The Lextype DB: A Web-Based Framework for Supporting Collaborative Multilingual Grammar and Treebank Development

Chikara Hashimoto[1], Francis Bond[2,⋆], and Dan Flickinger[3]

[1] Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, Japan 606-8501
ch@yz.yamagata-u.ac.jp
[2] Natural Language Research Group, NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan 619-0237
bond@nict.go.jp
[3] CSLI, Stanford University
Stanford CA 94305-2150 USA
danf@csli.stanford.edu

Abstract. We have constructed a web-based framework for collaborative multilingual grammar and treebank development in which developers are distributed around the world. It is important for developers of the world-wide collaboration to **i)** grasp and share the big picture of the grammar and treebank of each language and **ii)** understand commonalities of languages. Our framework, the Lextype DB, describes lexical types of the grammar and treebank. Lexical types can be seen as detailed parts-of-speech and are the essence for the two important points just mentioned. Information about a lexical type that the Lextype DB provides includes its linguistic characteristics; examples of usage from a treebank; the way it is implemented in a grammar; and correspondences to major computational dictionaries. It consists of a database management system and a web-based interface, and is constructed semi-automatically. Currently, we have applied the Lextype DB to grammars and treebanks of Japanese and English.

Keywords: Multilingual Grammar, Multilingual Treebank, Collaborative Development, Documentation, Web-based Technology.

1 Introduction

Treebanks constructed with detailed linguistic information play an important role in various aspects of natural language processing; for example, grammatical knowledge acquisition; world knowledge acquisition [1]; and statistical language model induction [2,3]. Such treebanks are typically semi-automatically constructed by a linguistically rich computational grammar. A detailed grammar

⋆ Current Affiliation: NICT Computational Linguistics Group.

in turn is a fundamental component for **precise** natural language processing. It provides not only detailed syntactic and morphological information on linguistic expressions but also precise structural semantics, which can be used in, for example, machine translation [4].

The Deep Linguistic Processing with HPSG Initiative (DELPH-IN)[1] has been constructing open-source linguistically precise grammars and treebanks for several languages, including English [5], Korean [6], German [7], Spanish, French, Norwegian and Japanese [8]. DELPH-IN grammars are compatible in that they are all based on the same formalism, which is Head-driven Phase Structure Grammar (HPSG) [9,10], and can be used by the same processors. The semantics are based on Minimal Recursion Semantics [11], a shallow semantic representation that allows for underspecification of scope.

Developers of DELPH-IN are distributed all over the world and are contributing their expertise in linguistics to the DELPH-IN grammar and treebank construction via the Internet. Most grammars are developed along with one or more treebanks of examples. The grammars and treebanks are then available for download, either as snapshots or through CVS.

One of the purposes of DELPH-IN is to capture commonalities of human languages in the course of the development. Capturing commonalities (or universality) of human languages is not only of interest for theoretical linguistics but also an aid to multilingual grammar development since it makes existing grammars more compatible and developing a new grammar much easier. The universal core of the DELPH-IN grammars is codified in the Matrix [12], a bottom-up approach to building a universal grammar.

However, grammars and treebanks are getting more complicated and hard to maintain in the course of development. This is brought about by two factors; one is the linguistically sophisticated nature of DELPH-IN grammars and treebanks, and the other is involved with difficulties in communication during the collaboration over the Internet. The former comes about as a natural result of hand-crafting large-scale HPSG grammars and treebanks for practical NLP purposes, and the latter is a natural consequence of a collaboration in which participants are located away from each other, speak different languages and have different backgrounds and interests.

At this point, the Lextype DB comes on stage. It plays two roles; one is to automatically summarize a grammar and a treebank for each language in terms of lexical types, and the other is to show the summary to developers around the world through the Web. With the Lextype DB, a developer can grasp the big picture of the grammar and the treebank no matter how large they are, and developers distributed over the world can share the big picture.

In addition to alleviating the complication of a large-scale grammar and treebank, the Lextype DB helps to facilitate the understanding of commonalities of human languages. This is because the Lextype DB reveals the linguistic essence of a grammar in terms of lexical types for whatever language it deals with. If developers who are in charge of a particular language show the linguistic essence

[1] `http://www.delph-in.net/`, `http://wiki.delph-in.net/`

of the grammar by means of the Lextype DB, other developers can compare it with their own grammars easily through the Web.

Accordingly, the relations between the Lextype DB and the intercultural collaboration are twofold:

1. The Lextype DB facilitates the DELPH-IN collaboration, members of which come from many cultural (or linguistic) backgrounds. Thus, it facilitates the intercultural collaboration.
2. Also, it facilitates the understanding of commonalities of human languages, which is indispensable for intercultural understanding. This is another way of facilitating the intercultural collaboration.

The next section illustrates what a lexical type is in detail and how helpful it is to understand lexical types for collaborative grammar and treebank development. In the Section 3, we describe what contents the Lextype DB provides and how it is used in the grammar and treebank development collaboration. In Section 4, we discuss how it can be useful as a general lexical resource. After describing related and future work in Sections 5 and 6, we conclude the paper in the final section.

2 A Lexical Type in Grammar and Treebank Development

A lexical type in a DELPH-IN context refers to a group of lexical items whose linguistic behavior is the same as each other. Thus, you can think of a lexical type as a kind of a part-of-speech, but a lexical type is usually a finer-grained notion.

For example, the DELPH-IN grammar of Japanese distinguishes several usages of the Japanese dative marker *ni*. The Japanese sentence (1) can represent the two meanings described in (1a) and (1b). Lexical type names for each usage of *ni* are written in `typewriter` font.[2]

(1)	hanasiai-wa	sinya-**ni**	itaru
	discussion-TOP	midnight-DAT	reach

a. "The discussion comes (to a conclusion) at midnight."
   ni as `adv-p-lex-1`
b. "The discussion continues until midnight."
   ni as `ga-wo-ni-p-lex`

The dative phrase, *sinya-ni* (midnight-DAT), can act as either an adjunct (1a)[3] or an object of *itaru* "reach" (1b). Clearly, these two usages of *ni* show differences in both syntax and semantics. Below is an example showing other usages of *ni*.

[2] These are actual names of the lexical types implemented in the Japanese grammar and might not be understandable to people in general.

[3] The object, *a conclusion*, is unexpressed, the so-called "pro-drop" phenomenon.

(2) Ken-wa yuka-o kirei-**ni** migaku
-TOP floor-ACC clean-DAT polish

a. "Ken polishes a floor clean."
(The floor is clean.)
ni as `naadj2adv-end-lex`

b. "Ken cleanly polishes a floor."
(His way of polishing the floor is clean.)
ni as `adv-p-lex-6`

The dative phrase, *kirei-ni* (clean-DAT), is used as an adjunct in both (2a) and (2b), but their usages and meanings are different. The usage in (2b) is an ordinary adverb that describes the manner of Ken's polishing the floor as clean, while in (2a) the dative phrase describes the resulting situation of the floor after polishing as clean. In addition, the *ni*s in (1) and (2) are different in that the former takes nouns as its complement while the latter takes adjectives. Thus, the four usages in (1a), (1b), (2a) and (2b) must be distinguished so that we can obtain correct syntactic structures and semantic representations. In our terms, these *ni*s are said to belong to different lexical types.[4] Similarly, the Japanese grammar distinguishes usages of other words, notably functional ones.

Although the details of the lexical types are language specific, the differences in the semantics (argument versus modifier in (2) and resultative versus modifier in (3)) appear in all languages. Even the syntactic similarities will be similar in closely related languages, such as Korean.

There are several kinds of linguistic notion implemented in DELPH-IN grammars and treebanks other than lexical types: rules, principles, and root nodes, among others. However, we chose the lexical type as a first step toward a full support of the world-wide collaboration of grammar and treebank development for several reasons.

Firstly, HPSG is a lexical grammar, where lexical elements and phrases are associated with categories that have considerable internal structure. A typical grammar will have many more lexical types than it has construction specific rules or rule schemata.

Secondly, in treebank annotation, the most frequent operation is judging the correct lexical type for each lexical item in each sentence in the treebank. Thus, treebank annotators must be familiar with all the lexical types so that they would know which word usage is correct for a lexical item at hand.

Thirdly, in grammar development, the most frequent operation is adding lexical items. Then, developers have to see which lexical type each lexical item should belong to, and they must be familiar with all the lexical types implemented in a grammar. Otherwise, they might add a new lexical type that has an overlapping functionality with an existing lexical type. This causes spurious ambiguity. And the grammar will be unnecessarily bloated, and the treebank will be easily inconsistent.

[4] Usages of the Japanese dative marker, *ni*, are extensively discussed in, for example, [13].

Finally, lexical types are not so theory specific, so can be used to link to external linguistic resources.

As for human language commonality, other linguistic notions like rules and principles show more commonalities than lexical types in general. However, lexical types reveal commonalities of a language that are much easier to understand for developers who are not familiar with the language. Usually, commonalities related to rules and principles are harder to understand.

In summary, a lexical type, which is dealt with most frequently in the grammar and treebank development, is usually such a complicated notion that developers around the world can hardly grasp and share the big picture of a grammar.

3 Architecture of the Lextype DB

The Lextype DB is required to alleviate the complication of lexical types implemented in a grammar so that developers around the world can access the information. Detailed information about lexical types is, of course, available in the grammar development environment (the LKB[14]). However, to see this, the grammar must be loaded. This makes it difficult to look at, for example, Korean and English grammars while developing Japanese. Further, the grammar development environment is not integrated with the treebank, so it is not easy to go from the types to their usage examples.

3.1 Content of the Database

First of all, what information should be included in such a database to help treebank annotators and grammar developers to work consistently? Obviously, once we construct an electronic lexicon, whatever information it includes, we can easily see what lexical types are assumed in the grammar and treebank. But we have to carefully consider what to include in the database to make it clear how each of the lexical types are used and distinguished.

We include five kinds of information:

(3) Contents of the Database
- a. Linguistic discussion
  - i Name
  - ii Definition and linguistic discussion
  - iii Criteria to judge a word as belonging to a given lexical type
- b. Exemplification
  - i Words that appear in a treebank
  - ii Sentences in a treebank that contain the words
- c. Implementation
  - i The portion of grammar source file that corresponds to the usage
  - ii Comments related to the portion
- d. Links to other lexical resources

That is, we describe each lexical type in depth (3a–3c) and show correspondences to other computational dictionaries (3d).

Linguistic Discussion. To understand lexical types precisely, linguistic observations and analyses are a basic source of information.

Firstly, the requirements for naming lexical-types in a computational system (3ai) are that they be short (so that they can be displayed in large trees) and easily distinguishable. Type names are not necessarily understandable for anyone but the developers, so it is useful to link them to more conventional names. For example `ga-wo-ni-p-lex` is a *Case Particle*.

Next, the definition field (3aii) contains a widely accepted definition statement of the lexical type. For example, `ga-wo-ni-p-lex` (1b) can be defined as "a particle that indicates that a noun it attaches to functions as an argument of a predicate. Individual particles are distinguished by their PFORM." Users can grasp the main characteristics from this. Links to representative papers or books dealing with the lexical type can also be added here. This allows the grammar developers to quickly check against existing analyses, and allows users to find more information.

Thirdly, the criteria field (3aiii) provides users with means of investigating whether a given word belongs to the class. That is, it provides positive and negative usage examples. By such usage examples, developers can easily find differences among lexical types. For example, `adv-p-lex-1` (1a) subcategorizes for nouns, while `adv-p-lex-6` (2b) subcategorizes for adjectives. Sentences like (1a) and (2b) that fit such criteria should also be treebanked so that they can be used to test that the grammar covers what it claims. This is especially important for regression testing after new development.

Exemplification. Examples help users understand lexical types concretely. As we have constructed a treebank that is annotated with linguistic information, we can automatically extract relevant examples exhaustively. We give the database two kinds of examples: words, that are instances of the lexical types (3bi), and sentences, treebanked examples that contain the words (3bii). This link to the linguistically annotated corpus examples helps treebankers to check for consistency, and grammar developers to check that the lexical types are grounded in the corpus data.

For the Japanese grammar we link to the Hinoki treebank[15], and for the English grammar to the Redwoods treebank[3]. DELPH-IN grammars are moving towards including treebanks with the grammars, updated with each release. Therefore, there will always be some usage examples available to the Lextype DB.

Implementation. Grammar developers need to know the actual implementation of lexical types (3ci). Figure 1 shows the implementation of `naadj2adv-end-lex`.

In addition to the actual implementation shown above, we currently show its parent type or types, category of the head, which is, "SYNSEM|LOCAL|CAT|HEAD" in HPSG terms, valence information, which is "SYNSEM|LOCAL|CAT|VAL," and the semantic type, which is "SYNSEM|LOCAL|CONT." This is not always visible in the type's definition, as it may be inherited from a supertype.

```
naadj2adv-end-lex := lexical_sign-word &
 [SYNSEM j-synsem &
        [LOCAL[CAT[HEAD case-adv_head,
                   VAL obj-arg &
                         [COMPS #comps &
                                [FIRST[OPT - ,
                                       LOCAL[CAT[HEAD na-adj_head,
                                                 VAL.UNSAT +],
                                             CONT[RELS <!#key & [ARG1 #xarg]!>,
                                                  HOOK[LTOP #tophand,
                                                       INDEX #ind]]]]]]],
               ARG-S #comps,
               CONT[HOOK[LTOP #tophand,
                         XARG #xarg,
                         INDEX #ind],
                    RELS <! !>,
                    HCONS <! !>]],
         LKEYS.KEYREL #key & [ LBL #tophand],
         NON-LOCAL[QUE <! !>,
                   AFFIX <! !>]],
  INFLECTED +].
```

Fig. 1. Actual implementation of `naadj2adv-p-end-lex`

Comments about the implementation (3cii) are also helpful to ascertain the current status. Although this section is necessarily framework-dependent information, all project groups that are constructing detailed linguistic treebanks need to document this kind of information. We take our examples from JACY [16], a large grammar of Japanese built in the HPSG framework. As actual implementations are generally incomplete, we use this resource to store notes about what remains to be done.

Links to Other Lexical Resources. This information helps us to compare our grammar's treatment with that of other dictionaries. This comparison would then facilitate understanding of lexical types and extension of the lexicon.

We currently link the Japanese grammar to those of ChaSen [17], Juman [18], ALT-J/E [19] and EDICT [20]. For example, `ga-wo-ni-p-lex` is linked to ChaSen's particle-case_particle-general, Juman's case_particle, and ALT-J/E's adjunct-case_particle-noun/particle_suffix [21]. In general, Jacy makes finer distinctions than ChaSen, Juman or EDICT, and has roughly the same level of granularity as ALT-J/E.

We link the English grammar's lexical types to COMLEX [22].

In addition to these language specific resources, we link the lexical types to GOLD's linguistic ontology.[5] GOLD is not a lexicon, rather it is an upper ontology for descriptive linguistics, providing a set of (possibly universal) linguistic notions. Hence, linking to GOLD is an aid to understand a implemented linguistics grammar and a treebank from a universal grammatical point of view. For example, `ga-wo-ni-p-lex` is linked to GOLD's Postposition.

[5] `http://www.linguistics-ontology.org/gold.html`

3.2 Method of Database Construction

The next question is how to construct such a database. Needless to say, fully manual construction of the database is not realistic, since there are hundreds of lexical types and they change as the grammar is developed. In addition, we assume that we will refer to the database each time we annotate parser outputs to build the treebank and that we develop the grammar based on the treebanking result. Thus the database construction process must be quick enough not to delay the treebanking and grammar development cycles.

To meet the requirement, our method of construction for the lexical type database is semi-automatic; most of the database content is constructed automatically, while the rest must be entered manually. This is depicted in Figure 2.

- Content that is constructed automatically
  - Lexical Type ID (Grammar DB)
  - Exemplification (3b) (Treebank DB)
  - Implementation (3ci,ii) (Grammar DB)
  - Link to Other Lexicons (3e) (OtherLex DB)
- Content that is constructed manually
  - Linguistic discussion (3a)

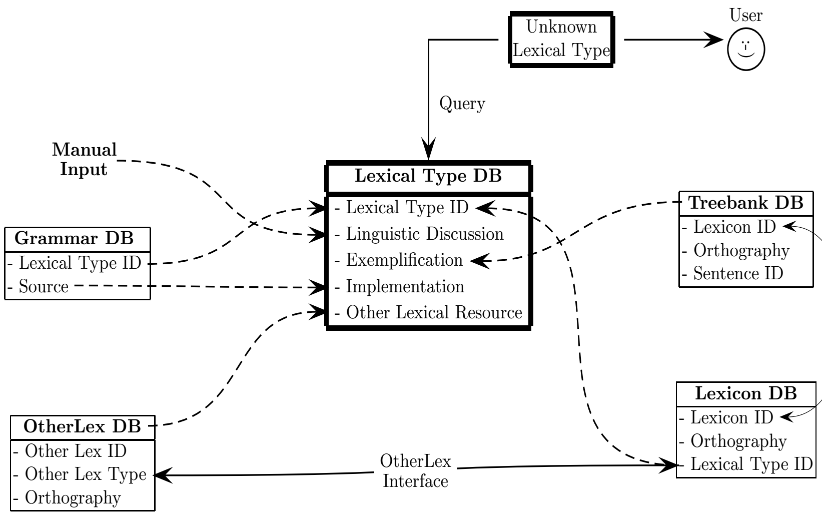

Fig. 2. The lexical type database construction

Component Databases. To understand the construction process, description of the four databases that feed the lexical type database is in order. These are the grammar database, the treebank database, the lexicon database, and the OtherLex database.

- The grammar database contains the actual implementation of the grammar, written as typed feature structures using *TDL* [23]. Although it contains the whole implementation (lexical types, phrasal types, types for principles and so on), only lexical types are relevant to our task.
- The lexicon database gives us mappings between words in the grammar, their orthography, and their lexical types. Thus we can see what words belong to a given lexical type. The data could be stored as *TDL*, but we use the Postgresql lexdb [24], which simplifies access.
- The treebank database stores all treebank information, including syntactic derivations, words, and the lexical type for each word. The main treebank is stored as structured text using the [incr tsdb()] competence and performance profiler [25]. We have also exported the derivation trees for the treebanked sentences into an SQL database for easy access. The leaves of the parse data consist of words, and their lexicon IDs, stored with the ID of the sentence in which the word appears.
- We also use databases from other sources, such as ChaSen, Juman, EDICT and GOLD.

These databases provide a snapshot of the current state of the grammar. They are constructed automatically from the grammar development environment and a series of perl scripts.

Linguistic discussion (3a) has to be entered manually. Linguistic discussion is especially difficult to collect exhaustively since the task requires an extensive background in linguistics. We have several linguists in our group, and our achievements in this task owe much to them. We plan to make the web interface freely accessible, as well as releasing the data and tools as open source to encourage the participation of anyone interested in the task.

The on-line documentation is designed to complement the full grammar documentation [26]. The grammar documentation gives a top down view of the grammar, giving the overall motivation for the analyses. The lexical-type documentation gives bottom up documentation. It can easily be updated along with the grammar.

Each time the grammar is revised based on treebank annotation feedback, grammar developers consult the database to see the current status of the grammar. After finishing the revision, the grammar and lexicon DBs are updated, as are the corresponding fields of the lexical type database. Each time the treebank is annotated, annotators can consult the database to make sure the chosen parse is correct. Following annotation, the treebank DB is updated, and so is the lexical type database. In parallel to this, collaborators who are familiar with linguistics continue to enter relevant linguistic discussions via the WWW.

As much as possible, we are integrating the Lextype DB infrastructure into the existing grammar development environment, so that any grammar developer can take advantage of it with almost no additional effort. To this end, we are now simplifying the software. In particular, we are trying to reduce the number of external dependencies: for example, moving the database from java to perl, which we need for the scripts anyway; and adding routines to dump any information needed directly from the grammar development environment and treebanking tools.

4 Lexical Type Database as a General Linguistic Resource

In this section, we discuss some of the ways the database can enhance collaboration between treebank annotators, grammar developers and other interested parties.

One way is by serving as a link to other lexical resources. As mentioned in the previous section, Lextype DBs are linked together thought the GOLD upper ontology. Each language also includes links to other resources such as EDICT and COMLEX. Many lexical resources have been developed, but their intercorrespondences are not always clear. These lexical resources often play complementary roles, so synthesizing them seamlessly will make a multilingual lexicon with the widest and deepest knowledge ever.

In particular the linguistic notions assumed in GOLD that has been briefly mentioned in §3 act as a "hub" that connects already-connected lexical resources of several languages (Figure 3). Although GOLD's linguistic notions are a bit

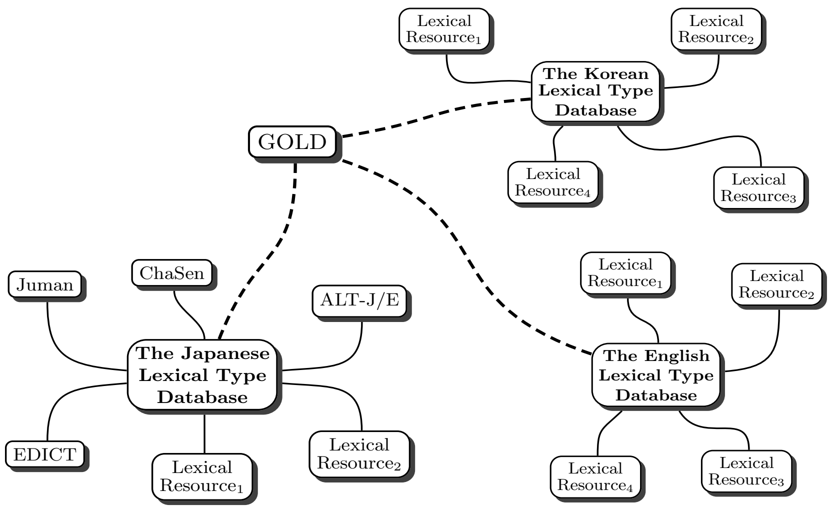

Fig. 3. Synthesis of universal lexical resources

coarse, they would tell us inter-lingual correspondences between lexical types of different languages. Further, the linking of precise lexicons in many languages to GOLD will allow it to be developed further into more specific types.

Apart from NLP, how can the database be used? In the short term our database is intended to provide annotators and grammar developers with a clear picture of the current status of the treebank and the grammar. In the long term, we expect to create successively better approximations of the languages described, as long as the deep linguistic broad coverage grammar describes its language's syntax and semantics precisely. Consequently, the database would be of use to anyone who needs an accurate description of a language. Language teachers can use its detailed descriptions of word usages, the links to other words, and the real examples from the treebank to show students subtle differences among words that look the same but are grammatically different. Lexicographers can take advantage of its comprehensiveness and the real examples to compile a dictionary that contains full linguistic explanations.

The confidence in the linguistic descriptions is based on the combination of the precise grammar linked to the detailed treebank. Each improves the other through the treebank annotation and grammar development cycle, and they are further enhanced by the incorporation of linguistic description of the phenomena.

5 Related Work

The ParGram consortium [27] are producing a family of precise grammars based on Lexical-Functional Grammar. Because the grammars are propriety, there is no attempt to make the documentation freely available. On the other hand the collaboration is close, and the grammars are kept harmonized at biannual meetings.

Earlier work with collaborative development of HPSG grammars has focused on integrating testing (regression testing and treebanking) into the grammar development cycle [28,29]. We are extending this work by (a) linking the grammars to external resources and (b) making the information more accessible. For example, we take the test sets, proposed originally for regression testing, then used for treebanking, and make them available as a corpus, indexed by word and lexical type.

Hypertextual Grammar development [30] attempts to support grammar collaboration by documenting grammars. They suggested creating the documentation in the same file along with the grammar, in the style of *literate programming*[31]. This is an attractive approach, especially for grammars that change constantly. However, we prefer the flexibility of combining different knowledge sources (the grammar, treebank and linguistic description, in addition to external resources).

The Montage project [32] aims to develop a suite of software whose primary audience is field linguists working on underdocumented languages. Among their tasks is to facilitate traditional grammatical description from annotated texts by means of one of their products, the Grammar export tool. Although in the paper there is little explicit detail about what the "traditional grammatical description" is, they seem to share a similar goal with us: in the case of Montage, making grammatical knowledge assumed in underdocumented languages explicit, while in our case making lexical types assumed in the treebank and the computational grammar understandable to humans. Also, some tools they use are used in our project as well. Consequently, their process of grammatical description and documentation looks quite similar to ours. The difference is that their target is underdocumented languages whose grammatical knowledge has so far not been made clear enough, while we target any language whose computational implementation is so large and complex as to be difficult to fully comprehend.

The Language Grid is a new initiative to increase the accessibility and usability of online language services[33]. The Lextype DBs make linguistic information *accessible*, as they are based on the open-source DELPH-IN grammars, and *usable* by abstracting the information in each language's grammar and treebanks and presenting it with a common interface, linked by a standard upper ontology. We hope to take advantage of the Language Grid's language service layer to automatically link the Lextype DBs to existing lexical resources in each language. Having a common API will make it possible to do this automatically each time we create a snapshot. We also hope that the Lextype DBs themselves can serve as useful resources within the grid.

Regarding inter-lexicon connection described in §4, [34] discuss the development of Global Grid, where WordNets of various languages are interconnected, and its challenges. They are more ambitious in that they try to establish mappings between words at the synset level and to understand cultural commonalities and differences that are encoded in languages.

6 Future Work

We are currently experimenting with moving some of the information (in particular the type name and criteria) into the actual grammar files, in the same way as [30]. This would make it easier to keep the information in sync with the actual grammar.

We have created lexical type databases for Japanese and English. However, the importance of such a database certainly holds for any large scale deep grammar. We are now integrating our software more directly into DELPH-IN tools in order to make the Lextype DBs available for groups working with other languages.

Another way we would like to improve the system is to add examples from other corpora by exploiting the links to other resources. For example, if the lexical type link is close enough, we can get examples using the Juman parts of speech, which are exemplified in the Kyoto Corpus [18].

7 Conclusion

We have constructed a web-based framework for supporting collaborative multilingual grammar and treebank development in which developers are distributed around the world. Our framework, which we call the Lextype DB, tells developers around the world about lexical types of the grammar and treebank they are developing. Lexical types can be seen as very detailed parts-of-speech and are the essence for the two important points just mentioned.

We have applied the Lextype DB to grammars and treebanks of Japanese and English, in the near future we plan to make the framework available to other grammar developers, and allow them to create their own Lextype DBs.

References

1. Bond, F., Nichols, E., Fujita, S., Tanaka, T.: Acquiring an Ontology for a Fundamental Vocabulary. In: 20th International Conference on Computational Linguistics (COLING-2004), Geneva, pp. 1319–1325 (2004)
2. Baldridge, J., Osborne, M.: Active learning for HPSG parse selection. In: Proceedings of the 7th Conference on Natural Language Learning (2003)
3. Toutanova, K., Manning, C.D., Flickinger, D., Oepen, S.: Stochastic hpsg parse disambiguation using the redwoods corpus. Research on Language and Computation 3(1), 83–105 (2005)
4. Bond, F., Copestake, A., Flickinger, D., Oepen, S., Siegel, M.: Open source machine translation with delph-in. In: Proceedings of The 10th Machine Translation Summit (2005)
5. Flickinger, D.: On building a more efficient grammar by exploiting types. Natural Language Engineering 6(1), 15–28 (2000)
6. Kim, J.B., Yang, J.: Parsing korean case phenomena in a type-feature structure grammar. In: Gelbukh, A. (ed.) CICLing 2005. LNCS, vol. 3406, pp. 60–72. Springer, Heidelberg (2005)
7. Crysmann, B.: Relative clause extraposition in German: An efficient and portable implementation. Research on Language and Computation 3(1), 61–82 (2005)
8. Siegel, M.: HPSG analysis of Japanese. In: Wahlster, W. (ed.) Verbmobil: Foundations of Speech-to-Speech Translation, pp. 265–280. Springer, Heidelberg (2000)
9. Pollard, C.J., Sag, I.A.: Head-Driven Phrase Structure Grammar. University of Chicago Press, Chicago (1994)
10. Sag, I.A., Wasow, T., Bender, E.M.: Syntactic Theory: A Formal Introduction, 2nd edn. CSLI Publications, Stanford (2003)
11. Copestake, A., Flickinger, D., Pollard, C., Sag, I.A.: Minimal Recursion Semantics. An introduction. Research on Language and Computation 3(4), 281–332 (2005)
12. Bender, E.M., Flickinger, D., Oepen, S.: The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In: Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics, Taipei, Taiwan, pp. 8–14 (2002)
13. Sadakane, K., Koizumi, M.: On the nature of the "dative" particle *ni* in Japanese. Linguistics 33, 5–33 (1995)
14. Copestake, A.: Implementing Typed Feature Structure Grammars. CSLI Publications, Stanford (2002)
15. Bond, F., Fujita, S., Hashimoto, C., Nariyama, S., Nichols, E., Ohtani, A., Tanaka, T., Amano, S.: The Hinoki Treebank — A Treebank for Text Understanding. In: Proceedings of the First International Joint Conference of Natural Language Processing, pp. 554–559 (2004)
16. Siegel, M., Bender, E.M.: Efficient Deep Processing of Japanese. In: Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization, Taipei, Taiwan (2002)

17. Matsumoto, Y., Kitauchi, A., Yamashita, T., Hirano, Y., Matsuda, H., Takaoka, K., Asahara, M.: Morphological Analysis System ChaSen version 2.2.1 Manual. Nara Institute of Science and Technology (2000)
18. Kurohashi, S., Nagao, M.: Building a Japanese parsed corpus — while improving the parsing system. In: Abeillé, A. (ed.) Treebanks: Building and Using Parsed Corpora, pp. 249–260. Kluwer Academic Publishers, Dordrecht (2003)
19. Ikehara, S., Shirai, S., Yokoo, A., Nakaiwa, H.: Toward an MT system without pre-editing – effects of new methods in ALT-J/E–. In: Third Machine Translation Summit: MT Summit III, Washington DC, pp. 101–106 (1991), `http://xxx.lanl.gov/abs/cmp-lg/9510008`
20. Breen, J.: Jmdict: a japanese-multilingual dictionary. In: Coling 2004 Workshop on Multilingual Linguistic Resources (2004)
21. Miyazaki, M., Shirai, S., Ikehara, S.: gengo katēsetsu-ni motozuku nihongo hinshi-no taikēka-to sono kōyō [a Japanese syntactic category system based on the constructive process theory and its use]. Journal of Natural Language Processing 2(3), 3–25, 1995 (in Japanese)
22. Grishman, R., Macleod, C., Myers, A.: COMLEX syntax: Building a computational lexicon. In: 15th International Conference on Computational Linguistics: COLING-94, Kyoto, Japan, vol. 1, pp. 268–272 (1994)
23. Krieger, H.U., Schafer, U.: $\mathcal{TDL}$ — a type description language for constraint-based grammars. In: Proceedings of the 15th International Conference on Computational Linguistics (1994)
24. Copestake, A., Lambeau, F., Waldron, B., Bond, F., Flickinger, D., Oepen, S.: A lexicon module for a grammar development environment. In: 4th International Conference on Language Resources and Evaluation (LREC 2004), Lisbon, vol. IV, pp. 1111–1114 (2004)
25. Oepen, S., Flickinger, D., Toutanova, K., Manning, C.D.: LinGO Redwoods: A Rich and Dynamic Treebank for HPSG. In: Proceedings of The First Workshop on Treebanks and Linguistic Theories, Sozopol, Bulgaria, pp. 139–149 (2002)
26. Siegel, M.: JACY, A Grammar for Annotating Syntax, Semantics and Pragmatics of Written and Spoken Japanese for NLP Application Purposes. ms (2006)
27. Butt, M., King, T.H., no, M.E.N, Segond, F.: A Grammar Writer's Cookbook. CSLI publications, Stanford (1999)
28. Oepen, S., Bender, E.M., Callmeier, U., Flickinger, D., Siegel, M.: Parallel distributed grammar engineering for practical applications. In: Proceedings of COLING 2002 Workshop on Grammar Engineering and Evaluation, Taipei, Taiwan, pp. 15–21 (2002)
29. Oepen, S., Flickinger, D., Bond, F.: Towards Holistic Grammar Engineering and Testing. Grafting Treebank Maintenance into the Grammar Revision Cycle. In: Su, K.-Y., Tsujii, J., Lee, J.-H., Kwong, O.Y. (eds.) IJCNLP 2004. LNCS (LNAI), vol. 3248, Springer, Heidelberg (2005)
30. Dini, L., Mazzini, G.: Hypertextual Grammar Development. In: Computational Environments for Grammar Development and Linguistic Engineering, Madrid, ACL, pp. 24–29 (1997)
31. Knuth, D.E.: Literate Programming. CSLI Publications, Stanford, CA (1992)
32. Bender, E.M., Flickinger, D., Good, J., Sag, I.A.: Montage: Leveraging Advances in Grammar Engineering, Linguistic Ontologies, and Mark-up for the Documentation of Underdescribed Languages. In: Proceedings of the Workshop on First Steps for the Documentation of Minority Languages: Computational Linguistic Tools for Morphology, Lexicon and Corpus Compilation, LREC2004, Lisbon, Portugal (2004)

33. Ishida, T.: Language grid: An infrastructure for intercultural collaboration. In: IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06), pp. 96–100, 2006(keynote address)
34. Fellbaum, C., Vossen, P.T.: Connecting the universal to the specific: Towards the global grid. In: Ishida, T., Fussell, S., Vossen, P.T. (eds.) Intercultural Collaboration I. LNCS, Springer, Heidelberg (2007)