is the assignment of Notions to the proper place in the TDS hierarchy. Database fields whose semantics are sufficiently similar to those of other databases (e.g., *language name*) are mapped to Notions with global scope. Such Notions are typically already defined elsewhere, in the master DTL specification; the schema for the component database instantiates them with data from the database. (If a Notion belongs in the global scope but is not already defined, it can be added to the master DTL schema at this point). Most Notions, however, are defined in a way that is specific to their database of origin and will receive local scope. They are grouped and organized hierarchically to represent the conceptual relations between the various fields in the database. Hierarchies of local Notions are embedded in the shared hierarchy; for example, the master DTL file defines the Notion *tds:Phenomena* (containing all linguistic phenomena) and a daughter called *tds:BasicWordOrder*, representing all information on word order from all databases. Databases containing word order information will define Notions with local scope within this subhierarchy, as appropriate. If some Notions defined with local scope are later found to be shared by several databases, they can be transferred to global scope so that they can be shared as needed.

In addition to situating and instantiating field Notions properly, the TDS knowledge specialists must look after their descriptive metadata. These consist of descriptive documentation entered directly into the DTL and associated with Notions at any grouping level, and of links to Concepts in

the global ontology. Appropriate links facilitate discovery of these Notions through the search interface, and bridge the gap between database-specific (local) semantics and global semantics.

Finally, an important part of the data import process is the harmonization of key values. Almost all typological data describes individual languages; merging data from different databases relies on knowing which language is being described. The TDS accomplishes this by basing the primary key for each language on its ISO code (SIL code). Ideally a component database will provide ISO codes for every included language; in practice there are plenty of omissions, differences between versions of the Ethnologue, and other complications that are resolved by the knowledge engineers on a case by case basis.

Similar problems arose in the harmonization of phoneme inventories; this case is described below (section 8.2). First, we turn to a simple example of constructing a TDS Notion that does not correspond to a single database field.

8.1 Finding "trochaic languages"

Stresstyp contains information on the foot type that is used in the derivation of stress systems in various places. Feet are used to derive the locations of primary and secondary stress (the separation being motivated by arguments not discussed here, but see Goedemans and Van der Hulst, this volume). As is well known, feet come in two flavours, iambic (right headed) and trochaic (left headed). In the view of StressTyp, a language is said to be a *trochaic language* if any of the following is true:

1. the default foot used to place stress in the primary stress domain (in every case in quantity-insensitive languages, and in case both syllables are light in the quantity-sensitive languages) is trochaic.
2. The foot used to derive the location of rhythm beats (secondary stress) is trochaic.
3. Both iambic and trochaic feet are used in the derivation of rhythm.

StressTyp does not directly record whether a language is trochaic, but the property is easily calculated. To find a list of languages with these parameters from within the StressTyp database, we need to execute the SQL query shown in Example 4.

```
SELECT Name
FROM   Stress
WHERE  Rhythm_Type="trochaic“
    OR Rhythm_Type="both“
    OR Stress_l_l="trochaic";
```

Example 4 *Trochaic feet SQL query*

For the TDS, it was decided that this property should be directly searchable in the system. Accordingly, a local Notion was defined, and its value is computed from the relevant StressTyp fields as appropriate (Example 5). The result is an ordinary field Notion that can be used in queries like any other field. A query for trochaic languages returns a list of 160 matching languages.

```
NOTION stressTyp:trochaicFeet
  LABEL "Trochaic language"
  DESCRIPTION "Trochaic feet are used in the analysis of the stress pattern
               observed in this language"
  LINK TO CONCEPT trochaicProperty
  IS (
       FIELD Rhythm_Type="both“
    OR FIELD Rhythm_Type="trochaic“
    OR FIELD Stress_l_l="trochaic“
  );
```

Example 5 *Trochaic feet DTL Notion*

8.2 Harmonizing phoneme inventories

The core source of information on phoneme inventories in the TDS is Ian Maddieson's UCLA Phonological Segment Inventory Database, or UPSID (Maddieson 1984; Ladefoged and Maddieson 1996). This database contains a set of 920 segments, from which individual languages select a subset to form their specific phoneme inventory. The segments in the original UPSID database are represented in a highly idiosyncratic, SAMPA-like encoding. This means that the identity of a certain segment is not transparent to users unless they are thoroughly versed in the UPSID encoding system.

The TDS Project has adopted the international standard IPA representations for phoneme representation. This was not possible when UPSID was originally created, but is relatively simple nowadays, since there are Unicode fonts in common use that contain the (majority of) necessary IPA charac-

	Bilabial		Labiodental		Dental	
	Voiceless	Voiced	Voiceless	Voiced	Voiceless	Voiced
Plosive	p	b		b̪	t̪	d̪
Plosive, aspirated	p^h				$t̪^h$	
Plosive, preaspirated	^{h}p					
Plosive, palatalized	p^j	b^j			$t̪^j$	$d̪^j$
Plosive, labialized	p^w	b^w				

Table 1 *The initial five rows and six columns of the Universal Phoneme Position Chart*

ters. Therefore, the first stage of the harmonization of UPSID was to transliterate the 920 SAMPA-encoded segments into IPA.

This achieved the dual purpose of upgrading to the more user-friendly and standard IPA, and enabling integration of UPSID with other databases containing information about phonemes. Thus a searchable list was created, encoded in Unicode IPA, together with a phonetic description of their articulatory and/or acoustic properties. This list is a reasonably comprehensive (but certainly incomplete) collection of the possible phonemes in human language, and might be of some interest in itself. But the main motivation in creating it was to support IPA notation in querying and displaying the phoneme inventories of individual languages, or groups of languages.

Representing the results of such queries on screen in a user-friendly fashion also proved to be a challenge. A long list of consonants and vowels is much less informative than a neat table, comparable to those found in grammars and IPA charts, where consonants and vowels are ordered according to their place and manner of articulation. This information is available in UPSID, along with all other segmental features. In order to be able to represent phoneme inventories on screen, we created the Universal Phoneme Position Chart (UPPC).

For consonants, the UPPC is in essence a single table of 30 columns and 171 rows, in which all segments are represented according to their manner and place attributes. The treatment of vowels is similar. The large number of rows results from the extensive number of secondary articulations that are potentially distinctive in individual languages. An excerpt from the upper-left corner of the UPPC is presented in Table 1.

With the aid of the UPPC, TDS query results involving phonemic inventory data can be presented on screen in table format. To generate an inventory of a single language, empty columns and rows are automatically removed, resulting in a table of manageable proportions. As an extra feature, we have added the possibility to render tables from queries over multiple languages. Such queries result in aggregated "phoneme inventories" presenting the phonemes of all languages in the query. This is supplemented by numbers indicating their frequency, with coloured backgrounds to reveal "hot spots" of common or very common phonemes.

8.3 Expanding the coverage

Starting with UPSID's extensive segment pool, we hoped to be able to incorporate other databases containing information on segment inventories relatively easily. The IPA representations, which are created from templates with fixed orders for multiple co-articulation modifiers to the central symbol, act as the database keys to which we can link phonemes from other databases. Our expectation was that UPPC would require little supplementation, beyond the 920 values from UPSID, to support other databases with phonemic information. But our experience with the incorporation of the SPIN database proved otherwise.

SPIN contains phoneme inventories for 110 languages. It is unique among the TDS databases in that it did not enter the Project as an electronic database; the data was available to us only on handwritten index cards, and was digitized by TDS participants.[36] Unicode IPA characters were used throughout, to ensure compatibility with the UPPC and the rest of the TDS. Once the data was imported, we matched the complete set of phonemes from SPIN to the 920 potential phonemes in the UPPC, expecting to have to iron out only a moderate amount of human errors. Unfortunately, the

[36] The data were entered in Excel worksheets and imported into the TDS using a custom conversion script.

"ironing" required proved to be quite substantial. There were 418 discrepancies between SPIN and the UPPC, attributable to six types of mismatches:

1. Homographic errors: incorrect symbols

A unique phoneme should be represented by a unique Unicode character, but in practice this is not always the case. Especially when differences between characters are not clearly visible in certain fonts, errors are easily made when entering data. When using a font in which /g/ looks exactly like /ɡ/, for instance, the former may be selected to represent the voiced velar plosive although the latter, a character with a different Unicode number, is the official IPA symbol. Errors of this type are easily corrected through visual inspection of the mismatches.

2. Variant grapheme mismatches

In some cases two official notations for the same phoneme exist, as with the case of velarization, which can be represented by either ɖ or dˠ for the voiced alveolar plosive. If two databases happen to choose different options, we get a mismatch that is not apparent but will lead to incorrect behaviour in queries, since the same phoneme is intended. We resolve this type of mismatch by permitting either grapheme as a valid representation of the phoneme and introducing a relationship which equates the two possible representations in the data representation language. Note that when more than one linguist fills a database, one may even get database-internal inconsistencies of this type, in which identical co-articulations for two different phonemes are represented differently.

3. Notational mismatches

Notation differences in older sources, or sources with different notational conventions. This type is similar to the previous one. Labialized segments, for instance, are normally represented with superscript /ʷ/. In some sources one may find a convention in which labialization is represented by a normal /w/. We can only correct this by looking at the source from which the inventory was taken to determine exactly what articulation was originally intended. If that information is available, we can represent the phoneme using the current IPA notation. Some times, however, an uncommon phoneme is intentionally represented using non-standard orthography; care

must be taken here not to equate such phonemes to a common phoneme that is already present in the UPPC.

4. Grapheme order mismatches

Multiple co-articulations may be represented in different order. A labialized, aspirated /k/ can be represented as /k^{wh}/ or /k^{hw}/. There are conventions, however, for the concatenation of diacritics, and we adhere to these, e.g., using /k^{wh}/ and revising alternate representations.

5. Diacritic order mismatches

Similar, thought not visibly so, are those phonemes written with multiple diacritics whose sequence is unordered. One diacritic may be placed above the segment, for instance, and one below it. These may be entered in either order. The result will be the same graphically, but the sequence of Unicode characters is different. This problem is easily solved by normalizing the Unicode representations,[37] or by visually examining the entries and manually imposing a fixed ordering scheme.

6. Input errors

Simple input errors are easily introduced. For instance, an extra diacritic may have unintentionally been typed after some segment. This may be poorly visible or invisible on the screen, but will lead to a different Unicode representation than the one intended. Moreover, diacritics may have been chosen that are simply incorrect, or errors may have been introduced when working from not very legible (especially handwritten) originals. These errors are usually easy to fix after reference to the original database.

After we corrected the mismatches of these types, we were left with 190 segments that are new and quite a few that needed checking in the original grammars. For a few languages, we consulted the original reference grammar and established that an unmatched segment should in fact be replaced by one already in the UPPC. The number of remaining new segments seems rather high to us, but since the purpose of the TDS is to integrate existing databases rather than to create new resources, we do not put further effort in resolving the differences between the datasets. We simply expand

[37] The Unicode standard defines several normalized forms, which include a canonical order for diacritics. Unicode-enabled software libraries provide functions that normalize Unicode strings.

the pool of possible segments with those phonemes from SPIN that were not eliminated during the procedure described above.

9 Guidelines for typological databases

During the process of importing the eleven databases currently in the TDS, several recurring problems were recognized in the metadata and data cleaning process. This section contains some guidelines for developers of typological databases which would improve their reusability, both as stand-alone resources and through incorporation into the TDS or a similar system.

1. Good database design

Choosing the appropriate relational design for a database (that is, the structure of tables and relationships that make up the database) can make a big difference to its usability, as well as to the ease of reusing its contents in the TDS or any other system. A properly designed database is easy to enter data into and extract information from; it is also easier to modify as one's research design evolves. We recommend making the effort to choose a suitable design, by finding and consulting knowledgeable colleagues if necessary, at the initial stages of constructing a new database.[38] Starting with the right design has such immediate and noticeable benefits for the database creators that its usefulness for eventual data integration is almost incidental.

2. Documenting assumptions and procedures

Typological databases have a long lifespan. Most of the databases currently incorporated in the TDS have existed for over a decade, even decades. During such long periods of time it is easy for the assumptions and procedures employed when defining fields and adding languages to the database to be

[38] The simplest typological databases consist of a single large table, with one record (row) per language and one column for each property being described. If a phenomenon could be described multiple times per language (e.g., information about each focus marker or syllable structure that occurs), these ought to occupy a separate table in a many-to-one relationship to the language table. Also examples, bibliographic citations, etc., should generally be in separate tables. The needs of each database differ, so the right design depends on the information being collected and the needs of the project. For an introduction to database design for linguistics, see Dimitriadis and Musgrave (this volume).

lost. Sometimes this may lead to inconsistencies. By documenting the database it becomes possible to refresh the project's collective memory, to keep data clean and consistent, and also to provide rich metadata if the database is eventually made public or is reused, for example by incorporating into the TDS. Some database systems provide a way to store documentation of the database's fields, but such support is typically very limited. It is often more convenient to maintain documentation in a separate document, or in a special table within the database (special effort must be made to keep these up to date, however).

3. Citations to the sources of information

Some typological databases, especially older ones, do not include bibliographic or other source information. Such information is essential for error-checking or further research into the information given in the database; even in cases where there is only one well-known reference grammar for a given language, in which case one might consider a citation of it to be redundant, this fact is only known to people who are already involved in study of this language. At a minimum, a typological database should list the source, or sources, of information for each language. Such information only needs to be entered once per source, and can be re-used for future research involving the language. Sources should be given in a separate table, allowing the same source to be easily referenced for multiple languages, or other records, where appropriate (and perhaps to be copied to a new database). Ideally, each group of information in the database would include a separate citation, consisting of a source plus the relevant page numbers; but this may be impractical if it will result in a very large number of citation fields. Simply listing sources for each language in the database is an extremely useful and workable compromise.

4. Key values

Record keys identify each record to the database. We recommend that you look for standards to take your key values from. Even if your database internally uses numeric keys or other ad hoc values, providing standard identifiers makes it easy to detect inadvertent duplications and is essential for the reusability (i.e., continued life) of the data. In the case of languages, which are always the core entity in a typological database, the standard to use is the most current ISO 639-3 specification (http://www.sil.org/iso639-3/), which contains unique language codes for living and extinct languages. The ISO 639-3 codes are the successor to the "SIL codes" used in the past

by the Ethnologue language directory. It can be difficult to fill in this information after the fact; often, the ISO code corresponding to a language cannot be determined without referring to the original reference grammar. The TDS is replete with data about languages whose identity (i.e., ISO code) could not be determined with certainty by the TDS analysts. ISO codes should always be determined (and recorded) when data entry for a language commences. The introductory section of most reference grammars provides sufficient information to identify the language in the Ethnologue.

What if the language variety being described is a dialect that differs in significant respects from the standard, or canonical variety, of the language listed in the ISO 639-3 directory? In this case the ISO code for the language should be provided, but the record must be distinguished from one that would describe the standard variety. This must include the assignment of a special key for the dialect; again, we recommend adhering to a systematic means of identification if possible. The Ethnologue contains additional information on language families and dialects which can be used to standardize keys. Only fall back on an internal key when a standard one cannot be found, since lack of a database-independent key will greatly complicate the integration of data for that specific language with data from other databases.[39] Non-standardized keys should be clearly distinguished from ISO codes or other standard keys (e.g., by beginning with a prefix such as *dialect-*).

5. Comments

Many databases contain a field with arbitrary notes or comments for any aspect of the record. Such general-purpose fields are difficult for the TDS to handle, since data is extracted and presented one field at a time. It is far better to have separate comment fields for each value or group of values that requires them (this is also more effective for the original database). At the least, a comment should name the data fields that it is relevant to.

---

[39] If the ISO codes are used as the primary key for a record, internal codes are necessary to avoid null keys; but if the ISO codes are given in a non-key field, only valid ISO codes should be provided; a three-letter code that is not part of some standard is worse than useless, since it might be misinterpreted during a future data integration.

Another common practice is to use "a comments field" for any information that the database creators decided to collect after the database was already designed, or that they cannot decide how to encode. Such practices are sometimes unavoidable, especially if information must be collected before a decision on its encoding can be made; but they should be replaced as soon as possible by dedicated fields. A comments field is only useable by humans inspecting the record, and cannot be properly utilized in database queries.

6. NULL values

NULL values are controversial in the database community because their semantics are not well-defined (cf. Date 2004: ch. 19). What does a NULL mean? Is the field irrelevant in this record? Should the NULL be interpreted as some default value for the field? Has a value simply not been entered yet? Or did the analyst try to determine the correct value but could not, because the source grammar does not cover the subject, or perhaps because complexities in the language under study make an answer impossible? We recommend that you choose specific values for each of these circumstances (as applicable), make them as explicit as possible, and explain their meaning in more detail in the database documentation. For example, NULL should mean not yet looked at, *irrelevant* can mean that the question does not apply, another value can mean "looked at but could not find the proper value." This will facilitate further data entry in the database, and will allow the TDS to provide this information to the end user.

7. Uncertainty

Values cannot always be assigned with complete certainty. Uncertain values are better than nothing, but it is good practice to make the uncertainty explicit. A common stratagem is to append a question mark to the value, or to put it between brackets. Sometimes comments are also added to the value. But this is a poor strategy from the perspective of database design, since the variant codes look like completely different values to the DBMS (for example, "X" and "X?"), and since it requires that the values be declared as text fields (rather than boolean or other enumerated types). A better strategy would be to have a separate field encoding the degree of certainty, but this might be impractical if it would need to be replicated for very many data fields. For the purposes of importing a database into the TDS, a simple and consistent means of marking uncertainty works best (e.g., by appending a question mark). The TDS can split up such construc-

tions into the value proper and the uncertainty value. Elaborate embedded notation for values is difficult and error-prone for database contributors to enter, and for the TDS to parse. In any event, it is important that any notational conventions be clearly documented.

10 Conclusions

The TDS approach to data integration is based on the principle that semantic integration of complex typological data from different sources is both impossible and undesirable. Impossible because the different typological data collections differ in subtle and complex ways; and undesirable because many such differences are inextricably linked to particular theoretical conceptualizations, which are the goal and end-result of typological research itself. Accordingly, the TDS data management schema is designed to accommodate, side by side, alternative ways of organizing and describing a semantic domain. This approach does not preclude full semantic unification where it is possible (for example, where the differences are primarily notational), but it allows data integration to be carried out without it. Instead of focusing on unification, primacy is given to preserving the explicit and implicit knowledge in the component databases, notably through the addition of detailed interpretive metadata in consultation with the creators of the component databases.

The databases currently included in the TDS represent a wealth of painstakingly collected typological information. The TDS allows for the first time their diverse contents to be examined side by side, and to be interpreted with the aid of descriptive documentation solicited and organized by the Project members. Its export-oriented output formats allow data to be transferred to external applications.

Internally, the "hybrid" approach to knowledge representation provides a global ontology of unifying linguistic Concepts (which still accommodate multiple theoretical perspectives), plus multiple local ontologies relating database fields to global Concepts, and to each other. Since the contents of each database are only related to the global ontology, this structure ensures that the Project can grow in scale without management becoming impractical (as it would if each new database needed to be related to all of the existing ones), and that sufficient breadth and depth of coverage is available to support querying.

The TDS does not assume responsibility for the correctness of the data included in its component databases, but for reporting it faithfully. The degree of confidence that users will place in the system will depend on the ability of the TDS to accurately reflect the analyses in the component databases. The provenance of all information is made visible at every stage of user interaction (searching, query construction and results display), allowing users to assess for themselves the reliability of the information and the theoretical perspective and assumptions it rests on. The result, we hope, is a system suitable for a research community increasingly interested in tools that will support empirically grounded linguistic typology, whose findings can be verified and reproduced.

TDS glossary

Concept	*When capitalized:* An entry in the TDS Global Ontology (TDS-GO).
DTL	Data Transformation Language. A *domain-specific language* created by the TDS Project. Provides a high-level declarative notation for transformation rules and metadata annotations that express the semantics of the resulting data.
ISO code	A unique three-letter code identifying a language according to the ISO 639-3 standard. A successor to the three-letter "SIL codes" used in the Ethnologue directory.
Metadata	Information about the format or meaning of a data field or collection of fields, intended to aid people or computers in its interpretation.
Notion	*When capitalized:* A node in the integrated data hierarchy of the TDS, representing a database field, a localized linguistic term or value, or a group of Notions.
Ontology	A formal representation of a set of concepts within a domain, and of the relationships between those concepts.
OWL	Web Ontology Language. A knowledge-representation language for ontologies.
Root Notion	The root (top-level node) of a tree in the TDS data hierarchy.

TDS-GO	The TDS Global Ontology of linguistic concepts.
Top Notion	A Notion that forms the root (top-level node) of a coherent semantic context.
UPPC	The Universal Phoneme Position Chart, a table of possible phonological segments organized for viewer-friendly presentation.

References

Bechhofer, Sean, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider and Lynn Andrea Stein

2004 OWL Web Ontology Language Reference. vol. 2005. World Wide Web Consortium. http://www.w3.org/TR/owl-ref/.

Chaffin, R, D. J Herrmann and M Winston

1988 An empirical taxonomy of part-whole relations: Effects of part-whole type on relation identification. *Language and Cognitive Processes* 3(1):17-48.

Comrie, Bernhard

1978 Ergativity. In *Syntactic Typology: Studies in the phenomenology of language*, edited by Winfred P. Lehmann. University of Texas Press, Austin.

1989 *Language Universals and Linguistic Typology*. 2 ed. Blackwell, Oxford.

Corbett, Greville G.

1998 Morphology and Agreement. In *The Handbook of Morphology*, edited by Andrew Spencer and Arnold M. Zwicky, pp. 191-205. Blackwell, Oxford, UK; Malden, Mass.

Date, C.J.

2004 *An Introduction to Database Systems*. 8th ed. Addison-Wesley.

Dimitriadis, Alexis, Adam Saulwick and Menzo Windhouwer

2005 Semantic relations in ontology mediated linguistic data integration. In *E-MELD Workshop on Morphosyntactic Annotation and Terminology: Linguistic Ontologies and Data Categories for Linguistic Resources (E-MELD 2005)*, Cambridge, Massachusetts.

Dixon, Robert M. W.

1972 *The Dyirbal language of North Queensland.* Cambridge Studies in Linguistics 9. Cambridge University Press, London.

Gómez-Pérez, Asunción, Mariano Fernández-López and Oscar Corcho

2004 *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer, London.

Gruber, Thomas R.

1993 Toward principles for the design of ontologies used for knowledge sharing. In *International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation*, edited by N Guarino and R Poli. Kluwer Academic Publishers, Deventer.

Gruber, Thomas R. and G. Olsen

1994 An ontology for Engineering Mathematics. Paper presented at the Fourth International Conference on Principles of Knowledge Representation and Reasoning, Bonn, Germany.

Haspelmath, Martin

2005 Argument marking in ditransitive alignment types. *Linguistic Discovery* 3(1).

Hengeveld, Kees, Jan Rijkhoff and Anna Siewierska

2004 Parts-of-speech systems and word order. *Journal of Linguistics* 40(3):527-570.

Kager, René

1999 *Optimality theory*. Cambridge textbooks in linguistics. Cambridge University Press, Cambridge, U.K.; New York.

Ladefoged, Peter and Ian Maddieson

1996 *The Sounds of the World's Languages*. Blackwell Publishing Limited, Oxford, OX, UK ; Cambridge, Mass., USA.

Landis, T. Y., D. J. Herrmann and R. Chaffin

1987 Development differences in the comprehension of semantic relations. *Zeitschrift für Psychologie* 195(2):129-139.

Lewis, William D.

2006 ODIN: A Model for Adapting and Enriching Legacy Infrastructure. In *Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*.

Maddieson, Ian

1984 *Patterns of sounds*. Cambridge Studies in Speech Science and Communication. Cambridge University Press, Cambridge.

Monachesi, Paola, Alexis Dimitriadis, Rob Goedemans, Anne-Marie Mineur and Manuela Pinto.

2002 A Unified System for Accessing Typological Databases. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 3)*, Las Palmas, Canary Islands, Spain. Paris: ELRA.

Nespor, Marina and Irene Vogel

1986 *Prosodic Phonology*. Studies in Generative Grammar. Foris, Dordrecht, Holland/Riverton, USA.

Payne, Thomas Edward

1997 *Describing morphosyntax: a guide for field linguists*. Cambridge University Press, Cambridge, U.K.; New York, NY.

Rector, Alan and Chris Welty

2005 Simple part-whole relations in OWL Ontologies. W3C Editor's Draft 24 Mar 2005 ed. vol. 2005. World Wide Web Consortium. http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/.

Rosch, Eleanor and Barbara B. Lloyd

1978 *Cognition and categorization*. L. Erlbaum Associates distributed by Halsted Press, Hillsdale, N.J. New York.

Saulwick, Adam, Rob Goedemans, Alexis Dimitriadis and Menzo Windhouwer

2006 Architecture and procedures for the integration of linguistic databases in the TDS. In *28. DGfS, AG 6 - Language Archives: Standards, Creation and Access*.

Saulwick, Adam, Menzo Windhouwer, Alexis Dimitriadis and Rob Goedemans

2005 Distributed tasking in ontology mediated integration of typological databases for linguistic research. In *International Workshop on Data Integration and the Semantic Web (DISWeb'05)*, edited by J. Castro and E. Teniente, pp. 13. vol. Proceedings of the CAiSE'05 Workshops. Springer, Porto.

Siewierska, Anna

2004 *Person*. Cambridge textbooks in linguistics. Cambridge University Press, New York.

Storey, Veda C.

1993 Understanding Semantic Relationships. *VLDB Journal* 2:455-488.

Stuckenschmidt, Heiner and Frank van Harmelen

2005 *Information Sharing on the Semantic Web*. Advanced Information and Knowledge Processing. Springer, Berlin.

Taylor, John R.

1989 *Linguistic categorization: prototypes in linguistic theory*. Clarendon Press; Oxford University Press, Oxford [England], New York.

Varela, Francisco J., Evan Thompson and Eleanor Rosch

1991 *The embodied mind : cognitive science and human experience*. MIT Press, Cambridge, Mass.

Winston, M. E., R. Chaffin and D. J. Herrmann

1987 A taxonomy of part-whole relations. *Cognitive Science* 11(4):417-444.