1

Efficient Parsing and Disambiguation for DELPH-IN Grammars

Stephan Oepen

The *Deep Linguistic Processing with HPSG Initiative* (DELPH-IN) is an informal network of practically minded, old-school computational linguists.[1] The name DELPH-IN was coined over dinner at the former Uszkoreit residence at Rothenbühlerweg in Saarbrücken only ten or so years ago, but to me DELPH-IN reflects no less than a common theme throughout three decades of professional work by Hans—the long-range pursuit of genuine language understanding. Among other elements, DELPH-IN combines formal linguistic analysis, constraint-based grammar engineering, algorithms for efficient parsing and generation, preferences between competing analyses and disambiguation, and of course practical applications. Ever since he was a graduate student at the University of Texas at Austin, these are research areas to which Hans has made important contributions. The DELPH-IN network is the fruit of a rich, bi-continental career, the product of a curious and persistent scientist with long-term visions, and of one who also excels at engaging others and creating productive working environments. This chapter attempts to summarize current best practices in parsing and disambiguation for DELPH-IN grammars, i.e. document a selection of techniques that have evolved throghout the years and at present (late

[1]Please see `http://www.delph-in.net/` for more factual background, including a list of participating sites, available technology, and languages covered.

A LaTeX Package for CSLI Collections.
Edie Tor and Ed Itor (eds.).
Copyright © 2010, CSLI Publications.

2010) are at the core of large-scale parsing in DELPH-IN.[2] There is active current work at several sites to further advance DELPH-IN technology, hence the current summary should not be read as a conclusion to this stream of work, but rather as an intermediate snapshot.

1.1 Background: The DELPH-IN Formalism and Tools

As unification-based (or constrained-based) approaches to grammatical description and processing became mainstream in the 1980s (if only temporarily, some might say), a broad variety of descriptive formalisms and mathematical interpretations of the use of (typed) feature structures were in active use—and were more or less tightly coupled to specific linguistic theories, as for example Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994), Lexical Functional Grammar (LFG; Dalrymple et al., 1995), or (Feature Structure) Tree Adjoining Grammar ((F)TAG; Vijay-Shanker and Joshi, 1988). The 1996 final report of the European Expert Advisory Group (EAGLES) on Linguistic Formalisms lists about a dozen implemented grammar development and processing environments (Uszkoreit et al., 1996). While the mathematical foundations for (typed) feature structures maybe were largely established by Rounds and Kasper (1986) and Carpenter (1992), inter alios, variation in the 1990s was primarily in the particular selection of descriptive devices that an individual system made. Open- vs. closed-world reasoning, single vs. multiple inheritance, various approaches to disjunction and negation (in different flavors), set-valued feature structures, the precise semantics of the type system, and the inclusion of implicational or relational constraints are some of the dimensions that, applied to the systems surveyed by EAGLES, for example, make each implementation unique—thus precluding interoperability of linguistic resources.

One accomplishment of the DELPH-IN consortium is its convergence on a common descriptive formalism—a conservative blend of Carpenter (1992), Copestake (1992), and Krieger and Schäfer (1994)—that allows grammars[3] to be processed across different platforms. This joint formalism is by no means the mere intersection (or, loosely speaking, the smallest region of overlap) between the traditions represented among

[2]The work presented here builds on almost twenty years of collaborative, incremental refinement. Numerous colleagues have contributed to this development, and it would seem impossible to properly acknowledge everyone. The selection of references used in this chapter seeks to provide pointers to contributors throughout the DELPH-IN history (and before) and across sites involved.

[3]In the HPSG literature (and accordingly the present article) the term 'grammar' is typically used holistically, referring to the linguistic system comprised of (at least) the type hierarchy, lexicon, and rule apparatus.

participating groups; instead, the selection of formal and descriptive devices was guided by two major concerns: (i) linguistic adequacy, grounded in two decades of joint experience in building large-scale HPSG implementations, and (ii) processing requirements, informed by earlier work on efficient implementations. The DELPH-IN decision to eliminate (explicit) disjunction from the specification language, for example, is motivated by theoretical and engineering considerations alike. Flickinger (2000) argues that a grammatical stipulation that makes disjunctive information explicit in underspecified types in the grammatical ontology (rather than by disjunctive enumeration) provides a stronger model of actual (co-)variation. At the same time, moving to a purely conjunctive feature logic enables the adaptation and fine-tuning of existing, very efficient unification techniques that avoid expensive backtracking and duplication of redundant structure (Tomabechi, 1991, Malouf et al., 2000).

The joint descriptive formalism can be informally characterized as a closed-world, conjunctive-only, multiple inheritance type system that enforces strong typing and strict appropriateness. Types can be associated with arbitrary (complex) constraints that are inherited and applied both at compilation and at run-time (e.g. when two types unify to a more specific, constraint-bearing subtype). HPSG well-formedness principles, immediate dominance schemata, and constituent ordering constraints are all spelled out in the type hierarchy (and cross-multiplied), yielding a set of phrase structure schemata that can be interpreted as rewrite rules over complex (typed feature structure) categories. A mathematical specification of this formalism as it is assumed throughout the volume is provided by Copestake (2000). And although our conservative choice of descriptive devices is fairly restrictive,[4] it has enabled the development of several large grammars as well as the implementation of processing systems that perform with previously unmatched efficiency.

With a development history dating back to the early 1990s, the LinGO English Resource Grammar (ERG; Flickinger, 2000) arguably is both the largest among current DELPH-IN grammars and the most comprehensive HPSG implementation to date. The ERG has continuously evolved through a string of R&D projects (and a handful of

[4]Some of our colleagues half-jokingly characterize the DELPH-IN formalism as 'PATR-III', i.e. an imaginary third revision of the formalism of Shieber et al. (1983). Like PATR-II, the DELPH-IN formalism indeed assumes an explicit phrase structure backbone (rewrite rules of fixed arity, where right-hand side categories are strictly ordered); the main difference, thus, is in the nature of categories in DELPH-IN, which transcends PATR-II noticeably in its use of strong typing and multiple inheritance.

commercial applications) and today allows the analysis of running text across domains and genres. The hand-built ERG lexicon of some 35,000 lemmata aims for complete coverage of function words and open-class words with 'non-standard' syntactic properties (e.g. argument structure). Built-in support for light-weight named entity recognition and an unknown word mechanism combining statistical PoS tagging and on-the-fly lexical instantiation for 'standard' open-class words (e.g. non-relational nouns and adjectives) typically enable the grammar to derive a complete analysis for $85 - - 90\,\%$ of all utterances in standard corpora, for example news-wire, English Wikipedia, or bio-medical research literature (Flickinger et al., 2010, Adolphs et al., 2008). Parsing times for these data sets average around a couple of seconds per sentence, i.e. time comparable to human production or comprehension.

Among the rich assortment of tools in the DELPH-IN open-source repository, two are directly related to our discussion of parsing and disambiguation approaches. The PET parsing system provides an industrial-strength synthesis of best current knowledge within our community; PET was originally developed by Callmeier (2000) and has since been maintained and extended as an open-source project involving several DELPH-IN sites.[5] The so-called [incr tsdb()] environment supports large-scale experimentation in DELPH-IN: test suites, corpora, and treebanks are maintained in a relational database, a standardized API generalizes over 'client' processors (among them PET), and a combination of graphical and batch-oriented tools support, among other tasks, regression testing of the parser, as well as training and evaluation of probabilistic disambiguation models (Oepen, 2003). The following sections focus on core parsing and disambiguation techniques provided by PET and [incr tsdb()], as these take a central (if not exclusive) role in parsing with DELPH-IN grammars. For a specific grammar (and in principle at least also for a token domain, genre, or task), DELPH-IN technology provides a variety of parameters and choice of strategies; core aspects of customization and optimization are discussed (somewhat superficially) where appropriate. Unless otherwise noted, experimental results are reported for the ERG, using the release of October 2010 and software revisions for PET and [incr tsdb()] as provided in the 2010 *Paris* snapshot.

[5]See `http://www.delph-in.net/pet` for background. Current active contributors to the PET code base include Peter Adolphs, Bart Cramer, Bernd Kiefer, Stephan Oepen, and Yi Zhang.

1.2 Parsing: Circumscribing the Space of Analyses

At its core, PET is a classic, agenda-driven chart parser (Kay, 1986), with some of its design decisions rooted in earlier work on implementing efficient unification-based parsing by Erbach (1991), Uszkoreit et al. (1994), and Kiefer et al. (1999). Agenda items are so-called *tasks* (rather than edges), for example the instantiation of a lexical entry, the application of a grammar rule to a passive edge, or the combination of one active and one passive edge. In a strictly bottom-up process (reflecting the lexicalized nature of HPSG), the right-hand sides of rewrite rules (i.e. daughter positions in the edge to be built eventually) are instantiated bi-directionally, where a per-rule specification of a so-called *key* daughter seeks to maximally constrain the applicability of each rule. The key daughter is instantiated first and typically is the right-hand side element that best discriminates amongst the set of possible categories derived by the grammar, often the linguistic head or another daughter that provides comparatively specific information.[6]

The categories of all edges in the chart are typed feature structures (TFSs), and category compatibility is tested by TFS unification. Thus, applying a grammar rule to a passive edge, for example, conceptually means unifying the TFS of the passive edge with one element of the right-hand side of the rule (one daughter position). In practice, both the left- and right-hand sides of rules are represented as a single TFS (thus trivially allowing reentrancy between 'mother' and 'daughter' categories), embedding the right-hand side sequence of categories as the value of a designated, list-valued feature ARGS in the category of the mother. In this scheme, daughter positions can be denoted merely by feature paths into the combined TFS. Where unifying a passive edge into the daughter position of a rule (or active edge) succeeds, a new edge is created whose category will be the unification result. Depending on the arity of the underlying rule, the new edge can either be active (with remaining, open daughter positions) or passive (i.e. complete). This chapter will not discuss the details of TFS manipulation. It is worth noting, however, that PET features a highly optimized unifier, combining the quasi-destructive technique of Tomabechi (1991)

[6]For an earlier version of the ERG (and a comparatively trivial corpus), Oepen and Callmeier (2000) report a doubling of parser efficiency through the identification of optimal key daughters. For a given grammar, this is a purely empirical process that can be aided by the [incr tsdb()] performance profiling tool, which supports accounting of active and passive edges, broken down by individual grammar rules. For a development corpus, the optimal choice of key daughters can be determined by comparing results for variable rule instantiation strategies, where the optimal key position for each rule is the one that generates the smallest number of active edges.

with the subgraph sharing improvements of Malouf et al. (2000) and the compact dag encoding and stack-based memory management of Callmeier (2002). Furthermore, a 'rule filter' (a three-dimensional table compiled statically by attempting to feed each of the grammar rules into all daughter positions of other rules) and the so-called 'quick check' (a pre-unification test of frequently failing feature paths) serve to effectively predict unification failure for a large proportion of parser tasks without actually invoking the full TFS unifier (Erbach, 1991; Kiefer et al., 1999).[7]

An optimization at the interface of the parser proper and the unifier is the technique of so-called hyper-active parsing. Given the large size of feature structures and the computational cost related to memory allocation, Oepen and Carroll (2000b) observe that it can be beneficial to *not* make explicit (i.e. copy after a successful unification) the category associated with an active edge. Some active edges are never used in subsequent parser tasks, and those who do on average only participate in a small number of attempts at instantiating another daughter position. Comparing the relative cost of one (quasi-destructive) unify operation to that of one TFS copy, it can be more efficient to re-do a small number of unifications—viz. those leading to the intermediate category of an active edge—on demand, thus eliminating the need to create an actual copy. Experiments using an earlier version of the ERG and a similar, though less optimized chart parser (implemented in Lisp) showed that around one third of both parsing time and run-time memory consumption can be saved by trading a few extra unifications for fewer copies. Hyper-active parsing today is the default strategy in PET. However, for grammars with larger numbers of right-hand side elements in rules or more generally 'untypical' properties, the beneficial trade-offs observed for the ERG (and some other DELPH-IN grammars) may not apply.[8]

Originally, it was most common to run the PET parser with a search strategy that would rely on the agenda to try and enumerate an n-best lists of results. Not more than about ten years ago, DELPH-IN gram-

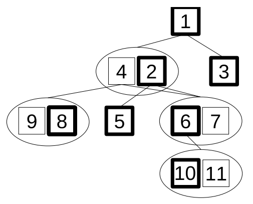
[7]Whereas the static rule filter is computed automatically in PET, premium quick check efficiency depends on a grammar providing a good set of feature paths to be used. Somewhat like the determination of strong key daughters, this problem is best approached in an empirical manner: for a development corpus, accounting of unification failures per path can be provided by PET. Ordering candidate feature paths by frequency of failure, then, the optimal number of paths to be included in the quick check (i.e. the best balance of pre-filtering effort to resulting savings) can be determined in an automated series of experimentation.

[8]While the formalism is agnostic about the maximum arity of rules, a majority of DELPH-IN grammars limit themselves to at most binary rewrite rules for linguistic reasons. This most likely adds to the efficacy of hyper-active parsing.

mars included a heuristic scoring mechanism—based on introspection and manual stipulation by the grammarian—for parser tasks (albeit assuming a different formal foundation, the earlier DELPH-IN approach to disambuation was akin to the use of so-called 'optimality' marks in LFG parsing at the time; Frank et al., 2001). Based on anecdotal evidence, tuning such scoring rules could account for the larger part of grammar adaptation to a specific genre and domain. Fortunately, the DELPH-IN software no longer supports this ill-fated method. Instead, probabilistic parse ranking techniques have been successfully adapted to unification-based parsing and the DELPH-IN universe. If operating PET in greedy n-best search today, the agenda scores are based on a mathematically well-founded statistical model, whose parameters are learned from treebanks (see Section 1.3 below). However, there are two inherent limitations in such n-best bottom-up enumeration of candidate parses: (i) given the specific properties of the probabilistic models used, there is no guarantee that the resulting n-best list will reflect the globally correct rank order; and (ii) it is not quite obvious how to integrate the scoring of agenda tasks in this fashion with the factoring (or packing) of local ambiguities.

In recent years, the predominant approach to combining parsing and disambiguation in PET is thus based on a two-phase approach: first, a complete packed forest is created, where subsumption-based packing and TFS restriction can yield a polynomial observed run-time complexity; second, a specialized graph search procedure is applied to the forest, to selectively enumerate results in exactly the order of their global probabilities. In analogy to context-free parsing, local ambiguity in the chart (i.e. distinct analyses deriving comparable categories for the same sub-string) can be 'packed' (or factored), to prevent the proliferation (i.e. cross-multiplication) of such ambiguities. Formally, edges indexed by sub-string positions in the chart represent nodes in the derivation tree, recording both a feature structure (the category of the node) and the identity of the underlying lexical entry or rule in the grammar.[9] Multiple edges derived for identical sub-strings can be

[9]A frequent source of confusion when discussing ambiguity packing for HPSG is the relationship between feature structures (as complex categories of nodes in the derivation tree) and the representation of derivation history. Textbook HPSG assumes a feature DTRS to record the daughters of a phrase in its TFS (the HPSG *sign*); likewise, the implementation of rewrite rules as a single feature structure in DELPH-IN further blurs the distinction between the local category of a node and its history of derivation. In chart parsing, however, recording derivation history in the categories would be both unnecessary and detrimental to local ambiguity packing. Hence, PET applies feature structure restriction (of the DTRS and ARGS features, as well as of part of the semantics) to the categories stored on each edge (Shieber, 1985).



$$\begin{aligned}
&\boxed{1} \rightarrow \left\langle \boxed{2}\boxed{3} \right\rangle \mid \left\langle \boxed{4}\boxed{3} \right\rangle \\
&\boxed{2} \rightarrow \left\langle \boxed{5}\boxed{6} \right\rangle \mid \left\langle \boxed{5}\boxed{7} \right\rangle \\
&\boxed{4} \rightarrow \left\langle \boxed{8}\boxed{6} \right\rangle \mid \left\langle \boxed{8}\boxed{7} \right\rangle \mid \left\langle \boxed{9}\boxed{6} \right\rangle \mid \left\langle \boxed{9}\boxed{7} \right\rangle \\
&\boxed{6} \rightarrow \left\langle \boxed{10} \right\rangle \mid \left\langle \boxed{11} \right\rangle
\end{aligned}$$

FIGURE 1 Sample (hypothetical) parse forest and sub-node decompositions.

'packed' into a single chart entry in case their feature structures are comparable, i.e. stand in an equivalence or subsumption relation. The latter, less restrictive test takes advantage of the fact that an edge e_1 with a less specific category is guaranteed to derive a superset of edges derivable from an edge e_2 with a more specific category, subsumed by that of e_1. By virtue of having each edge keep back-pointers to its daughter edges—the immediate sub-nodes in the tree whose combination resulted in the mother edge—the parse forest provides a complete and *explicit* encoding of all possible results in a maximally compact form.[10]

Figure 1 shows a hypothetical forest (on the left), where sets of edges exhibiting local ambiguity have been packed into a single 'representative' edge, viz. the one typeset in bold in each set. Ovals in the forest indicate packing of edges under subsumption, i.e. edges $\boxed{4}$, $\boxed{7}$, $\boxed{9}$, and $\boxed{11}$ are *not* in the chart proper. During unpacking, there will be multiple ways of instantiating a chart edge, each obtained from cross-multiplying alternate daughter sequences locally. The elements of this cross-product are pivotal points both for probabilistic scoring and dynamic programming in selective unpacking. The table on the right of Figure 1 shows all non-leaf decompositions for our example packed forest: given two ways of decomposing $\boxed{6}$, there will be three candidate ways of instantiating $\boxed{2}$ and six for $\boxed{4}$, respectively, for a total of nine full trees. Oepen and Carroll (2000a) and Zhang et al. (2007) observe that, for various DELPH-IN grammars, packing under feature structure

This practice resembles the core properties of the linguistic framework proposed by Sag (2010).

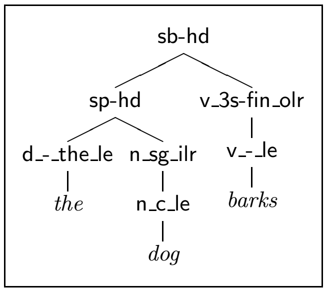
[10]This property of parse forests is not a prerequisite of the chart parsing framework. The basic CKY procedure (Kasami, 1965), for example, as well as many unification-based adaptations (e.g. the Core Language Engine; Moore and Alshawi, 1992) merely record the local category of each edge, which is sufficient for the recognition task and simplifies the search. However, reading out complete trees from the chart, then, amounts to a limited form of search, going back to the rules of the grammar itself to (re-)discover decomposition relations among chart entries.

subsumption is much more effective than packing under mere equivalence. Thus, for each pair of edges (over identical sub-strings) that stand in a subsumption relation, a technique that Oepen and Carroll (2000a) termed retro-active packing ensures that the more general of the two edges remains in the chart. When packing under subsumption, however, parts of the cross-product of local ambiguities in the forest may not be globally consistent. Assume for example that, in Figure 1, edges $\boxed{6}$ and $\boxed{8}$ subsume $\boxed{7}$ and $\boxed{9}$, respectively; combining $\boxed{7}$ and $\boxed{9}$ into the same tree during unpacking can in principle fail. Thus, unpacking needs to deterministically replay unifications, but this extra expense in our experience is negligible when compared to the decreased cost of constructing the forest under subsumption.[11]

Given the parse forest, a naïve unpacking procedure could be obtained from the cross-multiplication of all local combinatorics, which would be amenable to standard dynamic programming. However, the total number of complete analyses can grow exponentially in input length. Assuming a ranking on the space of candidate analyses (see Section 1.3 below), it is often preferable to *not* unpack exhaustively (and then rank) but rather seek to selectively enumerate an n-best list of consistent complete analyses from the forest. Carroll and Oepen (2005) develop a technique dubbed *selective unpacking* for this purpose, originally in the context of surface realization with DELPH-IN grammars. This technique is equivalent to the (final) algorithm of Huang and Chiang (2005), which was developed contemporaneously—both turn out to be reformulations of an approach originally described by Jiménez and Marzal (2000) (albeit for a more restricted class of grammars). At its core, selective unpacking is a specialized search procedure on a weighted and | or graph (also called a hypergraph), where for packed nodes (i.e. disjunctive 'or' nodes) local contexts of optimization are established on demand. Only as much of the local combinatorics is unpacked and evaluated as is actually called for in finding the globally best n complete analyses. Zhang et al. (2007) demonstrate that selective unpacking with DELPH-IN grammars can be implemented efficiently, adding only overhead (for small n) to the cost of constructing the parse forest. They further generalize the algorithm to support a broader class of probabilistic approaches in disambiguation; Section 1.3 below returns to the interplay of selective unpacking and statistical parse selection.

[11]Seeing that the forest construction applies feature structure restriction to improve ambiguity packing, in principle even with equivalence-based packing one would have to resort to original, unrestricted categories anyway; unless of course it could be guaranteed that the features of the restrictor do not actually constrain the space of possible derivations.

October 20, 2010



#	Sample Features
1	⟨0 sb-hd sp-hd v_3s-fin_olr⟩
1	⟨1 △ sb-hd sp-hd v_3s-fin_olr⟩
1	⟨0 sp-hd d_-_the_le n_sg_ilr⟩
1	⟨1 sb-hd sp-hd d_-_the_le n_sg_ilr⟩
1	⟨2 △ sb-hd sp-hd d_-_the_le n_sg_ilr⟩
3	⟨1 n_c_le *dog*⟩
3	⟨2 d_-_the_le n_c_le *dog*⟩
3	⟨3 ◁ d_-_the_le n_c_le *dog*⟩

FIGURE 2 Sample HPSG derivation tree for the sentence *the dog barks*.

1.3 Disambiguation: Choosing among Alternatives

Whereas grammarians may hold all alternate analyses of ambiguous utterances in equal regard, most practical applications of parsing require disambiguation (or ranking) among competing alternatives.[12] Following Abney (1997) and Johnson et al. (1999), parse selection for unification-based grammars is typically approached through discriminative (i.e. conditional) Maximum Entropy (or log-linear) statistical models. Given an utterance s and a set of trees $\{t_1 \ldots t_n\}$ assigned to s by the parser, a Maximum Entropy model is defined through a set of feature functions $\{f_1 \ldots f_m\}$ with corresponding weights $\{\lambda_1 \ldots \lambda_m\}$. Thus:

$$p(t_i|s) = \frac{\exp \sum_j \lambda_j f_j(t_i)}{\sum_{k=1}^{n} \exp \sum_j \lambda_j f_j(t_k)} \tag{1.1}$$

Feature functions f_j can test for arbitrary structural properties of analyses t_i, and their value typically is the number of times a specific property is present in t_i (both the function f_j and their values are at times referred to as just features).

Toutanova et al. (2005) propose an inventory of features that perform well in HPSG parse selection; currently PET restricts itself to the best-performing sub-set of these, of the form illustrated in Figure 2 (on the right). The left-hand side of the same figure shows the ERG derivation for a short utterance; here, phrasal nodes are labeled with (only) identifiers of grammar rules, and (pre-terminal) lexical nodes with class names for types of lexical entries. Formally, the Maximum Entropy approach allows one to deploy arbitrarily complex and potentially overlapping features of the parse trees (in principle also including

[12]Furthermore, for genuinely large-coverage grammars like the ERG, the full set of analyses can count in the hundreds of thousands or millions for longer utterances, hence would be computationally intractable to enumerate fully.

properties of the TFS categories, i.e. the full HPSG sign). In practice, however, feature design needs to balance the ability of properties to differentiate between 'preferred' and 'dispreferred' analyses, on the one hand, with the availability of training data and thus our ability to estimate the statistics of such properties reliably, on the other hand. In more than five years of recent research and experimentation, feature type # 1 from Figure 2 has proven most successful. These features comprise sub-trees of the derivation of depth one—using the grammar-internal identifiers as node labels—plus optionally a chain of one or more dominating nodes (i.e. levels of grandparents, where $\triangle$ depicts 'pseudo' parents at the root of the tree). If a grandparent(s) chain is present then a feature is called *non-local*, as it exceeds the domain of locality of a single rewrite rule (such features require special treatment in selective unpacking; see below). For expository purposes, our example features include another feature type, *n*-grams over leaf nodes of the derivation (where $\triangleleft$ is an utterance-initial anchor); such features (and several other attempts at 'carving' up an HPSG analysis) can bring additional disambiguation accuracy in some (stand-alone) experiments, though not sufficiently so to warrant support for these features in PET so far.

Training a discriminative parse selection model presupposes training material, i.e. a so-called treebank. To maximize the *conditional* likelihood of the training data—$p(t_i|s)$ for each utterance in the treebank—both the preferred and dispreferred analyses need to be available for the estimation of model parameters, i.e. the computation of feature weights $\{\lambda_1 \ldots \lambda_m\}$. Furthermore, the treebank ideally should disambiguate at the same level of linguistic granularity as is assumed in DELPH-IN grammars; a'classic' resource like the Penn Treebank (Marcus et al., 1993), for example, avoids making quite a number of distinctions assumed in HPSG (including a clear argument vs. adjuct contrast, finer points of subcategorization, NP-internal structure, and others). For these reasons, the most common type of training material for parse selection in DELPH-IN are so-called Redwoods-style treebanks (Oepen et al., 2004; Carter, 1997).

In Redwoods, the treebank is built from full HPSG analyses of a DELPH-IN grammar, coupled with manual annotation of which tree is correct. Annotation in Redwoods amounts to disambiguation among the candidate analyses proposed by the grammar and, of course, analytical inspection of the final result. To make this task practical, a specialized tree selection tool extracts a set of so-called discriminants from the complete set of analyses. Discriminants encode contrasts among alternate analyses—for example whether to treat a word like *record* as nominal or verbal, or where to attach a prepositional phrase modifier. Whereas picking one full complete analysis (among a set of hundreds or thousands of trees) would be daunting (to say the least), the isolated contrasts presented as discriminants are comparatively easy to judge for a human annotator, even with only a limited understanding of grammar internals. The [incr tsdb()] environment has full support for the process of parsing a selection of corpora, recording all (or a sub-set of) analyses, guiding an annotator through discriminant-based tree selection, and recording all resulting information in its database. As annotator decisions are recorded as primary data, it is possible to semi-automatically update a Redwoods-style treebank as the underlying grammar evolves. Oepen et al. (2004) suggest that this update procedure actually contributes very useful information to the grammar release cycle, and substantial HPSG treebanks are indeed maintained with each version for several of the DELPH-IN grammars today. The current ERG, for example, includes some 40,000 annotated utterances in its Redwoods treebank, drawing from a variety of genres and domains.

Given a treebank, the process of training a parse selection model is automated in [incr tsdb()] too. The tool supports experimentation with different data sets and variations of feature types or estimation hyper-parameters—i.e. the search for the best-performing configuration. For numeric optimization, [incr tsdb()] interfaces with a Maximum Entropy Learner (TADM; Malouf, 2002) and ultimately can export the feature types and corresponding weights in a format suitable for online use in PET. The selective unpacking procedure discussed already in Section 1.2 above maximizes Maximum Entropy scores for competing sub-structures and guarantees that a globally correct n-best list is extracted from the parse forest with minimal search. The extension of Zhang et al. (2007) further enables selective unpacking to correctly treat grandparenting (i.e. features with an extended domain of locality), and in principle also lexicalization (features of the form exemplified by type # 3 in Figure 2 above). The concluding Section below provides some indication of current parser and disambiguation performance for the ERG.

1.4 Summary: Lessions Learned This Far

To give an impression of current capabilities (and challenges) in parsing with the ERG, Table 1 provides a summary of grammatical coverage, run-time effiency, and disambiguation efficacy for two of the more complex sub-sets of the Redwoods treebank, viz. the LOGON corpus

Corpus	**Items**	**Coverage**	**Time**	**Accuracy**
LOGON	8573 | 732	96 %	1.66 | 0.22	52 | 83
WeScience	8421 | 810	90 %	5.68 | 0.78	45 | ??

TABLE 1 Coverage, parser effiency, and disambiguation figures for the ERG.

of tourism brochures and the WeScience excerpt of Wikipedia articles (with an average sentence length of 13.5 and 17.6 tokens, respectively, in the test sections). The *items* column indicates the number of utterances in the development and test sections, respectively, of each corpus; remaining numbers are for the test data. Coverage figures at 90 % and above (i.e. the percentage of inputs for which the parser found at least one analysis) on these two corpora is slightly higher than on truly unseen data (the Redwoods approach to treebanking necessarily exposes the grammarian to the data to be annotated); for the full English Wikipedia, for example, parsing coverage was at 85 % (see below); for the GENIA corpus (Tateisi et al., 2005), it is just below 87 %. Still, grammatical coverage figures are quite stable nowadays across a wide variety of genres and domains, which suggests that the ERG core accounts for a high proportion of common syntactic phenomena, and that the built-in mechanisms for light-weight named entities and unknown word handling perform effectively.

Parsing times in Table 1 distinguish between the two phases discussed in Section 1.2 above: construction of the parse forest and selective unpacking of up to 500 ranked analyses; clearly the cost of enumeration from the forest (even for a relatively large n) is dwarfed by the time taken to construct the parse forest. The stark contrast in average parser efficiency between the LOGON and WeScience corpora is in part owed to the difference in average length, in part to a historical artifact: work on LOGON predates the addition of unknown word handling, and the ERG has complete lexical coverage for this data (avoiding the extra non-determinism contributed by unknown word handling). Finally, the quality of parse ranking in Table 1 is quantified in the strictest possible measure, exact match of the HPSG derivation ranked most probable by the Maximum Entropy model against the gold-standard tree recorded in the treebank (a random choice baseline would range at well below ten percent for this metric). The second figure in the *accuracy* column attempts to mediate some of the harshness in this measure, counting a success whenever the correct (i.e. gold-standard) analyses is among the ten-best candidates according to the parse selection model. The observation that the exact right tree is on the ten-best list of the model in more than eighty percent of cases actually suggests quite high disambiguation accuracy.

Finally, a recent experiment applied the ERG and PET to the complete English Wikipedia (Flickinger et al., 2010). Despite the relatively high average cost of parsing a single sentence, it was possible to process a total of some 55 million sentences (or 900 million tokens) in about one week (using a commodity HPC cluster). A manual inspection of a random sample of 1,000 parses (using the principal ERG developer as an expert, if potentially mildly biased judge) of the quality of the top-ranked parse for each input found that about two thirds of analyses were 'complete correct' whereas more than 83 % were judged 'nearly correct'.[13]

On the one hand, it is gratifying to confirm that DELPH-IN technology can scale to comparatively massive amounts of data, where there is ample reason to expect that the ERG at least can deliver a good combination of grammatical coverage (of highly granular analyses) and disambiguation accuracy. On the other hand, average parse times of several seconds per sentence—while sufficient for many applications, potentially including interactive use scenarios—remain a technology barrier to Web-scale application of DELPH-IN parsing technology. While algorithmic, encoding, and implementation advances were in focus in the early DELPH-IN days (when HPSG parse times per sentence were at times measured in minutes rather than seconds), the past decade has seen more work on linguistic scalability (pushing syntactic and lexical coverage to practical levels for open-domain running text, and lowering the cost of grammar engineering for additional languages). From the reflections above, it might seem that it is once again time for members of the DELPH-IN community to put emphasis on improving the core technology in terms of its computational efficiency (where closely related efforts have developed a range of promising techniques in recent years; see Miyao and Tsujii, 2008; Clark and Curran, 2007; inter alios). At the same time, work on improving the portability across domains and genres (particularly in terms of pre-processing, unknown word handling, and disambiguation) has gained importance, as it is becoming plausible to look at a grammar like the ERG as a general-purpose resource,

[13] For a parse to be judged correct, every aspect of the analysis had to be fully adequate, including both syntax and semantics. Those items judged as nearly correct contained one or at most two minor errors which did not materially affect the overall meaning of the utterance; the errors were typically misbracketing within a complex nominal compound, misattachment of a modifying prepositional phrase, or an infelicitous coordination bracketing. If an analysis contained more than two such minor errors, or a more serious error resulting in substantial damage to the meaning of the utterance, the parse was judged to be incorrect.

applicable to quite a range of linguistic data and tasks.

In conclusion, and returning to the spirit of this volume: that DELPH-IN has come this far is in substantial parts owed to the thriving eco-system of the collaboration, its spirit of knowledge and resource sharing, and its long-term vision. These, in turn, strongly reflect the influence of Hans Uszkoreit throughout the DELPH-IN history this far.

References

Abney, Steven P. 1997. Stochastic attribute-value grammars. *Computational Linguistics* 23:597 – 618.

Adolphs, Peter, Stephan Oepen, Ulrich Callmeier, Berthold Crysmann, Dan Flickinger, and Bernd Kiefer. 2008. Some fine points of hybrid natural language parsing. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*. Marrakech, Morocco.

Callmeier, Ulrich. 2000. PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering* 6 (1) (Special Issue on Efficient Processing with HPSG):99 – 108.

Callmeier, Ulrich. 2002. Preprocessing and encoding techniques in PET. In S. Oepen, D. Flickinger, J. Tsujii, and H. Uszkoreit, eds., *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*. Stanford, CA: CSLI Publications.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Cambridge, UK: Cambridge University Press.

Carroll, John and Stephan Oepen. 2005. High-efficiency realization for a wide-coverage unification grammar. In R. Dale and K. F. Wong, eds., *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, vol. 3651 of *Lecture Notes in Artificial Intelligence*, pages 165 – 176. Jeju, Korea: Springer.

Carter, David. 1997. The TreeBanker. A tool for supervised training of parsed corpora. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*. Madrid, Spain.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33 (4):493 – 552.

Copestake, Ann. 1992. The ACQUILEX LKB. Representation issues in semi-automatic acquisition of large lexicons. In *Proceedings of the 3rd ACL Conference on Applied Natural Language Processing*, pages 88 – 96. Trento, Italy.

Copestake, Ann. 2000. Definitions of typed feature structures. *Natural Language Engineering* 6 (1) (Special Issue on Efficient Processing with HPSG):109 – 112.

Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen, eds. 1995. *Formal Issues in Lexical-Functional Grammar*. CSLI Lecture Notes. Stanford, CA: Cambridge University Press.

Erbach, Gregor. 1991. A flexible parser for a linguistic development environment. In O. Herzog and C.-R. Rollinger, eds., *Text Understanding in LILOG*, pages 74 – 87. Berlin, Germany: Springer.

Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6 (1):15 – 28.

Flickinger, Dan, Stephan Oepen, and Gisle Ytrestl. 2010. WikiWoods. Syntacto-semantic annotation for English Wikipedia. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 1665 – 1671. Valletta, Malta.

Frank, Anette, Tracy Holloway King, Jonas Kuhn, and John Maxwell. 2001. Optimality theory style constraint ranking in large-scale LFG grammars. In P. Sells, ed., *Formal and Empirical Issues in Optimality Theoretic Syntax*, pages 367 – 397. Stanford, CA: CSLI Publications.

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*, pages 53 – 64. Vancouver, Canada.

Jiménez, Víctor M. and Andrés Marzal. 2000. Computation of the n best parse trees for weighted and stochastic context-free grammars. In *Proceedings of the Joint International Workshops on Advances in Pattern Recognition*, pages 183 – 192. London, UK.

Johnson, Mark, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic 'unification-based' grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 535 – 541. College Park, MD.

Kasami, T. 1965. An efficient recognition and syntax algorithm for context-free languages. Tech. Rep. 65-758, Air Force Cambrige Research Laboratory, Bedford, MA.

Kay, Martin. 1986. Algorithm schemata and data structures in syntactic processing. In I. B. J. Grosz, K. S. Jones, and B. L. Weber, eds., *Readings in Natural Language Processing*, pages 35 – 70. San Francisco, CA: Morgan Kaufmann.

Kiefer, Bernd, Hans-Ulrich Krieger, John Carroll, and Robert Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 473 – 480. College Park, MD.

Krieger, Hans-Ulrich and Ulrich Schäfer. 1994. *TDL*. A type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 893 – 899. Kyoto, Japan.

Malouf, Robert. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*. Taipei, Taiwan.

Malouf, Robert, John Carroll, and Ann Copestake. 2000. Efficient feature structure operations without compilation. *Natural Language Engineering* 6 (1):29 – 46.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics* 19:313 – 330.

Miyao, Yusuke and J. Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics* 34 (1):35 – 80.

Moore, Robert C. and Hiyan Alshawi. 1992. Syntactic and semantic processing. In H. Alshawi, ed., *The Core Language Engine*, pages 129 – 148. Cambridge, MA: MIT Press.

Oepen, Stephan. 2003. *Competence and Performance Profiling for Constraint-Based Grammars. A new Methodology, Toolkit, and Applications*. Doctoral dissertation, Saarland University, Saarbrücken, Germany.

Oepen, Stephan and Ulrich Callmeier. 2000. Measure for measure: Parser cross-fertilization. Towards increased component comparability and exchange. In *Proceedings of the 6th International Workshop on Parsing Technologies*, pages 183 – 194. Trento, Italy.

Oepen, Stephan and John Carroll. 2000a. Ambiguity packing in constraint-based parsing. Practical results. In *Proceedings of the 1st Conference of the North American Chapter of the ACL*, pages 162 – 169. Seattle, WA.

Oepen, Stephan and John Carroll. 2000b. Performance profiling for parser engineering. *Natural Language Engineering* 6 (1):81 – 97.

Oepen, Stephan, Daniel Flickinger, Kristina Toutanova, and Christopher D. Manning. 2004. LinGO Redwoods. A rich and dynamic treebank for HPSG. *Journal of Research on Language and Computation* 2(4):575 – 596.

Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Chicago, IL and Stanford, CA: The University of Chicago Press and CSLI Publications.

Rounds, W. C. and Robert T. Kasper. 1986. A complete logical calculus for record structures representing linguistic information. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*. Cambridge, MA.

Sag, Ivan A. 2010. Sign-Based Construction Grammar. An informal synopsis. In H. Boas and I. A. Sag, eds., *Sign-Based Construction Grammar*, pages 39 – 160. Stanford, CA: CSLI Publications.

Shieber, Stuart, Hans Uszkoreit, Fernando Pereira, Jane Robinson, , and Mabry Tyson. 1983. The formalism and implementation of PATR-II. In B. J. Grosz and M. Stickel, eds., *Research on Interactive Acquisition and Use of Knowledge*, pages 39 – 79. Menlo Park, CA: SRI International.

Shieber, Stuart M. 1985. Using restriction to extend parsing algorithms for complex feature-based formalisms. In *Proceedings of the 23rd Meeting of the Association for Computational Linguistics*, pages 145 – 152. Chicago, IL.

Tateisi, Yuka, Akane Yakushiji, Tomoko Ohta, and J. Tsujii. 2005. Syntax annotation for the GENIA Corpus. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, vol. 3651 of *Lecture Notes in Artificial Intelligence*, pages 222 – 227. Jeju, Korea: Springer.

Tomabechi, Hideto. 1991. Quasi-destructive graph unification. In *Proceedings of the 29th Meeting of the Association for Computational Linguistics*, pages 315 – 322. Berkeley, CA.

Toutanova, Kristina, Christoper D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation* 3(1):83 – 105.

Uszkoreit, Hans, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, and Stephen P. Spackman. 1994. DISCO — an HPSG-based NLP system and its application for appointment scheduling. In *Proceedings of the 15th International Conference on Computational Linguistics*. Kyoto, Japan.

Uszkoreit, Hans, Tilman Becker, Rolf Backofen, Jo Calder, Joanne Capstick, Luca Dini, Jochen Dörre, Gregor Erbach, Dominique Estival, Suresh Manandhar, Anne-Marie Mineur, Gertjan van Noord, and Stephan Oepen. 1996. The EAGLES Formalisms working group. Final report. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Saarbrücken, Germany.

Vijay-Shanker, Krishnamurti and Aravind Joshi. 1988. Feature Structures Based Tree Adjoining Grammars. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 714 – 719. Budapest, Hungary.

Zhang, Yi, Stephan Oepen, and John Carroll. 2007. Efficiency in unification-based n-best parsing. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 48 – 59. Prague, Czech Republic.